Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jon Natanael Muhovič

Sledenje objektov z generaliziranim Houghovim transformom

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matej Kristan

Ljubljana 2014

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jon Natanael Muhovič, z vpisno številko **63100341**, sem avtor diplomskega dela z naslovom:

Sledenje objektov z generaliziranim Houghovim transformom

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 10. septembra 2014 Podpis avtorja:

Na tem mestu se zahvaljujem mentorju doc. dr. Mateju Kristanu in mag. Luku Čehovinu za uvajanje na področje sledilnih algoritmov ter pomoč pri razumevanju teoretične podlage za moje diplomsko delo. Zahvala gre tudi mojim bližnjim, ki so mi stali ob strani tekom mojega študija na FRI ter članom laboratorija ViCoS, za nudenje produktivnega okolja, v katerem je to delo nastalo.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Motivacija 2

1.2 Sorodna dela 3

1.3 Prispevki 5

1.4 Struktura naloge 5

2 Teorija **7**

2.1 Segmentacija 7

2.2 Houghov transform 13

2.3 Harrisova detekcija oglišč 14

2.4 Kalmanov filter 15

2.5 PixelTrack 18

3 Razširjeni PixelTrack **23**

3.1 Uporaba segmentacije z MRF 23

3.2 Uporaba Harrisove detekcije 24

3.3 Uporaba Kalmanovega filtra 24

3.4 Opis razširjenega algoritma 25

4 Eksperimenti **29**

4.1 Opis testne zbirke 29

4.2 Uporabljeni parametri v sledilniku 30

4.3 Eksperimenti 31

4.4 Diskusija rezultatov 33

5 Sklep **47**

5.1 Nadaljnje delo 48

Povzetek

Sledenje objektov je zelo raznoliko in uporabno področje računalniškega vida. Pristopov k reševanju tega problema je ogromno, namen tega dela pa je predstaviti nekaj metod, ki se uporabljajo za implementacijo naprednih sledilnih algoritmov, analizirati konkreten algoritem, ki sledenje izvaja z uporabo generaliziranega Houghovega transforma ter načrtati in izvesti izboljšave, ki bodo koristile algoritmovi robustnosti in natančnosti. Predlagane izboljšave temeljijo na uporabi Harrisove detekcije oglišč, Kalmanovega filtra in segmentacijskega algoritma, ki deluje na podlagi Markovovih slučajnih polj. Rezultat diplomskega dela je tako izboljšan algoritem, implementiran v C++, z dodanimi metodami, ki mu omogočajo boljše delovanje. Praktični eksperimenti so bili izvedeni v okolju, namenjemu testiranju sledilnih algoritmov, z uporabo raznolikih in zahtevnih video sekvenc. Rezultati eksperimentov jasno prikazujejo izboljšave, ki so jih povzročile predlagane metode.

Ključne besede: sledilni algoritem, generaliziran Houghov transform, sledenje netogih objektov, segmentacija.

Abstract

Visual object tracking is a very diverse and useful area of computer vision. There are many different approaches to solving this problem and the goal of the thesis is to first present some of the methods that are used for implementation of state-of-the-art tracking algorithms. Secondly, the analysis of a concrete algorithm that tracks the object by using generalized Hough transform and lastly to design and implement some enhancements that boost the algorithm's robustness and accuracy. The proposed enhancements are based on Harris corner detection, Kalman filter and a segmentation algorithm that uses Markov random fields. The result of the thesis is thus an improved algorithm, implemented in C++ with added methods that improve its performance. Practical experiments were carried out in a framework designed for testing tracking algorithms by using diverse and difficult video sequences. Experiment results clearly show the improvements caused by the proposed methods.

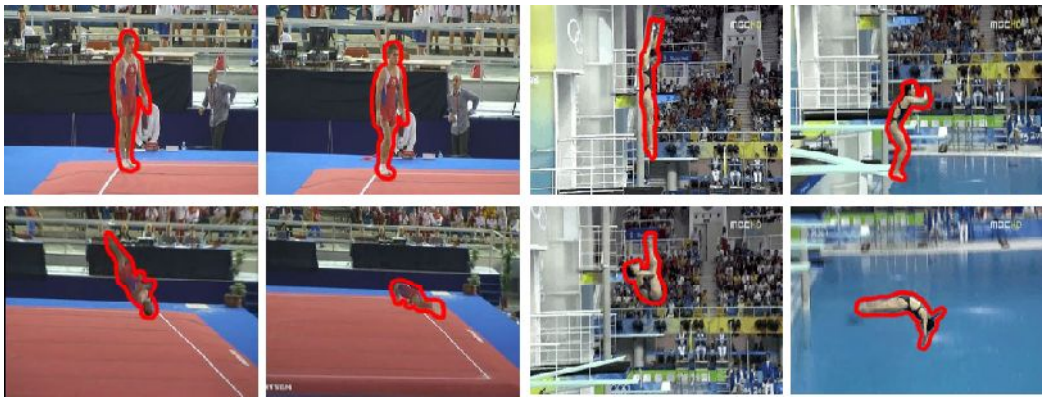
Keywords: tracking algorithm, generalized Hough transform, non-rigid object tracking, segmentation.

Poglavje 1

Uvod

Sledenje objektov je proces lociranja premikajočega se predmeta v času, z uporabo kamere. Sledeni predmet je v vsakem trenutku, torej v vsaki sliki videa, na določeni lokaciji, cilj sledilnih algoritmov pa je pravilna, zanesljiva in hitra določitev koordinat te lokacije. Zahtevnost te naloge je odvisna od lastnosti videa (spremembe barv/osvetlitve, premikanje kamere, prekrivanje sledenega objekta ipd.), lastnosti objekta (ali se deformira, ali je v sliki prisotnih več podobnih objektov ...) in od namena algoritma (kratkoročno/dolgoročno sledenje, morebitno predhodno znanje o objektu). Težje kot so okoliščine in manjše kot je predhodno znanje o objektu, kompleksnejši in robustnejši mora biti algoritem, da uspešno sledi objektu. Sledenje je zaradi široke uporabnosti zelo perspektivno področje, zato se je razvila velika množica različnih algoritmov, ki uporabljajo ogromno različnih pristopov k lociranju objekta (segmentacija [1], konture [2], sledenje značilnih točk [3] ipd.).

Namen uporabe algoritma igra pomembno vlogo pri njegovem razvoju in uporabljenih metodah. Za sledenje objekta s poznano obliko, ki se ne deformira, bi zadostovala enostavna primerjava vizualne predloge objekta s trenutno sliko, a bi naletela na težave ob spremembah osvetlitve ali ob prekrivanju. Spremembam izgleda objekta se zato ponavadi prilagaja z učenjem modela izgleda objekta (angl. visual model), ki se ga med izvajanjem algo-



Slika 1.1: Primeri netogih objektov med premikanjem.

ritma posodablja glede na opažene spremembe.

Poseben razred sledenih objektov predstavljajo objekti, ki so podvrženi t.i. netogim deformacijam (angl. *non-rigid deformations*), kjer objekt ne spreminja le svoje pozicije in kota v razmerju do kamere, ampak se njegova lastna oblika drastično spreminja, kot je videti na Sliki 1.1. Take spremembe sledilni algoritmi rešujejo na različne načine, z uporabo histogramov [4], s sledenjem po delih (angl. *part-based tracking*) [5], s sledenjem zaplat (*patch-based tracking*) [6] ali z uporabo generaliziranega Houghovega transforma [7]. V podpoglavju 1.2 so opisani različni pristopi k sledenju takih objektov. Med njimi zaradi zanimivega kombiniranega pristopa in hitre implementacije posebej izstopa algotirem Pixeltrack [8], ki ga bomo v diplomskem delu tudi podrobneje obravnavali.

1.1 Motivacija

Cilj razvoja sledilnih algoritmov je ustvariti čim bolj splošne in robustne algoritme, ki lahko iz podane inicializacije uspešno preberejo lastnosti objekta in mu natančno sledijo skozi video sekvenco kljub morebitnim motečim faktorjem. Dodatna zahteva za uporabnost takšnih algoritmov je čim večja hitrost, saj so možnosti aplikacije ogromne in večinoma povezane z delova-

njem v realnem času. Tukaj velja omeniti razliko med nepovezanim (angl. offline) in nalinijskim (angl. online) sledenjem: nepovezani sledilni algoritmi delujejo na shranjenih sekvencah in lahko izvedejo več prehodov skoznje, kar jim omogoča natančno delovanje, a za ceno hitrosti. Nelinijski sledilniki pa so zaradi poudarka na hitrosti in delovanju v realnem času omejeni le na informacije v trenutni sliki in inference iz prejšnjih slik, kar naredi nalogo sledenja težjo, a hkrati zelo razširi uporabnost takih algoritmov v primerjavi z nepovezanimi algoritmi. Kot primer uporabe hitrih sledilnih algoritmov omenimo naslednje:

- Video nadzorni sistemi (analiza dogajanja)
- Zaznavanje/sledenje obrazov (npr. v fotoaparatih)
- Obogatena resničnost
- Nadzor prometa (vinjete ipd.)
- Robotika (navigacija, preprečevanje trkov ipd.)
- Analiza športnih dogodkov
- Uporabniški vmesnik (Kinect)

Nalinijske sledilne algoritme dodatno razdelimo na kratkoročne in dolgoročne, kjer kratkoročni sledijo objektu od inicializacije do trenutka, ko objekta ne zaznajo več, torej delujejo zgolj kot sledilniki. Dolgoročni sledilniki pa delujejo tudi, če objekt zapusti vidno polje kamere in v tem času namesto sledenja izvajajo detekcijo ter ob morebitnem uspehu detekcije nadaljujejo s sledenjem. To delo se osredotoča na kratkoročne nalinijske algoritme.

1.2 Sorodna dela

Prejšnji prispevki na področju vizualnega sledenja objektov, kot so metode [9, 4, 10] večinoma opisujejo objekt z mejnim pravokotnikom (angl. boun-

ding box) ali podobno enostavno geometrično obliko in uporabljajo globalni model izgleda. Takšne metode so robustne na določeno mero spremembe izgleda in lokalne deformacije, poleg tega pa jih je mogoče hitro implementirati. Kadar pa so sledeni objekti podvrženi netogim deformacijam in velikim spremembam v izgledu (zakrivanje, spremembe v osvetlitvi ...), mnogokrat odpovedo. S sledenjem kontur lahko nekatere metode [2, 11] to zaobidejo, a za to zahtevajo dodatne pogoje, kot so premikanje objekta ali predhodno znanje o njem. Druge, na primer [3, 12], opišejo objekt z gosto množico značilnih točk, ki jim sledijo v vsaki sliki, a so še vedno omejene na dokaj toge objekte.

Veliko obstoječih metod izhaja iz sledenja na podlagi detekcije, kjer se zgradi diskriminativen model sledenega objekta, ki se posodablja med izvajanjem algoritma in se tako prilagodi spremembam v izgledu. Adam *et al.* [13], denimo, predstavijo model izgleda, ki temelji na zaplatah, z integralnimi histogrami barve in intenzitete. Dinamična predloga, sestavljena iz takšnih zaplat, omogoča prostorsko strukturo in robustnost na delno prekrivanje. Grabner *et al.* [14] so predlagali nalinijski Adaboost algoritem, ki dinamično izbira šibke klasifikatorje, ki diskriminirajo med ospredjem in ozadjem. Kalal *et al.* [15] so v sledilniku TLD (angl. Tracking-Learning-Detection) uporabili slučajne gozdove in jih učinkovito kombinirali z Lucas-Kanade sledilnikom, kjer algoritem posodablja detektor z uporabo prostorskih in časovnih omejitev, detektor pa ponovno inicializira sledilnik, če le-ta izgubi objekt. A vse te metode še vedno operirajo z regijami slik, ki so opisane s pravokotniki in imajo zato težave s predmeti, podvrženimi velikim deformacijam. Da bi se temu izognili, je več avtorjev v proces sledenja vključilo neke vrste segmentacijo. Nejhum *et al.* [1] so predlagali sledenje artikuliranih objektov z množico neodvisnih pravokotnih blokov, ki uporabijo za segmentacijo z algoritmom rez grafov (angl. *graph-cut*). Spet drugi [16] uporabljajo segmentacijo na nivoju superpikslov.

Nekateri algoritmi za sledenje uporabljajo generaliziran Houghov transform, opisan v članku [17], ki je namenjen sledenju objektov poljubnih oblik.

Gall *et al.* [18] opisujejo metodo za detekcijo in sledenje na podlagi slučajnih gozdov (angl. random forests). Njihova metoda deluje na podlagi zaznavanja zaplat v sliki, ki nato glasujejo po Houghovi glasovalni shemi, za določitev središča objekta. Godec *et al.* [7] podoben pristop uporabijo v kombinaciji s segmentacijo. S projekcijo delov, ki so glasovali za center objekta, avtorji inicializirajo algoritem rez grafa (angl. *graph-cut*) za pridobitev segmentacije na objekt in ozadje. Ta segmentacija se nato uporabi za posodobitev verjetnosti, da se zaplata nahaja v ospredju ali ozadju. Njihova metoda doseže zelo dobre rezultate na mnogo zahtevnih sekvencah, a je zaradi algoritma rez grafa relativno počasna. Duffner in Garcia [8] pa predstavita algoritem PixelTrack, ki uporablja generaliziran Houghov transform na nivoju pikslov in hkrati uporablja segmentacijski algoritem na podlagi globalnih modelov ospredja in ozadja. Houghov in segmentacijski model se križno posodabljata za čim boljše rezultate, zaradi učinkovite implementacije pa je algoritem tudi zelo hiter.

1.3 Prispevki

Ključen doprinos diplomskega dela so analiza obstoječega sledilnega algoritma [8], po teoretični in praktični strani, pregled teoretične podlage za razširitev algoritma z namenom doseganja boljših rezultatov, načrt ter izvedba teh izboljšav in izvedba eksperimentov, osnovanih na bazi video sekvenc VOT2013, ki je namenjena evalvaciji delovanja sledilnih algoritmov, ter poročilo o spremembah v delovanju algoritma. Kot del naloge je nastala tudi hitra implementacija segmentacijskega algoritma, ki lahko deluje samostojno in se jo da uporabiti tudi izven konteksta naloge.

