Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tina Strgar

Metoda za dolgoročno vizualno sledenje z značilnimi točkami

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matej Kristan

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Analizirajte problem dolgoročnega sledenja objektov v videoposnetkih. Osredotočite se na razred sledilnikov, ki temeljijo na modelih z značilnimi točkami. Predlagajte model, ki se bo sposoben prilagajati izgledu tarče, bo omogočal sporočati odpoved sledenja, ko tarča zapusti vidno polje in jo samodejno zaznati ter nadaljevati s sledenjem ob njenem ponovnem vstopu. Postopek analizirajte na javno dostopnih zbirkah in ga primerjajte s sorodnimi metodami.

Analyze the problem of long-term object tracking in video sequences. Focus your analysis on the key-point-based class of trackers. Propose a model that is able to adapt to the changes in target visual appearance, has ability to detect loss of target when the target leaves the field-of-view and autonomously continue with tracking upon re-entering. Analyze your method on a publicly-available dataset and compare it against related approaches.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Tina Strgar, z vpisno številko **63050445**, sem avtorica diplomskega dela z naslovom:

Metoda za dolgoročno vizualno sledenje z značilnimi točkami

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala 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 na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 19. septembra 2014 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
GHT	General Hough Transform	Posplošena Houghova transformacija
HT	Hough Transform	Houghova transformacija
LK	Lucas-Kanade	Lucas-Kanade
NCV	Nearly Constant Velocity	skoraj konstantna hitrost
ROI	Region of Interest	področje zanimanja

Povzetek

V nalogi naslovimo problem dolgoročnega vizualnega sledenja. Izziv predstavljajo dinamično učenje videza tarče, zaznavanje odsotnosti tarče in njeno ponovno detektiranje. Predlagamo sledilnik, ki vizualni model sledene tarče gradi na podlagi lokalnih značilnic in afine preslikave. Dolgoročno sledenje je izvedeno z detekcijo, kratkoročno pa tudi s pomočjo optičnega toka. Pri prileganju preslikave sledilnik uporablja gnezdenje metod: najprej oceni gručo točk, ki potencialno pripadajo tarči, nato pa robustno oceni afino deformacijo. Značilnice za dodajanje modelu so izbrane na podlagi globalne predloge oblike, hkrati pa le-te prispevajo k njenemu posodabljanju, kar tvori povratno zanko. Sledilnik smo testirali na dveh skupinah sekvenc. Prva je namenjena primerjavi dolgoročnih, druga pa kratkoročnih sledilnikov. Rezultate primerjamo s trenutno najnaprednejšimi metodami na področju. Sledilnik se jim v uspešnosti približa, večjo pozornost pa bi bilo potrebno nameniti problemu ponovne detekcije.

Ključne besede: računalniški vid, dolgoročno vizualno sledenje, dinamično učenje, posplošena Houghova transformacija, afina preslikava.

Abstract

In the thesis the problem of long-term visual tracking is addressed. The main challenges of the problem are on-line learning of the target's visual appearance, recognition of target's absence and it's redetection. A part-based tracker is proposed using local features and affine transformation. Long-term tracking is performed with tracking-by-detection, supported by optical flow in the short term. Two nested methods are used when fitting the transformation: firstly, a cluster of potential target points is defined, then the affine deformation is robustly estimated. New model features are added based on the global shape template, that is updated by the features themselves, forming a feedback-loop. The tracker is tested on two groups of sequences, the first targeting long-term and the second short-term trackers. The results are compared with the state-of-the-art methods. The performance of the tracker is comparable, though the problem of redetection should be more carefully addressed.

Keywords: computer vision, long-term visual tracking, online learning, general Hough transform, affine transformation.