1.4 Struktura naloge

Preostanek diplomske naloge je razdeljen na štiri poglavja. V Poglavju 2 predstavimo teoretično podlago algoritma PixelTrack, opišemo njegovo delo-

vanje in teorijo, uporabljeno v izboljšavah algoritma. Poglavje 3 obravnava načrtovanje izboljšav algoritma, uporabljenih metodah in vključitvi v obstoječi algoritem. V Poglavju 4 opišemo testno zbirko VOT2013, na kateri smo preizkusili spremembe v algoritmu, parametre, uporabljene v sledilniku, opažanja med razvojem izboljšav in zaključimo z diskusijo pridobljenih eksperimentalnih rezultatov in analizo vpliva predlaganih sprememb na delovanje algoritma (natančnost, robustnost). Poglavje 5 vsebuje sklep in predlog nadaljnjega dela.

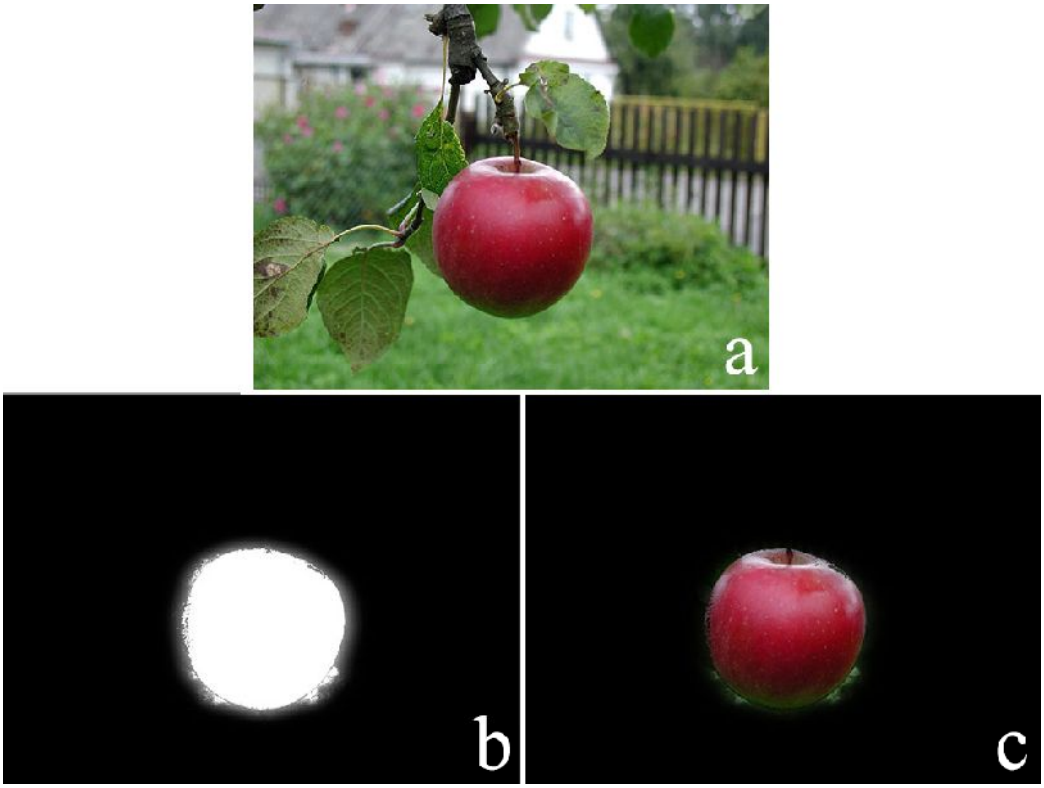
Poglavje 2

Teorija

V tem poglavju opišemo metode, uporabljene v algoritmu PixelTrack in njegovih izboljšavah. V podpoglavju 2.1 predstavimo segmentacijo, podpoglavje 2.2 obravnava (generaliziran) Houghov transform, podpoglavje 2.3 Harrisov detektor oglišč, v podpoglavju 2.4 je opisan Kalmanov filter 2.4, v podpoglavju 2.1.1 pa Markovova slučajna polja.

2.1 Segmentacija

V računalniškem vidu je segmentacija proces razdelitve slike v več segmentov, t.j. vsakemu pikslu določiti oznako tako, da imajo piksli z isto oznako podobne lastnosti, npr. barvo, teksturo ali intenziteto. Taki postopki lahko v mnogih aplikacijah služijo npr. za detekcijo objektov [19], za ločitev različnih organov v medicinskem slikanju [20] ipd. Število razredov je poljubno, glede na zahteve aplikacije. Najenostavnejša klasifikacija je taka, ki proizvede binarno masko, kar prikazuje Slika 2.1. Ob aplikaciji na vhodno sliko takšna maska proizvede ločitev na objekt in ozadje, kar se ob poznavanju lastnosti objekta lahko uporabi za njegovo detekcijo. Zaradi lastnosti nekaterih materialov (perje, lasje) in v želji po kvalitetnejši segmentaciji nekateri algoritmi za vsak piksel poleg klasifikacije v ospredje ali ozadje določijo tudi prosojnost, torej so svetlostne vrednosti pikslov maske na intervalu $[0, 1]$.



Slika 2.1: Slika prikazuje izhodiščno sliko (a), segmentacijsko masko (b) in z masko izrezano vsebino (c).

2.1.1 Markovova slučajna polja

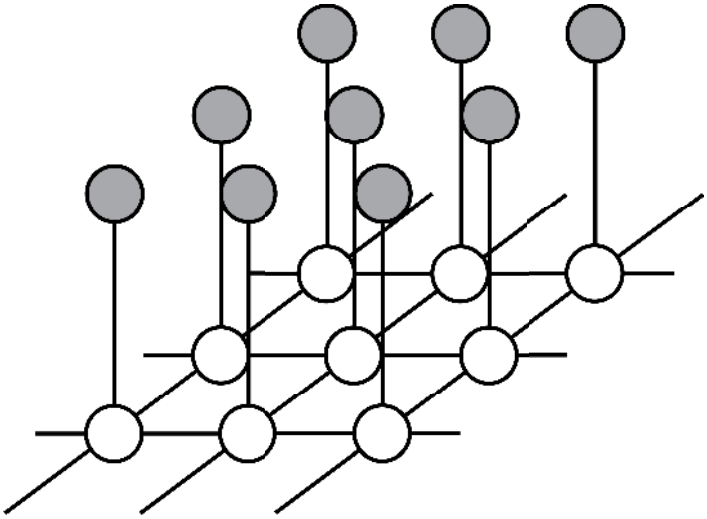
Markovovo slučajno polje (angl. Markov random field, kratica MRF) je množica slučajnih spremenljivk, ki imajo Markovovo lastnost in so opisane z neusmerjenim grafom. Če imamo graf $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, z množico vozlišč $\mathcal{V} = (1, 2, \ldots, i, \ldots, N)$, množico povezav $\mathcal{E}$, kjer je povezava $(i, j), i, j \in \mathcal{V}$ in so povezave neusmerjene, tako da se oznaki (i, j) in (j, i) nanašata na isto povezavo, ter množico slučajnih spremenljivk X_i, kjer vsaka pripada enemu vozlišču $i \in \mathcal{V}$, morajo veljati lokalne Markovove lastnosti:

- Parna Markovova lastnost: katerikoli dve nesosednji spremenljivki sta pogojno neodvisni, če imamo podane vse druge spremenljivke:

$X_u \perp\!\!\!\perp X_v | X_{\mathcal{V}\{u,v\}}$

- Lokalna Markovova lastnost: spremenljivka je pogojno neodvisna od vseh drugih, če imamo podane njene sosede:
  $X_v \perp\!\!\!\perp X_{\mathcal{V}cl(v)} | X_{ne(v)}$, kjer je $ne(v)$ množica sosedov v in $cl(v) = \{v\} \cup ne(v)$

- Globalna Markovova lastnost: katerikoli dve podmnožici spremenljivk sta pogojno neodvisni, če imamo podano razdelilno podmnožico (separating subset):
  $X_A \perp\!\!\!\perp X_B | X_S$, kjer vsaka pot iz vozlišča v A v vozlišče v B poteka skozi S.



Slika 2.2: Markovovo slučajno polje.

Markovova slučajna polja se uporabljajo v procesiranju slik in računalniškem vidu za restavracijo slik [21], segmentacijo [22], superresolucijo [23] ipd. V računalniškem vidu lahko služijo kot predhodni model za množico prikritih slučajnih spremenljivk, ki ležijo pod množico opazovanih vrednosti z. Tak sistem je analogen skritemu Markovovemu modelu, ki ga prikazuje Slika 2.5, le da deluje v 2D prostoru (vsako vozlišče je povezano z večimi sosedi), kot je prikazano na Sliki 2.2

2.1.2 Uporaba MRF pri segmentaciji

Metoda, opisana v članku [22], ki predstavi algoritem za segmentacijo in temelji na Markovovih slučajnih poljih, obravnava apriorne in aposteriorne vrednosti nad labelami razredov kot slučajne spremenljivke, ki tvorijo Markovovo slučajno polje. Algoritem za segmentacijo uporablja optimizacijo s postopkom EM (angl. *expectation maximization*) [24]. Temelji na minimizaciji energijske funkcije, ki opisuje posteriorne vrednosti Markovovega slučajnega polja. Splošna oblika funkcije je prikazana v enačbi

$$E = \lambda_1 \sum_i u_i + \lambda_2 \sum_{i,j \in N} \phi_{i,j}, \tag{2.1.1}$$

kjer je u_i t.i. unarni izraz (torej tak, ki se nanaša na posamezno vozlišče), $\phi_{i,j}$ pa parni (pair-wise) izraz, ki se nanaša na povezave med sosednjimi vozlišči. V parnem izrazu je vključena tudi kazen, ki poveča energijo, če sosednji vozlišči nimata iste vrednosti, kar zviša verjetnost razporeditev skritih vrednosti v konfiguracije, kjer ima mnogo sosednjih pikslov iste vrednosti, oz. kjer je obseg skupin istega razreda kar se da majhen.

Kriterijska funkcija v ozadju uporabljenega segmentacijskega algoritma je prikazana v enačbi

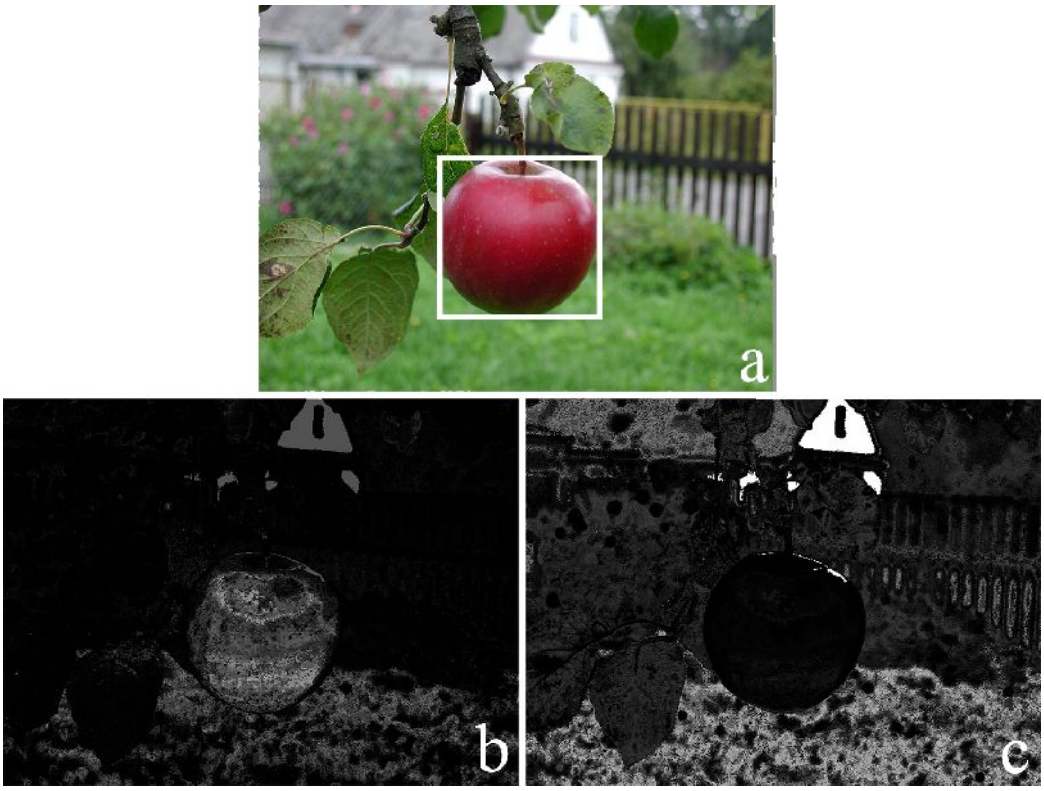
$$F = \sum_{i=1}^{M} [\log p(y_i, \Theta | \varphi_0) - \frac{1}{2}(D(s_i \parallel \pi_i \circ \pi_{N_i}) + D(q_i \parallel p_i \circ p_{N_i}))], \tag{2.1.2}$$

kjer $\circ$ označuje Hadamardov produkt (produkt po komponentah), $\| \cdot \|$ pa divergenco Kullback-Liebler, ki kaznuje razlike med predhodnimi distribucijami nad sosednjimi piksli. V praksi energijsko funkcijo minimiziramo z enačbami

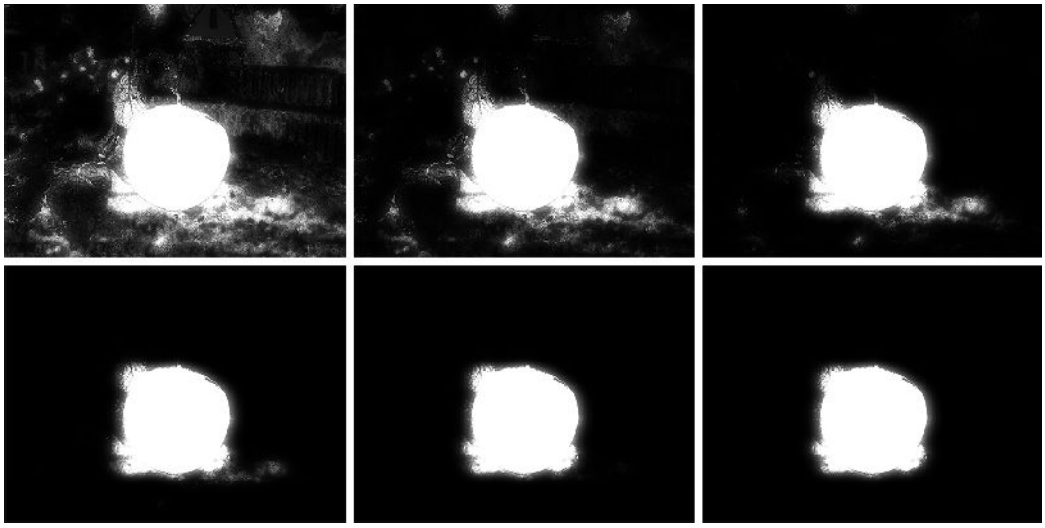
$$\begin{aligned} \hat{s}_{\cdot k} &= (\xi_{s\cdot} \circ \pi_{\cdot k} \circ (\pi_{\cdot k} * \lambda)) * \lambda_1, \\ \hat{q}_{\cdot k} &= (\xi_{q\cdot} \circ p_{\cdot k} \circ (p_{\cdot k} * \lambda)) * \lambda_1, \\ \pi_{\cdot k}^{opt} &= (\hat{s}_{\cdot k} + \hat{q}_{\cdot k})/4. \end{aligned} \tag{2.1.3}$$

Njihova lepa lastnost je, da se jih da učinkovito implementirati s konvolucijo in Hadamardovim produktom, saj izračun predhodne distribucije soseščine

piksla π_{N_i} za i-ti piksel zahteva uteženo kombinacijo predhodnih vrednosti sosednjih pikslov π_j. Če so $\pi_{.k}$ predhodne vrednosti k-te komponente, razporejene v matriko velikosti slike, potem je predhodne vrednosti soseščine mogoče izračunati z naslednjo konvolucijo: $\pi_{N_{.k}} = \pi_{.k} * \lambda$, kjer je λ diskretno jedro (discrete kernel), z osrednjim elementom postavljenim na 0 in vsoto 1. Faktor λ_1 je enak faktorju λ, le da ima osrednji element postavljen na 1. Segmentacijski algoritem kot vhod sprejme projekciji histogramov ospredja in ozadja (projekcijo pridobimo tako, da vsakemu pisklu v sliki dodelimo vrednost, ki jo ima njegova barva v normaliziranem histogramu – Slika 2.3), vrne pa masko, ki loči objekt od ozadja. Vrednosti izhodne maske ležijo na intervalu $[0, 1]$.



Slika 2.3: Slika prikazuje izhodiščno sliko (a) ter projekciji histograma ospredja (b) in ozadja (c).



Slika 2.4: Koraki optimizacije segmentacijske maske; izhodiščna slika je enaka tisti na Sliki 2.3.

Ob inicializaciji segmentacijski algoritem ustvari jedri, v enačbah opisani z λ in λ_1 ter nastavi izhodiščne ocene verjetnosti za ospredje in ozadje (uniformni matriki velikosti vhodnih slik, z vsemi vrednostmi postavljenimi na 0.5). Algoritem deluje iterativno in v vsakem koraku izračuna posteriorne vrednosti pikslov $p_{\cdot k}$, nato pa z uporabo enačb (2.1.3) izračuna še nove apriorne vrednosti pikslov $\pi_{\cdot k}^{opt}$, iz njih pa logaritemsko verjetnost (angl. *log-likelihood*), ki služi kot zaključni pogoj. Ko razlika te verjetnosti med dvema iteracijama pade pod določen prag, algoritem presodi, da je razlika med ospredjem in ozadjem zadostna za dobro segmentacijo, se zaključi ter vrne rezultat zadnje iteracije. Slika 2.4 prikazuje trenutne rezultate v zaporednih iteracijah algoritma.

2.1.3 Morfološko filtriranje šuma

Ker rezultat segmentacijskih algoritmov lahko vsebuje šum, obstajajo metode, ki binarne slike preoblikujejo glede na izbran strukturni element, njihov osnovni namen pa je spremeniti obseg področja, ki ga pokriva binarna

maska. Osnovni operaciji sta tukaj erozija in dilatacija, iz njiju izpeljani pa še odpiranje in zapiranje. Binarne morfološke operacije temeljijo na uporabi enostavnega strukturnega elementa za prehod preko slike in aplikaciji enostavnih operacij (unija, presek) na obstoječo binarno masko. Takšne operacije lahko uporabimo za izbris majhnih struktur v sliki, ki so posledica šuma, za izboljšanje kontur objektov ali za glajenje robov maske.

2.2 Houghov transform

Houghov transform, opisan v članku [25], je bil najprej namenjen zaznavanju enostavnih oblik v slikah (linije, parametrične krivulje). Temelji na ideji, da lahko posamezne točke glasujejo za vse možne vrednosti parametrov, ki jih same podpirajo (npr. vse premice, ki lahko potekajo skozi točko), ti glasovi se shranjujejo v večdimenzionalni matriki, parametre iskane strukture pa določajo koordinate z največ glasovi.

2.2.1 Generaliziran Houghov transform

V članku [17] je opisana generalizacija Houghovega transforma, ki omogoča zaznavanje poljubnih oblik po podobnem principu kot navaden Houghov transform. Postopek poteka tako, da se objektu poljubne oblike določi referenčno točko, v odnosu do nje pa za vsako točko x, ki leži na konturi objekta, izračuna smer gradienta ϕ in razdaljo do referenčne točke r. Detekcija oblike v sliki nato poteka po naslednjem postopku: za vsako točko na konturi objekta se inkrementira (poveča) vrednost v akumulatorski tabeli na koordinatah, določenih z vrednostmi parametrov ϕ in r. Koordinate z največjim številom glasov tako določajo center iskanega objekta. Metoda v članku [17] za izhodišče uporablja obrise iskanih objektov, druge metode [18, 7] pa lahko temeljijo tudi na zaplatah, iz katerih se zgradi izhodiščni model predmeta.

2.3 Harrisova detekcija oglišč

Pri sledenju in detekciji objektov v sliki je mnogokrat koristno poiskati značilne točke (*feature points*). Eden enostavnejših načinov za to je uporaba Harrisove funkcije za zaznavo oglišč. Teorija za izračun vrednosti funkcije, ki opisuje verjetnost, da se na določeni točki v sliki nahaja oglišče, je opisana v [26]. Izhodiščna ideja je, da se da oglišče definirati kot točko, kjer se stikata dva robova in se njuni smeri spremenita, zato je v taki točki sprememba gradienta v poljubno smer velika. Območja z uniformno teksturo imajo majhne spremembe gradienta v vse smeri, robovi močno spremembo le v eni smeri, pravokotno na to smer pa je sprememba intenzivnosti gradienta zelo majhna. Spremembo gradienta ob premiku v poljubno smer iz poljubne točke v sliki opišemo z enačbo

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2. \tag{2.3.1}$$

Iz te enačbe je preko Taylorjeve razširitve izpeljana formula za izračun vrednosti funkcije

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}, \tag{2.3.2}$$

kjer je M, imenovan Harrisova matrika

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{2.3.3}$$

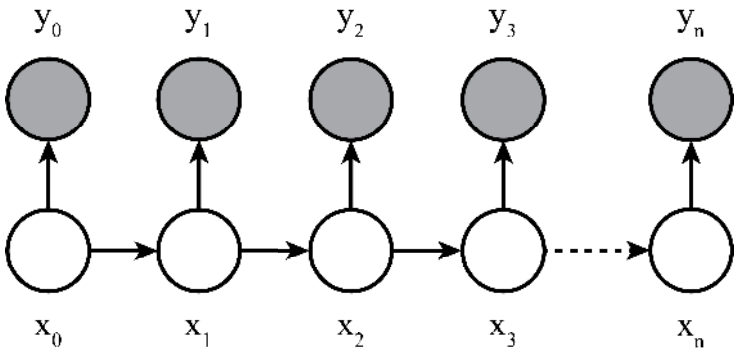
spremenljivka $w(x,y)$ označuje okno v sliki na koordinatah x in y, I_x in I_y pa sta parcialna odvoda vhodne slike I. Po formuli

$$R = det(M) - \alpha(trace(M))^2 \tag{2.3.4}$$

se tako iz gradientov v smereh x in y da izračunati vrednosti funkcije, ki je v uniformnih območjih majhna, na robovih negativna, v okolici oglišč pa velika – z upragovanjem (angl. thresholding) lahko nato določimo točke, kjer je vrednost dovolj velika, da lahko sklepamo na dejansko prisotnost oglišča.

2.4 Kalmanov filter

Kalmanov filter je algoritem, ki deluje na zaporedju podatkov, ki vsebujejo šum in skuša statistično optimalno oceniti dejansko stanje sistema. Deluje v dveh korakih, v koraku predikcije filter producira ocene spremenljivk, ki določajo stanje sistema (angl. *state variables*), skupaj z njihovimi negotovostmi. Po prejemu naslednje meritve (popačene s šumom) pa se te ocene posodobijo z uteženim povprečjem, kjer večjo težo dobijo parametri, ki jim bolj zaupamo. Zaradi svoje rekurzivne narave lahko ta deluje v realnem času, saj pri delovanju uporablja le trenutne meritve in stanje v prejšnjem koraku.



Slika 2.5: Prikriti Markovov model.

Kalmanov filter je poseben primer rekurzivnega Bayesovega filtra, kjer predpostavimo, da so vse porazdelitve Gaussove in da je opazovani sistem linearen. Lahko si ga predstavljamo kot analognega prikritemu Markovovemu modelu (angl. Hidden Markov model), prikazanem v Sliki 2.5, ker se v vsakem koraku ocenjuje vrednost skritih spremenljivk glede na opazovane vrednosti. Takšni filtri so sestavljeni iz štirih delov:

- stanje – lastnosti opazovanega sistema v nekem trenutku (npr. pozicija, hitrost, center objekta . . .)
- model opazovanja – preslika meritev v verjetnost, t.j. kakšna je verjetnost meritve y_k, če je sistem v stanju x_k, kar opisuje enačba $p(y_k|x_k)$

- dinamični model – predvidi trenutno stanje sistema iz ocene v prejšnjem koraku glede na nastavljene predpostavke
- inferenca – združi prejšnje tri dele tako, da posteriorno vrednost v trenutnem koraku zapiše kot funkcijo posteriorne vrednost iz prejšnjega koraka (in tako rekurzivno tudi upošteva vse prejšnje). To se da izraziti kot enačbo

$$p(x_k|y_{1:k}) \propto p(y_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1}, \tag{2.4.1}$$

  kjer $p(x_k|y_{1:k})$ predstavlja trenutno oceno stanja, $p(x_k|y_{1:k})$ model opazovanja, $p(x_k|x_{k-1})$ dinamični model, $p(x_{k-1}|y_{1:k-1})$ pa oceno stanja v prejšnjem časovnem koraku.

Delovanje dinamičnega modela opisuje enačba

$$\boldsymbol{x}_k = \boldsymbol{\Phi}\boldsymbol{x}_{k-1} + w_k, \tag{2.4.2}$$

kjer je $\boldsymbol{\Phi}$ sistemska matrika, $\boldsymbol{x}_{k-1}$ stanje v prejšnjem koraku, w_k pa predstavlja šum v sistemu in ima normalno porazdelitev $w_k \sim \mathcal{N}(0, \boldsymbol{Q})$. Gaussov šum v sistemu je definiran z matriko $\boldsymbol{Q}$, ki označuje njegovo kovarianco. Spremenljivka $\boldsymbol{x}_k$ tako opisuje posodabljanje stanja med zaporednimi koraki z upoštevanjem šuma. Drugi uporabljeni model pri izračunu posteriorne ocene je model opazovanja, ki ga prikazuje enačba

$$y_k = Hx_k + v_k, \tag{2.4.3}$$

kjer je $\boldsymbol{H}$ matrika opazovanja, v_k pa šum pri opazovanju z normalno porazdelitvijo $v_k \sim \mathcal{N}(0, \boldsymbol{R})$. Ta enačba preslika prostor stanja v prostor meritve in hkrati upošteva predviden šum v izmerjenih vrednostih. Kalmanov filter za inicializacijo potrebuje štiri matrike, torej sistemsko matriko $\boldsymbol{\Phi}$, matriko kovariance sistemskega šuma $\boldsymbol{Q}$, matriko opazovanja $\boldsymbol{H}$ in matriko kovariance šuma opazovanja $\boldsymbol{R}$. V koraku Kalman filtra se tako najprej izračuna predikcija stanja po enačbi

$$\tilde{x}_k = \boldsymbol{\Phi}\hat{x}_{k-1} + \boldsymbol{\Gamma}u_k, \tag{2.4.4}$$

kjer je $\mathbf{\Gamma} u_k$ zunanji vhod, če je ta na voljo. Hkrati se posodobi posteriorna matrika kovariance napake (angl. error covariance matrix) po enačbi

$$\tilde{\boldsymbol{P}} = \boldsymbol{\Phi} \boldsymbol{P}_{k-1} \boldsymbol{\Phi}^T + \boldsymbol{Q}. \tag{2.4.5}$$

Po pridobljenih meritvah se izračuna končno stanje $\hat{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k + \boldsymbol{K}(y_k - \boldsymbol{H}\tilde{\boldsymbol{x}}_k)$ ter spet posodobi matrika napake, z enačbo

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\tilde{\boldsymbol{P}}_k, \tag{2.4.6}$$

kjer je $\boldsymbol{K}$ t.i. Kalmanovo ojačanje (angl. *Kalman gain*),

$$\boldsymbol{K} = \tilde{\boldsymbol{P}}_k \boldsymbol{H}^T (\boldsymbol{H}\tilde{\boldsymbol{P}}_k \boldsymbol{H}^T + \boldsymbol{R})^{-1}. \tag{2.4.7}$$

Vrednost $\boldsymbol{K}$ določa stopnjo zaupanja meritvam, saj v enačbi matriki $\boldsymbol{P}$ in $\boldsymbol{R}$ določata razmerje med uporabo predvidenega stanja ($\tilde{\boldsymbol{x}}_k$) in izmerjenih vrednosti ($\boldsymbol{y}_k$). Kadar je magnituda matrike $\boldsymbol{R}$ majhna, to pomeni večje zaupanje meritvam, posledično se rezultat zanaša predvsem nanje, in obratno, kadar je vrednost $\boldsymbol{H}\tilde{\boldsymbol{P}}_k \boldsymbol{H}^T$ majhna v primerjavi z $\boldsymbol{R}$, filter večinoma ignorira meritve in se zanaša predvsem na predikcijo, izpeljano iz prejšnjega stanja ($\tilde{\boldsymbol{x}}_k$).