Poglavje 1

Uvod in motivacija

Vizualno sledenje je eden izmed pomembnih problemov računalniškega vida. Termin označuje sledenje objektu (tarči) skozi zaporedje slik in je v uspešni izvedbi široko aplikativno. Uporablja se v videonadzoru [14], navigaciji [15], interaktivnih vmesnikih [5] in virtualni resničnosti [8]. Razlikujemo med kratkoročnim in dolgoročnim vizualnim sledenjem. Prvo se nanaša na sledenje objektu, ki je v vizualnem polju ves čas prisoten, le občasno in delno zakrit z drugim elementom, njegov videz pa ni podvržen večjim spremembam. Nasprotno mora dolgoročni sledilnik tarčo po daljši odsotnosti oz. zakritosti ponovno prepoznati in upoštevati spremembe v njeni vizualni pojavnosti, ki nastopijo zaradi sprememb okolice, izvenravninske rotacije, netogosti tarče ipd. Pojavnost objekta abstrahiramo v t. i. *vizualni model*. Ločimo celostne modele (ang. *global*), ki tarčo opisujejo v celoti, ter modele, ki združujejo opise njenih posameznih delov (ang. *part-based*).

Pri sledenju objektov v realnem, nenadzorovanem okolju, izziv predstavljajo spremenljivi pogoji, spreminjanje vizualnih lastnosti ter delno ali popolno zakrivanje (ang. *occlusion*) s kakšnim drugim objektom. Poseben problem predstavlja posodabljanje modela skozi dinamično učenje brez predhodnega poznavanja tarče (ang. *online learning*), kjer uspešnost učenja močno zavisi od točnosti prejšnjega predvidevanja. Z napačnim učenjem se hitro zgodi, da sledilnik zdrsne s tarče in jo zamenja z njeno okolico.

1.1 Pregled področja

Vizualno sledenje je živahno področje računalniškega vida, zato so raznolike tudi metode reševanja problemov, ki pri tem nastopijo. V nadaljevanju so predstavljene najsorodnejše.

V prvi vrsti se metode za sledenje razlikujejo po načinu zapisa izgleda tarče. Uporabljajo se globalni vizualni modeli [16, 27, 2], v zadnjem času pa se uveljavljajo predvsem metode, ki opisujejo posamezne dele tarče (ang. *part-based tracking*) [23, 22, 12, 30, 9]. Slednje pri sledenju predpostavljajo začasno geometrijsko skladnost delov, ki jo ocenjujejo z robustnimi metodami prileganja modelov, kot sta prileganje z vzorčenjem [23] in posplošena Houghova transformacija [22]. Če dele tarče predstavljajo lokalne značilnice, je geometrijska skladnost lahko upoštevana že pri sami detekciji le-teh [12]. Predpostavlja se globalna skladnost [23, 12, 22] ali pa lokalna [27, 9]. Metode, ki predpostavljajo globalno skladnost, se nadalje razlikujejejo po stopnji kompleksnosti geometrijskih sprememb, ki jo lahko zajamejo. V zadnjem času sta se pojavila dva sledilnika, ki problem dolgoročnega sledenja še posebej uspešno naslavljata: PREDATOR [16] in ALIEN [23]. PREDATOR se prilagajata spremembi velikosti, kjer spremembi v x in y smeri nista korelirani. ALIEN [23] upošteva preslikavo podobnosti, kjer je faktor skaliranja v obeh smereh enak, in zajema tudi rotacijo.

Pri učenju modela se sledilniki večinoma zanašajo na ocenjeno področje tarče [23, 22] v vsakem časovnem koraku. K problemu klasifikacije vizualne informacije za posodabljanje modela in sledenje se pristopa z uporabo pozitivnih in negativnih učnih primerov [16, 23, 2, 30], kjer se uporablja informacijo ozadja tarče. Tudi problem zakrivanja metode rešujejo na podlagi negativnih učnih primerov [16, 23, 2, 30] ali pa implicitno [22].

1.2 Prispevki diplomske naloge

V nalogi predlagamo dolgoročni vizualni sledilnik, kjer vizualni model gradimo s pomočjo lokalnih značilnic. Model posodabljamo na način, ki upošteva spremembe v videzu in ohranja opise, ki so pomembni za dolgoročno poznavanje in prepoznavanje tarče.

Tekom sledenja morebitne položaje tarče določimo s posplošeno Houghovo

transformacijo [3] in točkam, ki transformacijo podpirajo, poiščemo afino preslikavo. Ta zajema skaliranje v x in y smeri ter rotaciji baznih vektorjev. Zaradi gnezdenja metod prileganja modela podatkom je sledilnik robusten v širokem naboru situacij.