2.4.1 Model skoraj konstantne hitrosti

Pogosto uporabljen dinamični model pri Kalman filtru predpostavlja, da je hitrost spreminjanja opazovanega sistema skoraj konstantna, imenujemo ga model skoraj konstantne hitrosti (angl. *Near-constant velocity model*, kratica NCV). Spremembo stanja sistema med koraki lahko opišemo s stohastično diferencialno enačbo

$$\dot{x} = \boldsymbol{F}x + \boldsymbol{L}w, \tag{2.4.8}$$

kjer je $\boldsymbol{F}$ deterministična matrika prehoda, w pa normalno porazdeljen šum, definiran s kovarianco q_c. Matriko $\boldsymbol{\Phi}$ iz prejšnjega podpoglavja lahko tako dobimo z uporabo enačbe

$$\boldsymbol{\Phi}(\Delta t) = e^{\boldsymbol{F}\Delta t}. \tag{2.4.9}$$

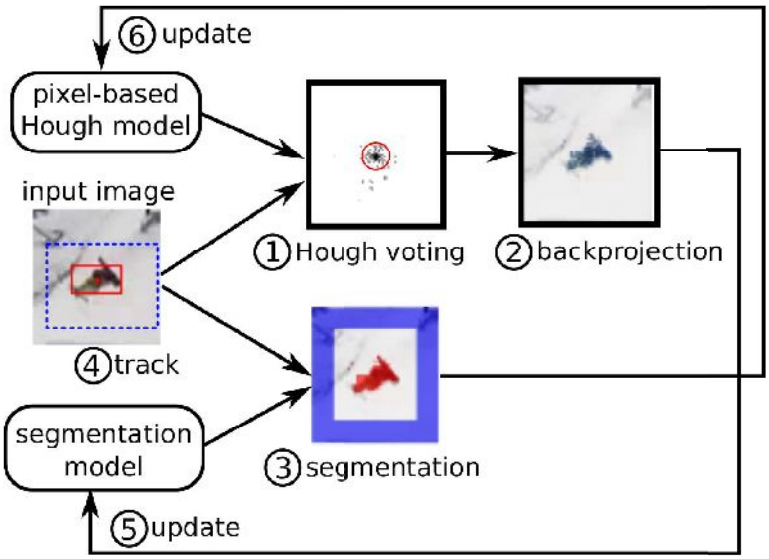
Ker moramo diferencialno enačbo diskretizirati, za pridobitev kovariančne matrike $\boldsymbol{Q}$, ki določa spremenljivko w_k, uporabimo enačbo

$$\boldsymbol{Q}_k = \int_0^{\Delta t} (\Phi(\xi)\boldsymbol{L}) q_c (\Phi(\xi)\boldsymbol{L})^T d\xi. \tag{2.4.10}$$

Izračunane matrike za konkretni primer so navedene v podpoglavju 4.2.

2.5 PixelTrack

V tem podpoglavju predstavimo algoritem PixelTrack [8] in opišemo kako uporablja segmentacijo in generaliziran Houghov transform ter kako sta ta dva dela algoritma povezana, da omogočata uspešno sledenje objektov. Diagram delovanja algoritma je prikazan na Sliki 2.6.



Slika 2.6: Diagram delovanja, povzeto po [8].

V algoritmu je objekt opisan z mejnim pravokotnikom, ki ima konstantno velikost, zato se da pozicijo objekta v vsakem koraku opisati s koordinatama

x in y znotraj slike. Izgled objekta je opisan z dvema modeloma, prvi je Houghov model, ki deluje na nivoju pikslov in za vsako barvo piksla določa odmike v smeri centra objekta (kot prikazuje Slika 2.7), drugi pa segmentacijski model, ki za vsakega izmed pisklov znotraj mejnega pravokotnika določa, ali pripada objektu ali ozadju. Algoritem kot vhod sprejme prvo sliko sekvence ali video datoteko, mejni pravokotnik in rezultat segmentacije v prejšnji sliki, če je na voljo. Iz območja znotraj mejnega pravokotnika se izračuna histogram objekta, znotraj preiskovanega območja (angl. *search window*), ki je dvakrat povečan mejni pravokotnik, pa histogram ozadja. V vsakem koraku algoritma se nato znotraj preiskovanega območja izvede generaliziran Houghov transform. Uporabljena različica transformacije je podrobneje opisana v podpoglavju 2.5.3. Iz rezultata Houghovega transforma se nato izračuna povratna projekcija (angl. *backprojection*), t.j. označimo piksle, ki so glasovali za točko z največ Houghovimi glasovi. Ta projekcija kasneje omogoča posodobitev segmentacijskega modela, kot je vidno na Sliki 2.6. Vzporedno se znotraj istega območja izvede segmentacija glede na trenutni model, t.j. vsakemu pikslu se določi pripadnost objektu ali ozadju, kot je podrobneje opisano v podpoglavju 2.5.2. Nova pozicija objekta se določi kot linearna kombinacija izhodnih vrednosti obeh delov algoritma po enačbi

$$X_t = \alpha x_s + (1 - \alpha) x_{max}, \tag{2.5.1}$$

kjer je x_s center mase segmentacije, x_{max} pa pozicija točke z največ glasovi pri GHT.

Faktor α določa, koliko zaupamo rezultatu segmentacije v primerjavi s Houghovim transformom in se dinamično določi v vsakem koraku glede na odstotek pikslov, ki so prešli iz ospredja v ozadje ali obratno. Če je ta vrednost velika, lahko sklepamo, da se je oblika objekta opazno spremenila in je zato Houghov transform manj zanesljiv.

Center mase segmentacije se izračuna po enačbi

$$x_s = \frac{1}{S} \sum_{x \in \Omega} p(c_x = 1|y) x, \tag{2.5.2}$$

kjer je S vsota vseh verjetnosti za ospredje $p(c_x = 1|y)$ v preiskovanem območju Ω.

2.5.1 Posodabljanje modelov

Za izognitev drsenju se model za Houghovo glasovanje in segmentacijski model križno posodabljata. To pomeni, da se povratna projekcija, pridobljena iz Houghovega glasovanja, uporabi kot izhodišče za izračun barvnih porazdelitev ospredja in ozadja in linearno posodobi segmentacijski model, kot prikazuje enačba

$$p(y_t|c_t = 1) = \delta p(y|b > 0.5) + (1 - \delta)p(y_{t-1}|c_{t-1} = 1), \tag{2.5.3}$$

kjer je $p(y|b > 0.5)$ barvna distribucija projiciranih pikslov v projekciji b, δ pa faktor posodobitve.

Houghov model se posodablja glede na rezultat segmentacije. Za vsakega izmed pikslov, ki z določeno verjetnostjo pripada ospredju ($p(c_x = 1|y) > 0.5$), se izračuna odmik od novega centra objekta, ta odmik pa se doda modelu oz. posodobi (če odmik za to barvo piksla že obstaja) glede na verjetnost $p(c_x = 1|y)$.

2.5.2 Segmentacija

Vzporedno s Houghovim modelom se trenira tudi globalni segmentacijski model. Teorija, ki jo za segmentacijo uporablja PixelTrack, izhaja iz pripevka [27]. Njegov namen je, da se prilagaja različnim oblikam objekta in skuša doseči čim boljšo diskriminacijo med ospredjem (objektom) in ozadjem, predvsem, kadar se oblika in izgled spreminjata drastično in nenadoma. Za ta namen se v algoritmu uporabi probabilistična mehka segmentacija. Naj bo $c_{t,x} \in \{0, 1\}$ razred piksla na lokaciji $\boldsymbol{x}$ ob času t: 0 za ozadje in 1 za ospredje, $y_{1:t,\boldsymbol{x}}$ pa naj bo opazovana barva piskla na $\boldsymbol{x}$ v času med 1 in t. Za vključitev segmentacije prejšnje slike v sekvenci uporabimo rekurzivno Bayesovo formulacijo, kjer se klasifikacija piksla v ospredje za korak t določi

z enačbo

$$p(c_t = 1|y_{1:t}) = Z^{-1} p(y_t|c_t = 1) \sum_{c'_{t-1}} p(c_t = 1|c'_{t-1}) p(c'_{t-1}|y_{1:t-1}), \qquad (2.5.4)$$

kjer je Z normalizacijska konstanta.

Distribucije $p(y_t|c_t)$ se modelirajo z HSV barvnimi histogrami, zapisanimi v barvnem prostoru HSV. Histogrami vsebujejo 12x12 polj za kanala H in S in ločenih 12 polj za kanal V. Histogram ospredja se inicializira iz dela slike, ki ga opisuje mejni pravokotnik okoli objekta v prvi sliki, histogram ozadja pa iz območja, ki obkroža objekt. Verjetnosti za prehod med ospredjem in ozadjem $p(c_t|c_{t-1})$ so avtorji določili eksperimentalno, in so opisane v podpoglavju 4.2.2.

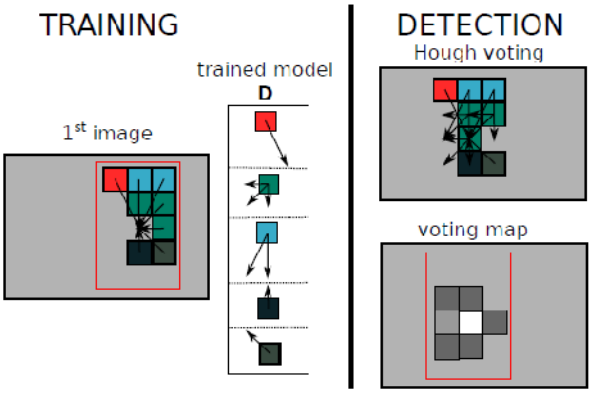
2.5.3 Generaliziran Houghov transform

Poleg segmentacije PixelTrack uporablja tudi generaliziran Houghov transform, ki temelji na principu, opisanem v članku [17], a za razliko od drugih metod na tem področju deluje na nivoju pikslov. To, povzeto po prispevku [8], ima več prednosti:

- Deskriptorji na nivoju pikslov so primernejši za zaznavanje zelo majhnih struktur v slikah.
- Prostor značilk (angl. feature space) je relativno majhen in ni odvisen od prostorske soseščine, kar poenostavi trening in posodabljanje modela.
- Trening in aplikacija detektorja je zelo hitra, ker se jo da implementirati z vpoglednimi tabelami.

To v praksi pomeni, da se pri gradnji izhodiščnega modela upoštevajo vsi piksli znotraj preiskovanega območja, vsak izmed njih je kvantiziran glede na svojo HSV vrednost in smer gradienta, kar je prikazano na Sliki 2.7. Model je tako opisan kot N–dimenzionalen histogram, ki ima za vsakega izmed

upoštevanih pikslov shranjen vektor odmika, ki kaže proti središču objekta (piksel neke barve se lahko pojavi večkrat in ima lahko več različnih smeri).



Slika 2.7: Uporaba GHT v Pixeltracku, povzeto po [8].

Detekcija objekta po tej metodi deluje na naslednji način: V preiskovanem območju nove slike se za vsak piksel v modelu preveri shranjene odmike (glede na njegovo barvo in gradient) ter poveča vrednost akumulacijske matrike na ustreznih koordinatah. Center iskanega objekta se nato določi z lokacijo točke z največ glasovi.

Poglavje 3

Razširjeni PixelTrack

To poglavje je namenjeno razširitvam, uporabljenim pri izboljšavi algoritma PixelTrack. Podpoglavje 3.2 opisuje uporabo Harrisove detekcije oglišč, v podpoglavju 2.1.2 je predstavljena uporaba segmentacijskega algoritma na osnovi Markovovih slučajnih polj, podpoglavje 3.3 pa vsebuje opis uporabe Kalmanovega filtra za pomoč pri sledenju. V podpoglavju 3.4 se nahaja predstavitev razširjenega algoritma in opis njegovega delovanja.

3.1 Uporaba segmentacije z MRF

V razširjenem algoritmu PixelTrack smo prejšnji segmentacijski model zamenjali s tistim, ki je opisan v podpoglavju 2.1.2. Novi algoritem smo povezali s segmentacijskim delom na isti način, kot je bil z njim povezan osnovni algoritem, kar nam je omogočilo tudi uporabo istih virov. Rezultat tega je, da novi segmentacijski algoritem za vključitev v osnovni algoritem ne potrebuje lastne funkcije za pripravo vhodnih podatkov, saj veliko tega dela izvede že PixelTrack sam. Funkcija, ki je v originalnem algoritmu izvajala segmentacijo, sedaj prevzame nalogo priprave vhoda za novi algoritem, torej računanje projekcij histogramov ospredja in ozadja. Zatem se izvede segmentacija z MRF, ki, tako kot stari algoritem, proizvede binarno masko.

3.2 Uporaba Harrisove detekcije

V Pixeltracku je Harrisova detekcija oglišč uporabljena na podlagi razmisleka, da so nekatere točke v sliki oz. v preiskovanem območju bolj informativne od drugih, t.j. nosijo več podatkov o vsebini slike kot druge. V predlaganih spremembah se tako v vsakem koraku sledenja izračuna vrednost Harrisove funkcije za celotno sliko, rezultati pa se pri sledenju uporabijo kot utež za glasove v Houghovi glasovalni shemi. Zaradi velikega intervala izhodnih vrednosti Harrisove funkcije je rezultat stisnjen preko eksponentne funkcije

$$y = (1 - exp(\frac{-x}{\lambda})), \tag{3.2.1}$$

kar preslika vrednosti na interval $(0, 1)$. Tako se v vsakem koraku sledenja po zaključku Houghovega glasovanja glasovalna matrika (angl. voting map) po elementih pomnoži z rezultatom Harrisove funkcije. Posledica tega je, da imajo točke z večjo vrednostjo Harrisove funkcije (idealno so te točke na konturah sledenega objekta) večjo težo pri glasovanju za center sledenega objekta, kar pripelje do boljše ocene njegove trenutne lokacije. Sama povezava Harrisove funkcije z osnovnim algoritmom je izpeljana zelo enostavno. PixelTrack v vsakem koraku izračuna odvode po x in y, ki jih potrebuje za Houghovo glasovanje, hkrati pa se direktno uporabijo za izračun vrednosti Harrisove funkcije. Te vrednosti se nato kot dodaten parameter podajo funkciji, ki skrbi za Houghovo glasovanje.

3.3 Uporaba Kalmanovega filtra

Kalmanov filter v razširjenem PixelTracku ne deluje kot interni del algoritma, marveč se aktivira po zaključku vseh drugih izračunov in iz lokacije sledenega objekta, ki jo je določil algoritem, izračuna popravljeno lokacijo glede na svoje parametre. Stanje sistema je v tem primeru vektor trenutne lokacije sledenega objekta in njegove hitrosti v obliki $\begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^T$. Meritev v vsakem koraku pa je v obliki $\begin{bmatrix} x_k & y_k \end{bmatrix}^T$, torej x in y koordinati, ki ustre-

zata rezultatu, ki ga proizvede PixelTrack. Filter, ki deluje v PixelTracku, uporablja dinamični model NCV, kot je opisan v podpoglavju 2.4.1.

Lokacija, ki jo predvidi Kalmanov filter, je tako končni rezultat algoritma, služi pa predvsem temu, da ob morebitnem zdrsu z sledenega objekta upošteva njegovo predhodno premikanje, temu primerno popravi rezultat in skuša zadržati algoritem na pravilni lokaciji.

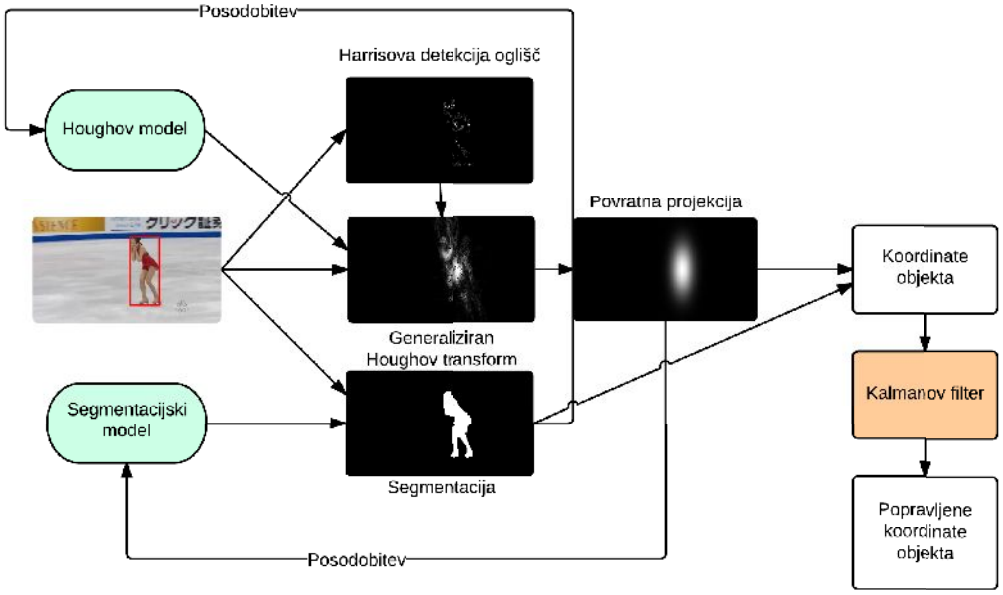
3.4 Opis razširjenega algoritma

Razširjeni algoritem PixelTrack vsebuje vse tri opisane spremembe, od katerih Harrisova detekcija in segmentacija z MRF delujeta znotraj algoritma, Kalmanov filter pa vpliva direktno na izhodne vrednosti algoritma v vsakem koraku. Notranje spremembe so izvedene na enostaven način in se skušajo v ogrodje osnovnega algoritma vključiti s čim manj spremembami.

Algoritem kot vhod sprejme video sekvenco in mejni pravokotnik, ki opisuje izhodiščno pozicijo sledenega objekta. Pred začetkom izvajanja se ustvarijo vpogledne tabele, ki omogočajo hitro izvajanje Houghovega transforma. Zatem se izračunata histograma ospredja in ozadja, ki služita kot podlaga za segmentacijski algoritem. Ustvari se tudi Houghov model objekta, ki je kasneje uporabljen za izvedbo Houghovega glasovanja. Inicializira se tudi Kalmanov filter. Nato se začne iteracija preko zaporednih slik v vhodni sekvenci, kjer se za vsako sliko izvedejo naslednji koraki: najprej se izračunata odvoda po x in y. Iz njiju se pridobijo vrednosti Harrisove funkcije. Nato se izvedeta osrednja postopka algoritma, torej Houghovo glasovanje in segmentacija MRF. Za namen posodabljanja obeh vizualnih modelov se nato izračuna povratna projekcija. Oba modela se nato križno posodobita, z linearno kombinacijo pa se določi tudi nova pozicija sledenega objekta. Kalmanov filter nato glede na svoje trenutno stanje popravi izhodne koordinate, ki so tudi končni rezultat vsakega koraka.

Delovanje razširjenega algoritma je povzeto v Algoritmu 1, Slika 3.1 pa predstavlja diagram poteka razširjenega algoritma, ki jasneje prikaže glavne

točke algoritma in povezave med njimi.



Slika 3.1: Diagram poteka delovanja razširjenega algoritma PixelTrack.

Algoritem 1 : Razširjeni algoritem PixelTrack.

Vhod:

Sekvenca slik, ki sestavljajo video in mejni pravokotnik za inicializacijo.

Izhod:

Seznam lokacij sledenega objekta v vsaki sliki sekvence (lokacija je opisana s pravokotnikom).

Postopek:

1: Ustvari vpogledne tabele za Houghovo glasovanje.
2: Izračunaj barvne histograme ospredja in ozadja, kot je opisano v podpoglavju 2.5.2.
3: Nauči se Houghov model objekta, postopek je opisan v podpoglavju 2.5.3.
4: Inicializiraj Kalmanov filter, ki uporablja dinamični model NCV, opisan v podpoglavju 2.4.1 in parametre, navedene v podpoglavju 4.2.
5: **Dokler** ostajajo slike v sekvenci **izvedi:**
6: Preberi naslednjo sliko.
7: Izračunaj odvode po x in y ter Harrisovo funkcijo po enačbi (2.3.3).
8: Izvedi Houghovo glasovanje.
9: Izvedi segmentacijo po postopku, opisanem v podpoglavju 2.1.2, z uporabo enačb (2.1.3).
10: Z rezultati Houghovega glasovanja izvedi povratno projekcijo.
11: Določi novo pozicijo objekta po enačbi (2.5.1).
12: Z uporabo Kalmanovega filtra popravi rezultat algoritma.
13: Posodobi modela, kot je opisano v podpoglavju 2.5.1.
14: **Konec zanke**

Poglavje 4

Eksperimenti

V tem poglavju opišemo izvedene eksperimente. Podpoglavje 4.1 predstavi testno zbirko in ogrodje izziva VOT2013, v podpoglavju 4.2 so navedeni uporabljeni parametri za posamezne dele algoritma, podpoglavje 4.3 pa predstavi izvedene eksperimente. Sledi podpoglavje 4.4, ki vsebuje evalvacijo osnovnega algoritma, opis sprememb pri dodajanju izboljšav ter grafe, ki prikazujejo končni rezultat razširjenega algoritma in primerjavo z drugimi sledilniki.

4.1 Opis testne zbirke

Najprej smo pri testiranju algoritma uporabili sekvence, citirane v [8], ki so povzete po prispevku [28]. Zaradi nujne pristranskosti takšnih sekvenc pa smo za neodvisnejše in intenzivnejše testiranje izbrali bazo, uporabljeno pri izzivu VOT2013, ki je predstavljena v prispevku [29]. V tej bazi se nahaja 16 sekvenc, izbranih iz množice primerov, ki jih raziskovalci pogosto uporabljajo za testiranje svojih sledilnih algoritmov. Ta podmnožica je bila izbrana tako, da se v njenih sekvencah pojavljajo različne lastnosti iz resničnega življenja, kot so zakrivanje, spremembe v osvetlitvi, premikanje kamere ipd., ki otežujejo delo sledilnega algoritma in pripomorejo k čim bolj intenzivnemu tesitranju natančnosti in robustnosti algoritmov. Poleg tega VOT2013 predlaga tudi metodologijo za primerjavo sledilnikov na podlagi

njihovih rezultatov (števila izgub objekta in povprečne natančnosti sledenja) in omogoča generiranje grafov, ki na enem mestu prikazujejo rezultate večih sledilnikov in tako poenostavijo primerjave med njimi. Delovanje okolja VOT2013 je podrobno opisano v članku [30].

4.2 Uporabljeni parametri v sledilniku

4.2.1 Kalmanov filter

Uporaba formul iz poglavja 2.4.1 nam za stanje oblike $\begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^T$ proizvede matriki

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$\boldsymbol{Q} = q \begin{bmatrix} \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 0 & 0 \\ 0 & 0 & \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ 0 & 0 & \frac{\Delta t^2}{2} & \Delta t \end{bmatrix}, \tag{4.2.1}$$

ki ju lahko direktno uporabimo v Kalmanovem filtru. Izraz Δt tukaj označuje časovno razliko med enim in drugim korakom in je za uporabo pri sledenju enak 1. Matriko opazovanja $\boldsymbol{H}$ določimo kot identitetno matriko velikosti 4×2, matrika $\boldsymbol{R}$ pa je enaka $\rho \boldsymbol{I}$, kjer je $\boldsymbol{I}$ identitetna matrika velikosti 2×2. Parametra q in ρ določimo na podlagi eksperimentov ali predhodnega znanja o natančnosti meritev. Za eksperimente sem določil dva seta, od katerih daje eden prednost predikciji, drugi pa meritvam. Prvi set ima vrednost $q = 1\text{e}3$, $\rho = 1$ ter bolj upošteva predvideno stanje, drugi pa $q = 1\text{e}2$, $\rho = 1\text{e}{-6}$ in se zanaša predvsem na meritve, predikcija pa mu je le za oporo.

4.2.2 Parametri segmentacije

V enačbi (2.5.4) so omenjene verjetnosti, da med enim in drugim korakom piksel preide iz ospredja v ozadje ali obratno. Te vrednosti so določene eksperimentalno:

$$p(c_t = 0|c_{t-1}) = 0.6 \qquad p(c_t = 1|c_{t-1}) = 0.4. \tag{4.2.2}$$

4.2.3 Harrisova detekcija oglišč

Pri Harrisovi detekciji oglišč parameter α v enačbi (2.3.4) predstavlja nastavljivo konstanto, ki vpliva na občutljivost filtra. Vrednosti tega parametra so določene eksperimentalno in se nahajajo med 0.04 in 0.06.

4.2.4 Segmentacija z MRF

Uporabljen algoritem za segmentacijo z uporabo MRF zahteva zaključni pogoj, ki ga predstavlja razlika med logaritemskimi verjetnostmi rezultatov zaporednih iteracij. Ta meja je v algoritmu postavljena na 10^-2, za kompromis med kvalitetno segmentacijo in izvajanjem nepotrebnih iteracij, ko so spremembe zanemarljive.

4.3 Eksperimenti

VOT 2013 vsebuje tri eksperimente:

- *baseline*: vsebuje barvne sekvence z ustreznimi regijami za inicializacijo
- *region-noise*: iste sekvence kot *baseline*, regije za inicializacijo pa so 'pokvarjene' z Gaussovim šumom (10%), za testiranje robustnosti
- *grayscale*: črnobele sekvence

Okolje VOT2013 vsebuje metode za generiranje primerjav med rezultati različnih sledilnikov. Grafi (A-R plots) so v obliki dvodimenzionalnega grafa,

ki ima na eni osi natančnost, na drugi pa zanesljivost sledilnika. Ti dve vrednosti sta pridobljeni tekom izvajanja evalvacije in sta v članku [30] definirani kot:

- natančnost (angl. accuracy): povprečno prekrivanje dejanskega območja sledenega objekta in območja rezultata, ki ga proizvede sledilnik skozi sekvenco.

- zanesljivost (angl. reliability): verjetnost, da bo algoritem uspešno sledil objektu do S slik po zadnji izgubi. Izguba objekta je tukaj definirana kot trenutek, ko je prekrivanje med dejanskim območjem objekta in območjem, ki ga proizvede sledilnik, enako 0.