Za reševanje problema filtriranja osamelcev (ang. *outliers*) v nasprotju z [16, 23] ne uporabljamo negativnih primerov, temveč za to predlagamo uporabo povratne informacije točk, ki ocenjeno preslikavo podpirajo. S tem gradimo globalno predlogo oblike, ki služi kot filter pri dodajanje značilnic modelu ter značilnih točk množici za ocenjevanje optičnega toka. Problem zakrivanja s tem rešujemo implicitno.

1.3 Struktura diplomske naloge

Delo je razdeljeno na pet poglavji. V Poglavju 2 predstavimo teoretično ozadje uporabljenih metod. Poglavje 3 opisuje razvit algoritem. Najprej podrobneje opišemo gradnjo vizualnega modela in način posodabljanja le-tega, nato pa celotno zasnovo sledenja. V Poglavju 4 predstavimo metodo vrednotenja algoritma, rezultate analiziramo in jih primerjamo s konkurenčnimi sledilniki. V Poglavju 5 povzamemo sklepne ugotovitve in navedemo možnosti za nadaljnji razvoj sledilnika.

Poglavje 2

Teorija

V poglavju so predstavljene metode, na katerih temelji zasnova lastnega sledilnika. V Podpoglavju 2.1 opišemo lokalno značilne točke in lokalne značilnice ter podrobneje predstavimo značilne točke GFTT [26] ter značilnice SURF [4]. Nato v Podpoglavju 2.2 predstavimo dve metodi prileganja podatkov modelu: posplošeno Houghovo transformacijo [3] in metodo MLESAC [28]. V Podpoglavju 2.3 predstavimo optični tok Lucas-Kanade ter v Podpoglavju 2.4 dinamični model in Kalmanov filter [17].

2.1 Lokalne značilnice

Lokalne značilnice omogočajo izražanje lastnosti področij v sliki, kot so barva, tekstura in oblika, s številčnimi vrednostmi. Ločimo med značilnimi točkami (ang. *keypoints*), ki jih podajamo s koordinatami, in lokalnimi značilnicami (ang. *local features*), ki podajo opis regije okoli značilne točke. Značilnico zgradimo v štirih korakih: 1) detektiranje značilne točke, 2) definiranje regije okoli točke, 3) normalizacija regije (neobvezno) in 4) izračun lokalnega deskriptorja (ang. *feature vector*) .

Lokalne značilnice uporabljamo predvsem pri problemih iskanja korespondenc med slikami, kot so ustvarjanje panoram, poizvedovanje po slikovnih bazah, detekcija objektov, vizualno sledenje itn., za kar potrebujemo 1) ponovljivost detekcij in 2) opis, ki bo dobro služil za prepoznavanje podobnih in razlikovanje različnih značilnic, hkrati pa čim manj občutljiv na vizualne spremembe, ki niso vezane na

strukturo področja tarče.

Tako za detekcijo kot za gradnjo značilnic obstaja veliko metod. Med popularnejšimi so: KAZE [1], FREAK (ang. *Fast Retina Keypoint*) [29], BRISK (ang. *Binary Robust Invariant Scalable Keypoints*) [19], ORB (ang. *Oriented FAST and Rotated BRIEF*), SURF (ang. *Speeded Up Robust Features*) [4], SIFT (ang. *Scale-invariant Teature Transform*) [20].

Pri dolgotrajnem opazovanju objektov v nenadzorovanem okolju želimo značilnice, katerih opis je čim manj občutljiv na zunanje spremembe, vezane na osvetljavo, strukturo ozadja in artefakte samega zajema slike, ter notranje spremembe, vezane predvsem na prostorske transformacije. Za take značilnice pravimo, da so invariantne.

Deskriptorje značilnic med seboj primerjamo s funkcijo razdalje d, ki je opredeljena glede na lastnosti deskriptorja. Posebno aktualni so binarni deskriptorji, ki jih primerjamo s Hammingovo razdaljo [11] in omogočajo kaskadno primerjanje.

2.1.1 Značilne točke GFTT in značilnice SURF

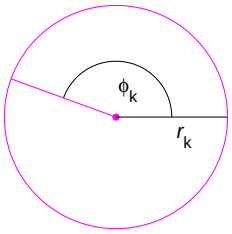
Značilne točke GFTT (ang. *Good Features To Track*) [26] uporabljamo za iskanje korespondenc pri ocenjevanju optičnega toka. Najdemo jih na mestih, kjer so izrazite svetlostne spremembe v dveh pravokotnih smereh (npr. oglišča). Definiramo jih s pomočjo lastnih vrednosti Hessianove matrike. Značilno točko opredelimo kot $\boldsymbol{\psi}_k = \{x_k, y_k\}$, kjer sta x_k in y_k koordinati k-te točke.