Grafe je moč interpretirati na naslednji način: višje kot je točka, ki predstavlja algoritem, na vertikalni osi, bolj je le-ta natančen, in bolj desno kot je, manjkrat izgubi sledeni objekt. Idealni sledilnik bi se tako nahajal v desnem zgornjem kotu. VOT2013 za vse algoritme generira A-R grafe za vsako sekvenco in za vsak eksperiment, kar naredi analizo rezultatov in vplivov različnih eksperimentov enostavno.

Za generiranje A-R grafov sem pripravil naslednje različice algoritma PixelTrack (na koncu so imena, uporabljena v grafih):

- brez sprememb – *pt-base*

- Harrisova detekcija oglišč – *pt-harris-6*

- Harris ($\alpha = 0.04$) – *pt-harris-4*

- Harris in Kalmanov filter s šibkimi parametri – *pt-harris-kalman-weak*

- Harris in Kalmanov filter z močnimi parametri – *pt-kalman-strong*

- Harris in nova segmentacija (MRF) – *pt-new-segm*

- Harris in nova segmentacija (MRF) z močnim odpiranjem – *pt-new-segm-opening*

Harrisov filter ima vedno vrednost parametra $\alpha = 0.06$, razen če je napisano drugače. Kalmanov filter uporablja parametre, opisane v podpoglavju 4.2. V zadnji različici se na rezultat novega segmentacijskega algoritma aplicira močna operacija odpiranja, z namenom ohranjanja le večjih struktur v segmentaciji (žrtvovanje kvalitete segmentacije za večjo robustnost).

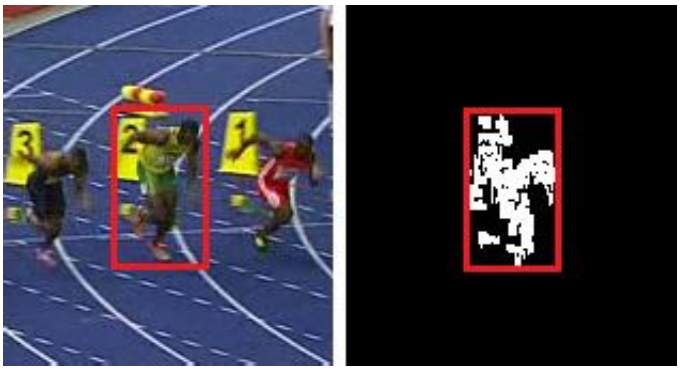
4.4 Diskusija rezultatov

Ker smo algoritmu vsako spremembo dodali posebej in ker so med sabo neodvisne, bomo njihov vpliv na delovanje algoritma opisali po delih in zaključili z evalvacijo razlik v rezultatih, ki so jih povzročile te spremembe.

4.4.1 Prvi vtisi

Na sekvencah, povzetih po Babenko *et al.* [28], PixelTrack seveda deluje dobro, čeprav so med njimi primeri, ki veljajo za težavne (*skiing, diving*). Opazno pa je, da je algoritem precej občutljiv na inicializacijo in pri težjih primerih zahteva natančno določeno izhodiščno regijo, da uspešno sledi objektu (to je predvsem opazno v sekvenci *Tiger 2*). Drug problem, ki hitro postane očiten, je pogosto nezadostna segmentacija, ki v slabših pogojih mnogokrat ne zadostuje za uspešno sledenje. V članku je omenjeno, da je zaradi zaželene čim večje hitrosti uporabljen enostaven segmentacijski algoritem, saj ni cilj dobiti popolne segmentacije, marveč le čim boljšo oceno lokacije objekta. Slaba stran tega pa je, da taka segmentacija odpove v nekaterih zahtevnejših primerih (med najbolj očitnimi sta *bolt* in *jump*), kjer se objekt in ozadje po videzu ne razlikujeta dovolj močno. V takih primerih se k segmentaciji dodajo deli ozadja (vidno na Sliki 4.1), kar skoraj vedno pripelje do popačenega središča mase in posledično do zdrsa z objekta.

V sekvencah, kot sta *singer* in *juice*, se pozna tudi pomanjkljivost PixelTracka, saj velikosti mejnega pravokotnika med izvajanjem ne prilagaja velikosti sledenega objekta, zato lahko prihaja do težav pri sledenju. Možno je, da se velikost objekta spremeni, histogram pa se še vedno računa na podlagi



Slika 4.1: Detajl iz sekvence *bolt*, ki prikazuje slabo segmentacijo.

izhodiščne velikosti preiskovanega območja, kar pripelje do slabe razločitve med ospredjem in ozadjem in posledično do slabe segmentacije.

Sekvence izziva VOT2013 so namenoma zahtevne, saj vsebujejo veliko različnih oteževalnih okoliščin za sledilne algoritme in hitro izpostavijo pomanjkljivosti algoritma. Po dobro izbranih sekvencah iz članka so tiste v izzivu bistveno manj popustljivo testirale algoritem. Po opažanjih, da v VOT PixelTrack razmeroma pogosto izgubi objekt, sem VOT sekvence, kjer se pojavijo težave, v grobem kategoriziral v tri razrede.

- Sekvence, kjer je do izgube prišlo zaradi nezadovoljive/napačne segmentacije.

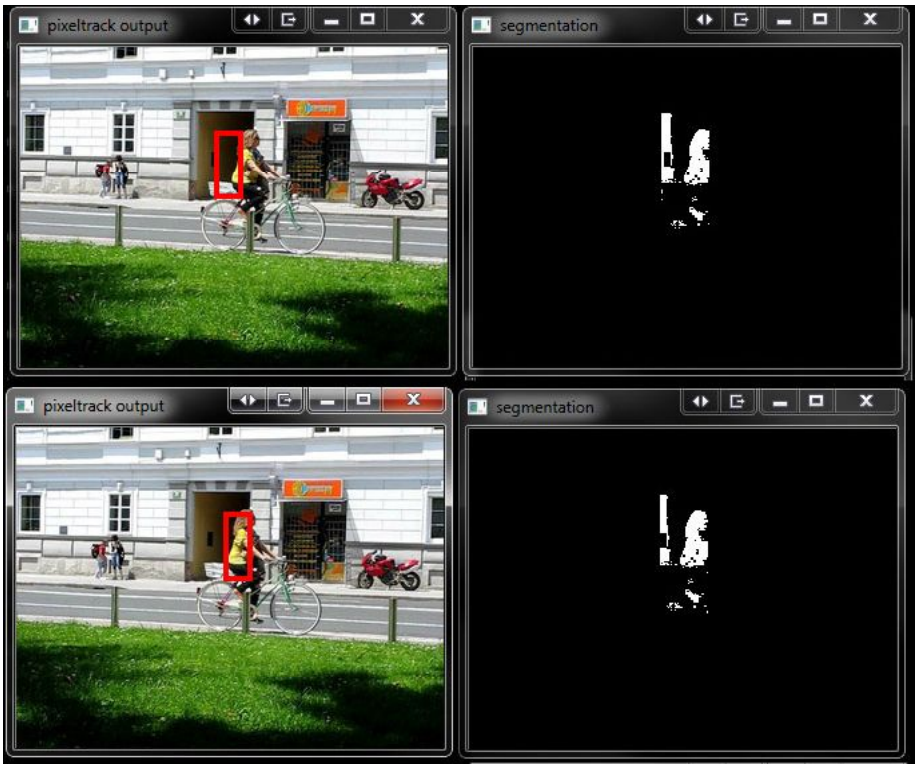
- Sekvence, kjer je prišlo do zdrsa z objekta kljub kvalitetni segmentaciji in sekvence, kjer je bila segmentacija večinoma dobra, na nekaterih mestih pa je njena kvaliteta malo padla, kar je povzročilo zdrs.

- Sekvence brez težav (uspešno sledenje od začetka do konca).

Tako sem se najprej osredotočil na drugi razred, ki je izgledal najbolj perspektiven za morebitne spremembe.

4.4.2 Harrisova detekcija oglišč

Prva sekvenca, kjer je bila segmentacija večinoma dovolj dobra, a je v zelo kratkem času degradirala in povzročila zdrs, je bila sekvenca *bicycle*, kjer je algoritem izgubil objekt v trenutku, ko je ženska na kolesu zapeljala pred črno ozadje (Slika 4.2).



Slika 4.2: Zgoraj: zdrs zaradi pokvarjene segmentacije, spodaj: ista slika brez zdrsa, zaradi delovanja Harrisove detekcije oglišč.

Zaradi majhnosti objekta v sliki in majhnih barvnih razlik med objektom in ozadjem se je črno ozadje dodalo v segmentacijo, kar je center mase fiksiralo na podhod in povzročilo izgubo objekta. Izgledalo je, da tu algoritem potrebuje nekakšno 'sidro', ki bi preprečilo, da bi slaba segmentacija preveč vplivala na Houghovo shemo. Izkaže se, da je zelo enostavna rešitev Harrisov detektor oglišč, saj so le-ta zelo informativna in tako omogočijo

bolj natančno lociranje objekta. Houghovi glasovi se tako v vsakem koraku pomnožijo z rezultatom Harrisove funkcije in tako (vsaj v sekvenci *bicycle*) preprečimo zdrs. Druge sekvence se na spremembo niso tako dobro odzvale, sprememba ni pokazala vpliva, ali pa je (pri sekvencah z zdrsom) algoritem še bolj zmedla.

4.4.3 Kalmanov filter

V sekvenci *torus* se je pojavil nekoliko drugačen problem, saj je pomanjkljiva segmentacija povzročila, da je algoritem prepočasi sledil objektu. Center mase segmentacije se je odklonil, zaradi česar je v nekaj slikah objekt popolnoma izgubil, kot je vidno na Sliki 4.3.

Slika 4.3: Primer zdrsa z objekta zaradi težav pri segmentaciji (prikazane so slike 57, 65 in 76 iz sekvence *torus*).

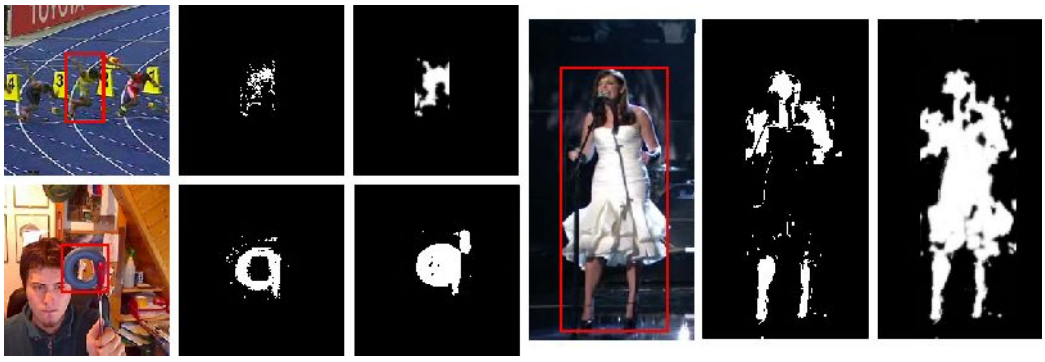
Zato je bila naslednja izboljšava dodatek Kalmanovega filtra. Izhodišče te odločitve je bil razmislek, da bi algoritmu lahko koristil dinamični model,

ki bi rezultat povlekel v pravo smer in tako preprečil zdrs pri problematičnih sekvencah. Izkazalo se je, da se z natančno izbiro parametrov da omogočiti algoritmu uspešno sledenje v tej sekvenci. Vpliv se kaže tudi v drugih sekvencah, a rezultati se zelo razlikujejo. Zaradi težavnosti natančnega preverjanja vpliva Kalmanovega filtra na vse sekvence sem pri testiranju uporabil dva različna seta parametrov (opisani so v podpoglavju 4.2) z različno intenzivnim vplivom.

4.4.4 Segmentacija MRF

Ko algoritem poženemo na nekaj težavnejših sekvencah, se hitro opazi, da je najbolj ključni del algoritma segmentacija, saj imajo sekvence v tretjem razredu, opisanem v podpoglavju 4.4.1, težave zaradi premajhnih razlik med objektom in ozadjem. To povzroči dodajanje delov ozadja k rezultatu segmentacije, zaradi česar je algoritem izgubil sledeni objekt. Originalno segmentacijo smo zato zamenjali z drugim algoritmom, ki je opisan v podpoglavju 2.1.2.

Segmentacijski algoritem lahko deluje tudi povsem samostojno in kot vhod sprejme povratno projekcijo histogramov ospredja in ozadja ter rezultat prejšnje segmentacije (za vse korake razen prvega). Priprava vhoda ni del same optimizacije, marveč se izvede posebej, kar omogoča uporabo kode tudi za kakšen drug namen. Lahko se, naprimer, uporabijo različni načini izračuna histogramov (v PixelTracku se uporabi HSV histogram iz originalnega algoritma), različne velikosti vhodnih projekcij ipd. Ker se v PixelTracku v vsakem koraku posodabljata histograma ospredja in ozadja (ospredje je tukaj enako mejnemu pravokotniku, ozadje pa preiskovanemu območju), je med izvajanjem algoritma enostavno pridobiti projekcijo obeh histogramov. Segmentacija se nato izračuna znotraj preiskovanega območja, saj lahko objekt pričakujemo zgolj tam, namen segmentacije pa je predvsem odstraniti piksle, ki pripadajo ozadju (pri nepravokotnih objektih). Ta omejitev tudi zmanjša zamik v vsakem koraku, saj se izognemo obdelavi večinskega dela slike, ki zagotovo ne vsebuje sledenega objekta. Konkretno,



Slika 4.4: Razlika med segmentacijama v sekvencah *bolt* (levo zgoraj), *torus* (levo spodaj) in *singer* (desno). Segmentacija v originalnem algoritmu je prikazana na levi strani, segmentacija z MRF pa na desni.