Značilnice SURF (ang. *Speeded Up Robust Features*) [4] so izboljšana verzija popularnih značilnic SIFT [20]. So rotacijsko in velikostno invariantne. Detekcija točk temelji na determinanti Hessianove matrike, kjer se jo aproksimira z uporabo t. i. filtri Box nad integralnimi slikami. Operacije so preproste in primerne za aplikacijo v realnem času. Deskriptor opisuje porazdelitev Haarovih valčnih odzivov in je lahko rotacijsko normaliziran.

Ob detekciji značilnico opišemo kot $\theta_k = \{x_k, y_k, r_k, \phi_k, \boldsymbol{\eta}_k\}$, kjer sta x_k in y_k koordinati značilnice v slikovnem okviru, r_k in ϕ_k pa velikost in rotacija značilnice. Vektor $\boldsymbol{\eta}_k$ je deskriptor značilnice, na Sliki 2.1 ponazorjen z barvo. Za primerjanje

deskriptorjev značilnic SURF se uporablja evklidska norma oz. razdalja l_2:

$$d(\boldsymbol{\eta}_1, \boldsymbol{\eta}_2) = \sqrt{\sum_{i=1:n} \left(\eta_i^{(1)} - \eta_j^{(2)}\right)^2}, \text{ kjer } \boldsymbol{\eta}_1, \boldsymbol{\eta}_2 \in \mathbb{R}^i. \tag{2.1}$$



Slika 2.1: Značilnica z definiranim kotom ϕ in velikostjo r.

2.2 Robustno prileganje modelov

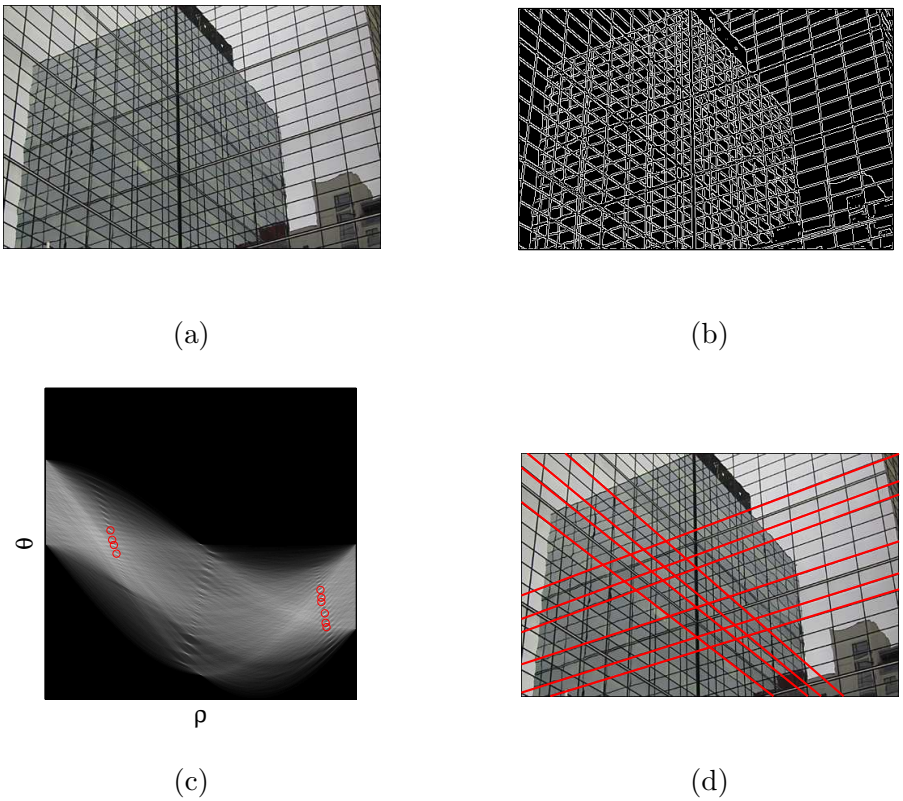
Pri prileganju modela nas zanima transformacija $T_t(M_t, I_t)$, ki model v času t M_t najbolje prilega podatkom I_t glede na definirano kriterijsko funkcijo. Glavni izziv prilaganja modelov so šumni podatki, ki jih je v realnih primerih lahko celo več kot polovico.

V nadaljevanju predstavimo dve metodi: 1) Posplošeno Houghovo transformacijo [3], ki je robustna in hitra pri manjšem številu iskanih parametrov, ter 2) metodo prilaganja z vzorčenjem MLESAC [28], s katero lahko ocenjujemo bolj kompleksne transformacije, a je kljub dobri robustnosti na šum bolj občutljiva.

2.2.1 Posplošena Houghova transformacija

Houghova transformacija (HT, ang. *Hough Transform*) [13] je popularna metoda iskanja primitivnih parametričnih geometrijskih oblik. Je metoda glasovanja v n-dimenzionalnem Houghovem prostoru, kjer n ustreza številu iskanih parametrov. Zaloge vrednosti parametrov so diskretizirane. Entitete (podatki, npr. piksli, točke, značilnice) nato glasujejo za vse modele (kombinacije parametrov modela), s katerimi so skladne. Maksimumi glasovalnega prostora ustrezajo najbolj verjetnim kombinacijam parametrov. Primer uporabe HT za detekcijo premic prikazuje Slika 2.2.

Metoda dobro deluje tudi ob veliki količini šumnih podatkov, primerna pa je za ocenjevanje modelov z manjšim številom parametrov, saj je računsko zahtevna. Slaba stran metode je, da lahko lokalni maksimum v prostoru tvorijo tudi samo šumni podatki. V izogib temu se v praksi uporablja predhodno znanje za omejevanje možnih vrednosti parametrov in uteževanje glasov. Na učinkovitost metode prav tako vpliva kvantizacija zalog vrednosti.

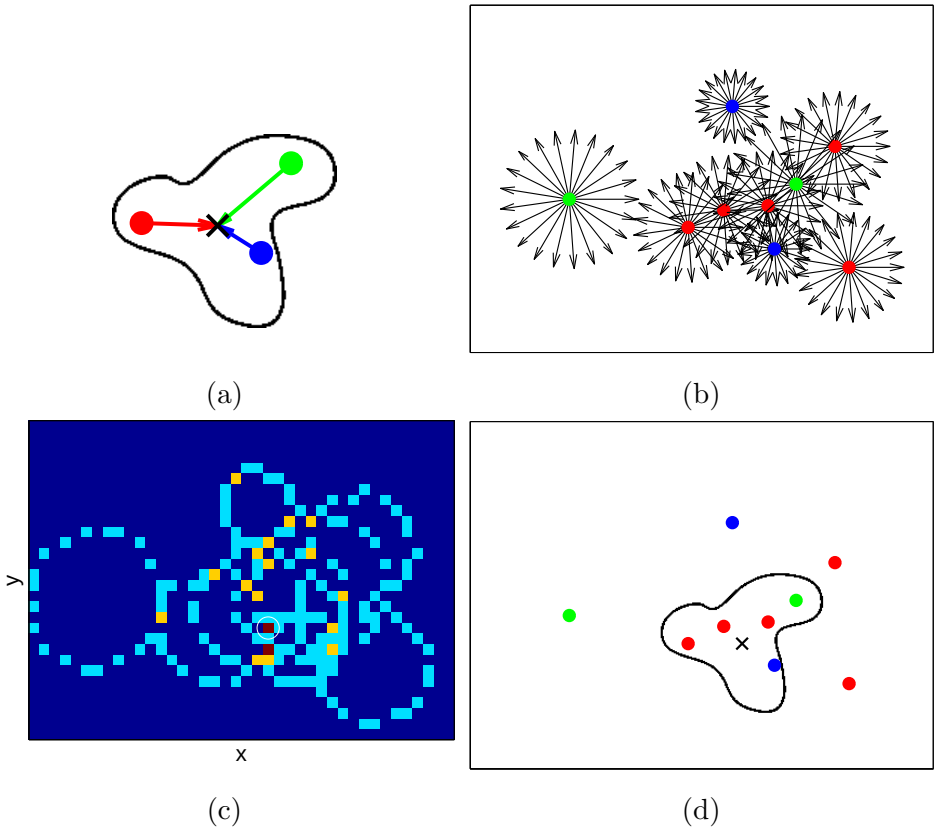


(a) (b) (c) (d)