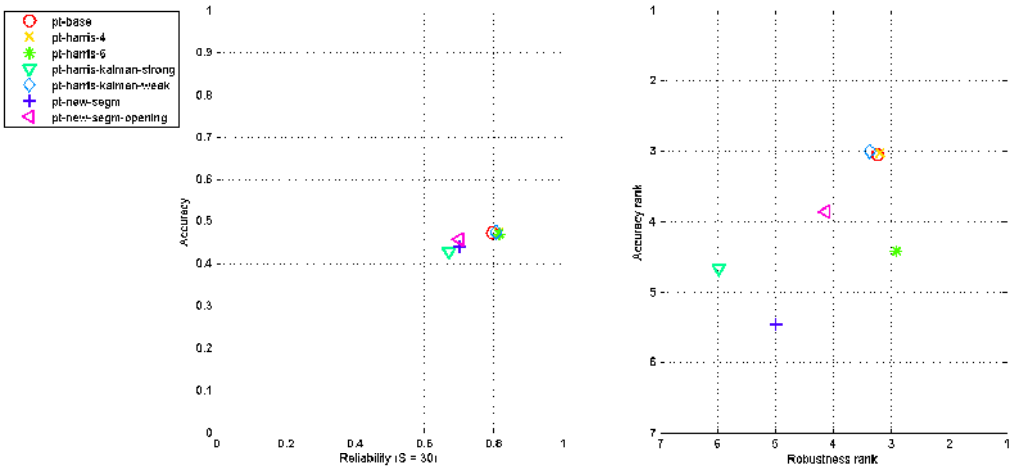
pri računanju preko cele slike nastane pri segmentaciji zamik cca 500 ms, omejitev izključno na preiskovano območje pa to zmanjša na 10-30 ms.

Slika 4.4 vsebuje rezultate obeh segmentacijskih algoritmov nad isto sliko in prikazuje podobnosti in razlike med njima. Originalna segmentacija ima bolj grobe robove in pogosto prektiva le večji del, ne pa celega območja objekta. V primerjavi pa nova segmentacija proizvede bolj gladke in polne rezultate (zvezen obris, brez negativnih pikslov znotraj objekta). Največja razlika se pojavi pri inicializaciji sekvence *torus*, kjer originalni algoritem uspešno izloči ozadje iz luknje v središču objekta, novemu pa to ne uspe.

Kljub očitnim razlikam v delovanju obeh algoritmov, tako v principu kot v rezultatih, razlika med izvajanjem ni tako intenzivna, saj originalna segmentacija v razmeroma enostavnih sekvencah deluje dovolj dobro, v zahtevnejših pa ima predlagan algoritem podobne težave zaradi samih lastnosti sekvenc. Oba algoritma namreč delujeta na podlagi barvnih atributov pikslov, kar v sekvencah kot je npr. *bolt* zaradi pojavljanja podobnih barv v ospredju in ozadju in ponekod tudi sprememb v velikosti objekta pripelje do neobhodnih težav.

4.4.5 Diskusija rezultatov

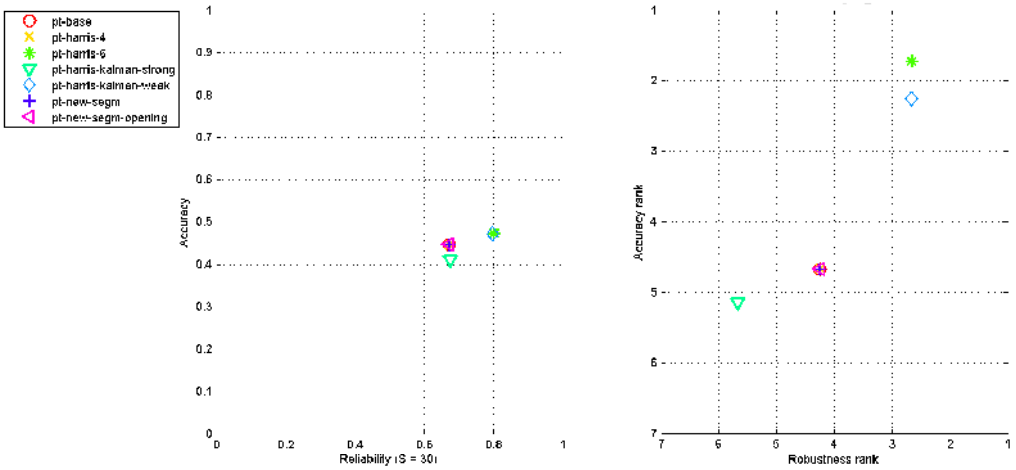
Za čim boljšo primerjavo različnih verzij algoritma sem pripravil več setov rezultatov in iz njih generiral A-R grafe, ki prikazujejo razlike, ki jih je vsaka sprememba povzročila.



Slika 4.5: A-R grafa eksperimenta *baseline*.

Slike 4.5, 4.6 in 4.7 prikazujejo uspešnost vsake od različic algoritma v vsakem od treh eksperimentov okolja VOT2013. Na levi strani so prikazani dejanski rezultati, na desni pa rangiranje sledilnikov. Rezultati za *baseline* kažejo razmeroma majhne izboljšave glede na osnovni algoritem, a ob pogledu na druga dva eksperimenta vidimo, da je prvi vtis zavajajoč. Gotovo drži, da preveliko zanašanje na ocene Kalman filtra prinaša slabe rezultate ter da je bil originalni segmentacijski algoritem zelo premišljeno izbran, saj segmentacija, osnovana na MRF, ne da boljših rezultatov, kljub temu, da je sama po sebi kvalitetnejša.

Vpliv predlaganih sprememb se pokaže v eksperimentu *region_noise*, kjer izrazito izstopata različici algoritma, ki uporabljata Harrisov detektor in Kalmanov filter, torej *pt-harris-6* in *pt-harris-kalman-weak*. Kot že rečeno, ima PixelTrack pri zahtevnejših sekvencah težave, če inicializacija ni dovolj na-

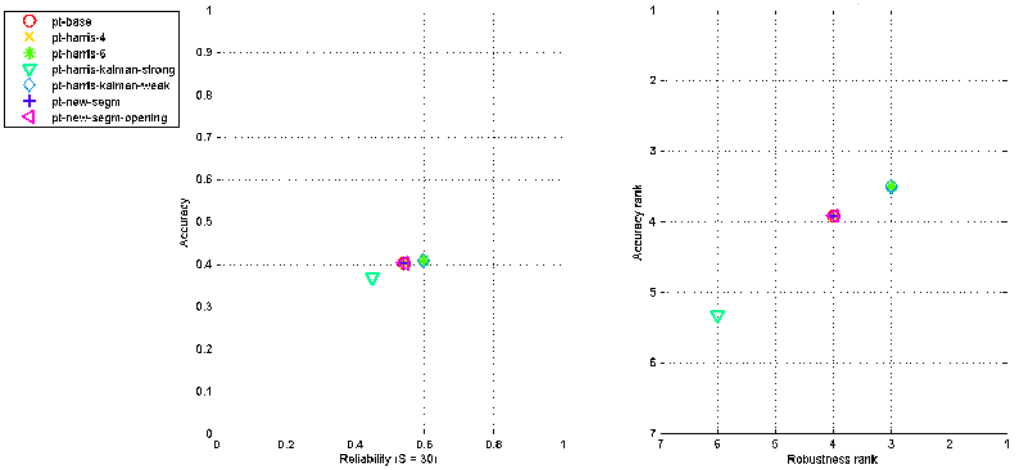


Slika 4.6: A-R grafa eksperimenta *region_noise*.

tančna, zato je izrazito izboljšan rezultat v tem eksperimentu toliko bolj presenetljiv. Zanimivo je tudi dejstvo, da so različice algoritma v eksperimentu *baseline* razporejene po večjem območju grafa in se (razen *pt-base*, *pt-harris-4* in *pt-harris-kalman-weak*) ne grupirajo, v eksperimentih *grayscale* in *region_noise* pa opazimo tri strogo ločene skupine:

- različici *pt-harris-6* in *pt-harris-kalman-weak* izstopata po kvaliteti rezultatov
- *pt-harris-kalman-strong* izstopa zaradi izrazito slabih rezultatov
- preostale različice so združene v eno gručo

Tako lahko opazimo, kako močan vpliv ima manj kvalitetna inicializacija in katere spremembe jo najbolje prenesejo. Podobno je z eksperimentom *grayscale*, kjer se dobro vidi, da se pri njem najbolje izkažejo različice, ki se ne zanašajo na barvne informacije. Harrisov detektor deluje na podlagi odvodov slike, medtem ko je Kalmanov filter že tako čisto neodvisen od informacij v sliki.

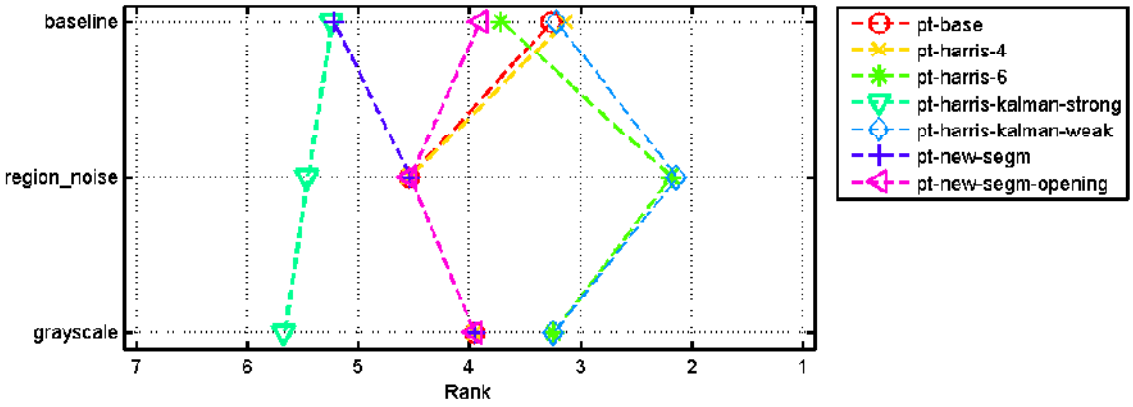


Slika 4.7: A-R grafa eksperimenta *grayscale*.

Okolje VOT2013 sledilnike tudi rangira glede na njihov rezultat, kar prikazuje Slika 4.13. Grupiranje in kvaliteta rezultatov se tukaj še jasneje vidita in potrjujeta, da sta se različici s Harrisovo detekcijo in zmernim Kalmanovim filtrom odrezali najbolje. V eksperimentu *baseline* ne toliko, sta pa nedvomni zmagovalki obeh drugih eksperimentov. Slika 4.10 in Slika 4.9 prikazujeta še surove rezultate, ki primerjajo še konkretne vrednosti rezultatov različic algoritma PixelTrack.

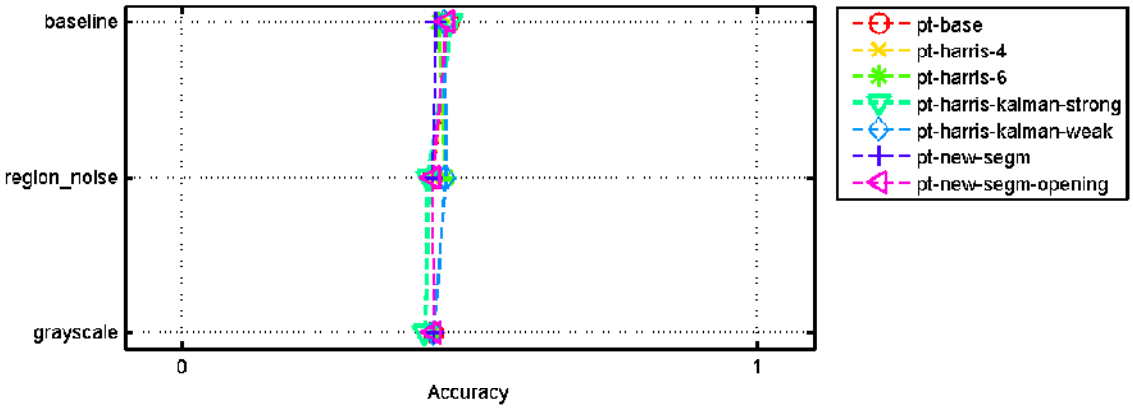
Barvni deskriptorji na nivoju pikslov imajo dobre lastnosti, v mnogo primerih pa zanašanje na barvo povzroča težave, ki se jim ne da izogniti z boljšim algoritmom na podobni osnovi (segmentacija MRF). Lahko pa jih zaobidemo z drugimi metodami, ki so neodvisne od barvnih informacij (značilne točke, konture ipd.). Če bi želeli uspešno slediti objektom v vseh eksperimentalnih sekvencah, bi za to morali v temelju spremeniti delovanje algoritma ali pa ga izboljšati na opisani način, torej z dodajanjem relativno neodvisnih postopkov, ki naslavljajo probleme, ki jih ima algoritem Pixeltrack zaradi svoje zasnove.

Sledijo še A-R grafi vseh sledilnikov iz izziva VOT2013, z vključenima originalnim algoritmom PixelTrack in različico *pt-harris-kalman-weak* (v grafih

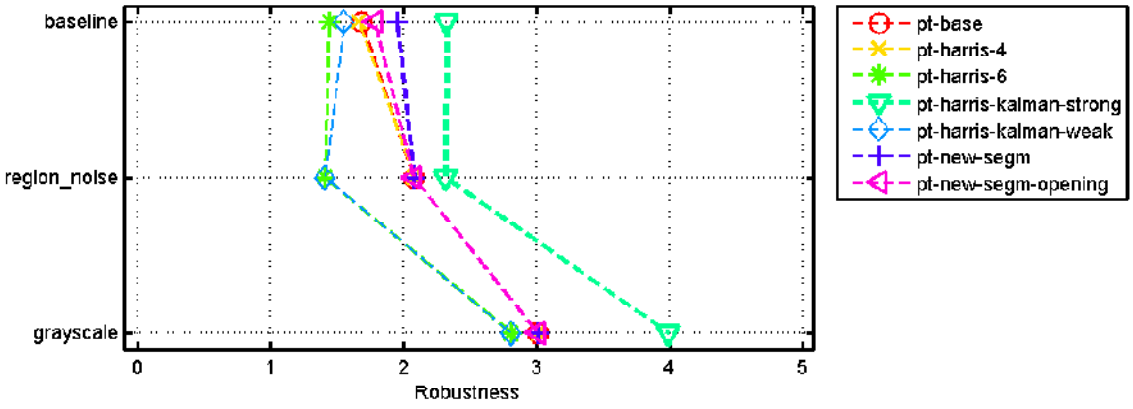


Slika 4.8: Rangiranje različic preko vseh eksperimentov.