Slika 2.2: Houghova transformacija za detekcijo premic. Premico izrazimo v polarni obliki $x\cos\theta - y\sin\theta = \rho$. Za vsak robni piksel (b) se izračuna ρ za vse možne vrednosti θ in poveča vrednost ustreznih celic v Houghovem prostoru. (c) Maksimumi definirajo najbolj verjetne pare (ρ, θ). (d) 10 premic z največjo podporo.

Posplošena Houghova transformacija (ang. *Generalised Hough Transform*, GHT) [3] omogoča iskanje splošnih objektov, opisanih z modelom in ne z ana-

litično enačbo, kot velja za HT. Iskanje objekta prevedemo na iskanje njegovega položaja v sliki z iskanjem parametrov transformacije, ki model preslika v slikovno polje. Primer GHT prikazuje Slika 2.3.



Slika 2.3: Posplošena Houghova transformacija. (a) Modelu definiramo središče in za vsako od treh barvnih pik izračunamo razdaljo do njega. (b) Ob detekciji vsaka barva glasuje v Houghovem prostoru v vse smeri v izračunani razdalji. (c) Diskretiziran Houghov prostor po glasovanju. (d) Maksimum prostora določa novo središče objekta.

2.2.2 Prileganje z vzorčenjem

Metode prileganja z vzorčenjem temeljijo na soglasju naključnega vzorčenja in so posebno primerne za naslavljanje problemov, za katere sicer poznamo analitične rešitve, ki pa so precej občutljive na šum.

Splošno transformacijo opredelimo kot $\boldsymbol{A}\boldsymbol{\chi} = \boldsymbol{\gamma}$, kjer so $\boldsymbol{A}$ matrika preslikave, $\boldsymbol{\chi}$ podatki modela in $\boldsymbol{\gamma}$ izmerjeni podatki. Če je kriterijska funkcija zastavljena na podlagi razlike med dejanskimi in preslikanimi podatki, je preslikava optimizirana tudi glede na osamelce (ang. *outliers*) in tako hitro popačena.

Obstaja več metod prileganja z vzorčenjem, vse pa temeljijo na popularnem algoritmu RANSAC (ang. *Random Sample Consensus*) [10]. Osnovni koraki algoritma so: 1) naključno vzorčenje najmanjšega števila podatkov, potrebnih za oceno preslikave, 2) izračun preslikave $\boldsymbol{A}$ in 3) izračun števila podatkov, ki preslikavo podpirajo (ang. *inliers*) glede na funkcijo napake E.

Algoritem korake iterativno ponavlja, izhodna preslikava $\boldsymbol{A}$ je preslikava z največjo podporo. Število iteracij je lahko določeno vnaprej tako, da glede na ocenjen odstotek šumnih podatkov maksimiziramo verjetnost vzorčenja podatkov, ki dejansko pripadajo modelu. Odstotek šumnih podatkov lahko ocenjujemo tudi tekom algoritma in število iteracij dinamično prilagajamo.

Metoda MLESAC (ang. *Maximum Likelihood Estimation Sample Consensus*) [28] gre korak naprej in preslikave, generirane na podlagi vzorčenih podatkov, vrednoti na podlagi logaritemske funkcije verjetja. Za vsako preslikavo $\boldsymbol{A}_i$ najprej izračuna napako $\delta_k^{(i)} = E(\boldsymbol{A}_i, k)$ za vsak podatek k, na podlagi katere določi verjetnost, da podatek pripada modelu, p_{in} in verjetnost, da ne pripada modelu, p_{out} po enačbah

$$p_{in}(k|\boldsymbol{A}_i) = \lambda \frac{e^{-\frac{{\delta_k^{(i)}}^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \qquad , \qquad p_{out}(k|\boldsymbol{A}_i) = \frac{1-\lambda}{\mu}, \tag{2.2}$$

kjer je λ t. i. faktor mešanja, σ varianca normalne porazdelitve in μ globalna maksimalna napaka, definirana kot maksimalna evklidska razdalja med podatki. Faktor λ je inicializiran na $1/2$, nato pa se ga računa interativno po

$$\lambda = \frac{1}{n} \sum_{k=1:n} \frac{p_{in}^{(k)}}{p_{in}^{(k)} + p_{out}^{(k)}}, \tag{2.3}$$

kjer je n število podatkov. Končna transformacija $\boldsymbol{A}_{opt}$ je določena po

$$\underset{\boldsymbol{A}_i}{\operatorname{argmax}} \Big(- \sum_{k=1:n} \ln \left(p_{in}^{(i,k)} + p_{out}^{(i,k)} \right) \Big). \tag{2.4}$$