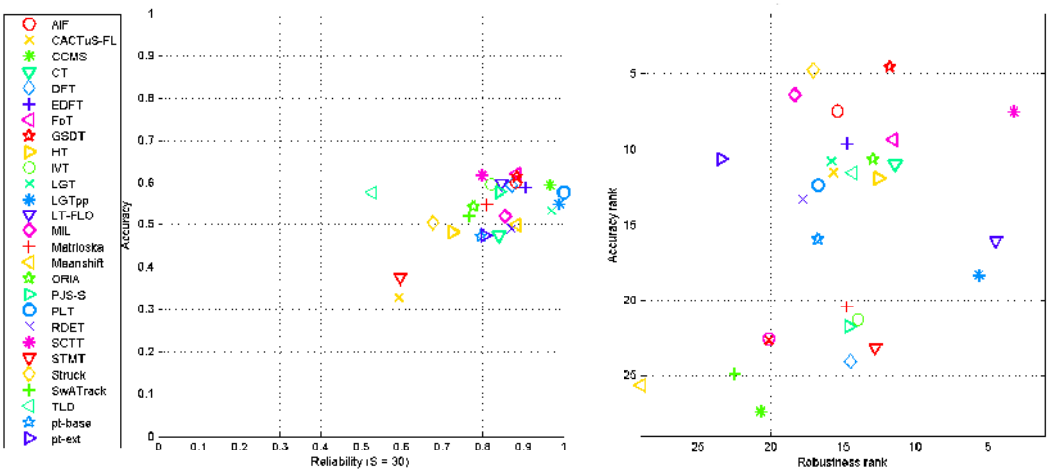
z vsemi sledilniki je poimenovana *pt-ext*), ki se v eksperimentih izkazala za najboljšo izmed preizkušenih. Razširjeni algoritem tu očitno prekaša originalnega, predvsem v eksperimentih *region_noise* in *grayscale*, kjer izboljša tako zanesljivost kot tudi natančnost.



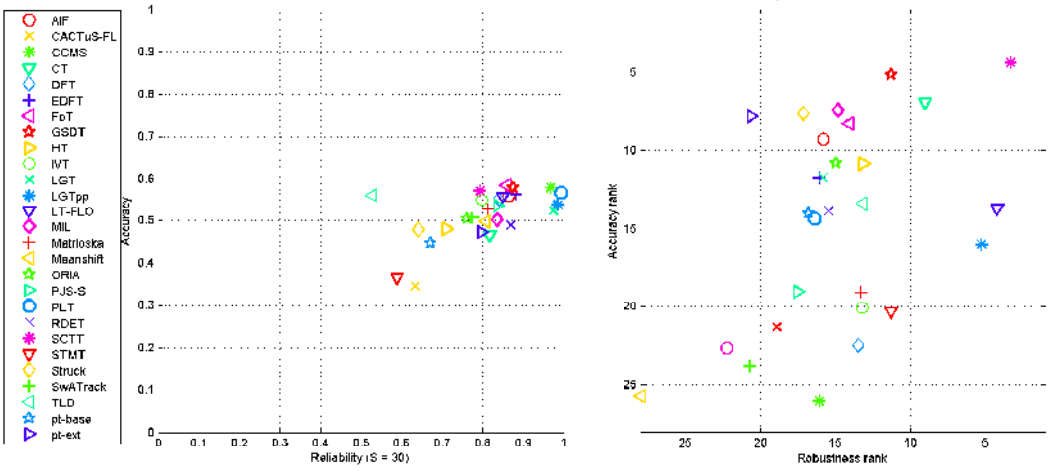
Slika 4.9: Natančnost različic algoritma PixelTrack.



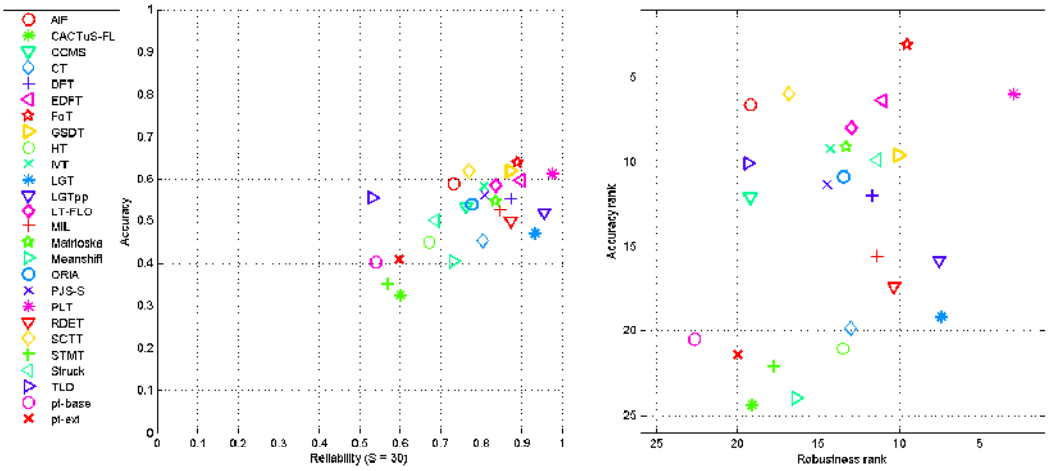
Slika 4.10: Robustnost različic algoritma PixelTrack.



Slika 4.11: A-R grafa eksperimenta *baseline* za vse sledilnike v VOT2013, z dodanima *pt-base* in *pt-ext*.



Slika 4.12: A-R grafa eksperimenta *region_noise* za vse sledilnike v VOT2013, z dodanima *pt-base* in *pt-ext*.



Slika 4.13: A-R grafa eksperimenta *grayscale* za vse sledilnike v VOT2013, z dodanima *pt-base* in *pt-ext*.

Poglavje 5

Sklep

V nalogi je predstavljen razred sledilnih algoritmov, ki uporabljajo generaliziran Houghov transform za sledenje netogih objektov. Analizirali smo teoretično podlago metod, uporabljenih v konkretnem algoritmu PixelTrack in ga z uporabo nekaterih drugih metod skušali izboljšati. V Poglavju 2 so raziskani teoretični vidiki teh metod, ki so bile, kot poroča Poglavje 3, implementirane in vključene v osnovni algoritem. Različice izboljšanega algoritma so bile nato testirane znotraj okolja VOT2013, pridobili smo rezultate in generirali grafe, na podlagi katerih smo evalvirali implementirane spremembe. Kot stranski produkt je nastala tudi hitra (programski jezik C++ in knjižnica OpenCV) in samostojna implementacija kvalitetnega segmentacijskega algoritma. Prioriteta je bila ohraniti algoritem hiter, saj so se k temu nagibali tudi njegovi avtorji, kar sem dosegel z uporabo enostavnih, a učinkovitih metod (Harris, Kalman) in s hitro implementacijo časovno zahtevnejših algoritmov (MRF segmentacija). Nekatere različice algoritma so tako enako hitre kot originalni, a hkrati ponujajo večjo robustnost in natančnost, tudi pod neugodnimi pogoji (eksperiment *region_noise*).

5.1 Nadaljnje delo

Možnosti za nadaljevanje dela v kontekstu te naloge je ogromno. PixelTrack se trenutno ne prilagaja spremembam velikosti sledenega objekta, kar bi se dalo implementirati s prilagajanjem velikosti preiskovanega območja rezultatu segmentacije (če je le-ta dovolj kvalitetna). Ker uporabljen algoritem MRF za segmentacijo ne deluje bolje kot originalni, bi se dalo poiskati druge alternative za segmentacijo, ki niso tako odvisne od barvnih vrednosti, kar je vzrok za večino težav v zvezi s slabo segmentacijo. Mogoča bi bila tudi uporaba bolj prefinjenih opisov značilnih točk kot jih ponuja Harrisova detekcija oglišč. Primeri takšnih algoritmov so denimo FAST [31], SURF [32] ali ORB [33], ki so hitri in proizvedejo deskriptorje značilnih točk v sliki, katerim lahko sledimo preko zaporednih slik v sekvenci. Seveda vsebujejo še precej več informacij kot navadna Harrisova detekcija. Prav tako bi se dalo razširjeni PixelTrack uporabiti v praktični aplikaciji, saj je zaradi hitrosti in fleksibilnosti primeren za uporabo v realnem času.

Literatura

[1] S. S. Nejhum, J. Ho, and M.-H. Yang, "Online visual tracking with histograms and articulating blocks," *Comput. Vis. Image Underst.*, vol. 114, pp. 901–914, Aug. 2010.

[2] N. Paragios and R. Deriche, "Geodesic active contours and level sets for the detection and tracking of moving objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 266–280, Mar. 2000.

[3] V. Lepetit and P. Fua, "Keypoint Recognition using Randomized Trees," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1465 – 1479, 2006.

[4] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 603–619, May 2002.

[5] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-based multiple-person tracking with partial occlusion handling.," in *CVPR*, pp. 1815–1821, IEEE, 2012.

[6] J. Kwon and K. M. Lee in *CVPR*, pp. 1208–1215, IEEE.

[7] M. Godec, P. M. Roth, and H. Bischof, "Hough-based tracking of non-rigid objects," in *ICCV'11*, pp. 81–88, 2011.

[8] S. Duffner and C. Garcia, "PixelTrack: a fast adaptive algorithm for tracking non-rigid objects ," in *International Conference on Computer*

Vision (ICCV 2013), Proceedings of the International Conference on Computer Vision, pp. 2480–2487, Dec. 2013.

[9] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[10] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *In Proc. ECCV*, pp. 661–675, 2002.

[11] A. Yilmaz, X. Li, and M. Shah, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 1531–1536, Nov 2004.

[12] P. H. S. Torr, A. Saffari, and S. Hare, "Efficient online structured output learning for keypoint-based object tracking," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 0, pp. 1894–1901, 2012.

[13] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 798–805, June 2006.

[14] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via online boosting," in *Proceedings of the British Machine Vision Conference*, pp. 6.1–6.10, BMVA Press, 2006. doi:10.5244/C.20.6.

[15] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-n learning: Bootstrapping binary classifiers by structural constraints," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 49–56, June 2010.

[16] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1323–1330, Nov 2011.

[17] D.H. and Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111 – 122, 1981.

[18] J. Gall, A. Yao, N. Razavi, L. V. Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 2188–2202, November 2011.

[19] L. Wang, J. Shi, G. Song, and I.-F. Shen, "Object detection combining recognition and segmentation," in *Proceedings of the 8th Asian Conference on Computer Vision - Volume Part I*, ACCV'07, (Berlin, Heidelberg), pp. 189–199, Springer-Verlag, 2007.

[20] A. Tsai, A. Yezzi, W. W. III, C. Tempany, D. Tucker, A. Fan, W. Grimson, and A. Willsky, "A shape-based approach to the segmentation of medical imagery using level sets," vol. 22, pp. 137–154, 02 2003.

[21] A. George, B. R. Rajakumar, and B. S. Suresh, "Article: Markov random field based image restoration with aid of local and global features," *International Journal of Computer Applications*, vol. 48, pp. 23–28, June 2012. Full text available.

[22] A. Diplaros, N. Vlassis, and T. Gevers, "A spatially constrained generative model and an em algorithm for image segmentation," *IEEETNN*, vol. 18, no. 3, pp. 798–808, 2007.

[23] A. Martins, A. Levada, M. R. P. Homem, and N. Mascarenhas, "Map-mrf super-resolution image reconstruction using maximum pseudo-likelihood parameter estimation," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pp. 1165–1168, Nov 2009.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 39, no. 1, pp. 1–38, 1977.

[25] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, pp. 11–15, Jan. 1972.

[26] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[27] C. Aeschliman, J. Park, and A. C. Kak, "A probabilistic framework for joint segmentation and tracking," in *CVPR'10*, pp. 1371–1378, 2010.

[28] B. Babenko, M.-H. Yang, and S. Belongie, "Visual Tracking with Online Multiple Instance Learning," in *CVPR*, 2009.

[29] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Čehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, A. Khajenezhad, A. Salahledin, A. Soltani-Farani, A. Zarezade, A. Petrosino, A. Milton, B. Bozorgtabar, B. Li, C. S. Chan, C. Heng, D. Ward, D. Kearney, D. Monekosso, H. C. Karaimer, H. R. Rabiee, J. Zhu, J. Gao, J. Xiao, J. Zhang, J. Xing, K. Huang, K. Lebeda, L. Cao, M. E. Maresca, M. K. Lim, M. E. Helw, M. Felsberg, P. Remagnino, R. Bowden, R. Goecke, R. Stolkin, S. Y. Lim, S. Maher, S. Poullot, S. Wong, S. Satoh, W. Chen, W. Hu, X. Zhang, Y. Li, and Z. Niu, "The Visual Object Tracking VOT2013 challenge results," in *ICCV Workshops*, pp. 98–111, 2013.

[30] L. Čehovin, M. Kristan, and A. Leonardis, "Is my new tracker really better than yours?," in *WACV 2014: IEEE Winter Conference on Applications of Computer Vision*, IEEE, Mar 2014.

[31] E. Rosten, G. Reitmayr, and T. Drummond, "Real-time video annotations for augmented reality," in *International Symposium on Visual Computing*, pp. 294–302, 2005.

[32] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, June 2008.

[33] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf," in *ICCV'11*, pp. 2564–2571, 2011.