2.3 Optični tok Lucas-Kanade

Optični tok označuje navidezno gibanje v slikovnem okvirju. Gre za ocenjevanje korespondenc med zaporednima slikama $\boldsymbol{I}_t$ in $\boldsymbol{I}_{t+\Delta t}$. Ocenjuje se ga lahko gosto za vsak piksel slike, ali pa razstreseno za regije okoli določenih točk.

Metoda za ocenjevanje optičnega toka Lucas-Kanade (metoda LK) sloni na dveh predpostavkah: 1) svetlost piksla po premiku v času Δt ostaja enaka (ang. *brightness constancy constraint*), 2) premik piksla v času Δt bo majhen [21].

Prvo predpostavko predstavlja enačba

$$\boldsymbol{I}(x, y, t) = \boldsymbol{I}(x + \Delta x, y + \Delta y, t + \Delta t), \tag{2.5}$$

na podlagi druge pa jo lahko razvijemo v Taylorjevo vrsto po

$$\boldsymbol{I}(x + \Delta x, y + \Delta y, t + \Delta t) \approx \boldsymbol{I}(x, y, t) + \frac{\partial \boldsymbol{I}}{\partial x}\Delta x + \frac{\partial \boldsymbol{I}}{\partial y}\Delta y + \frac{\partial \boldsymbol{I}}{\partial t}\Delta t. \tag{2.6}$$

Enačbo (2.6) izpeljemo v

$$\boldsymbol{I}_x \Delta x + \boldsymbol{I}_y \Delta y = -\boldsymbol{I}_t, \tag{2.7}$$

ki jo imenujemo enačba optičnega toka (ang. *optical flow constraint equation*), kjer sta $\boldsymbol{I}_x$ in $\boldsymbol{I}_y$ odvoda v x in y smeri.

Enačba (2.7) ima dve neznanki, zato je ne moremo oceniti na podlagi enega piksla. Gre za t. i. problem odprtine (ang. *aperture problem*), ki ga metoda LK rešuje s predpostavljanjem enakega toka v soseščini piksla. S tem zastavimo predeterminiran sistem, ki se ga rešuje z metodo najmanjših kvadratov in psevdoinverzom. Omejitev majhnih premikov se lahko omili s širjenjem toka skozi Gaussovo piramido [6].

2.4 Dinamični model in Kalmanov filter

Dinamični model opisuje obnašanje sistema skozi čas. V nadaljevanju razlage se omejimo na uporabo na področju računalniškega vida in osredotočimo na model skoraj konstantne hitrosti (ang. *nearly constant velocity*, NCV).

Dinamičnimi modeli večinoma opisujejo lastnosti spreminjanja položaja objekta. Na podlagi stanja modela v času t lahko predvidimo položaj v času $t + 1$. Ločimo

med implicitnimi in eksplicitnimi, deterministič nimi in stohastič nimi dinamič nimi modeli.

Stanje modela v času t opredelimo kot $\hat{\boldsymbol{x}}_t = [x, x', \ldots]^\mathsf{T}$, sam model pa tipično predstavimo z diferencialnimi enačbami. Model skoraj konstantne hitrosti je ekspliciten stohastič en model, za dvorazsežni prostor opredeljen z enačbami

$$\hat{\boldsymbol{x}}_t = \begin{bmatrix} x & y & x' & y' \end{bmatrix}^\mathsf{T} = \boldsymbol{\Phi}(\Delta t)\hat{\boldsymbol{x}}_{t-1} + W_{t-1} \quad , \quad \hat{\boldsymbol{x}}'_t = \boldsymbol{F}\hat{\boldsymbol{x}}_{t-1} + \boldsymbol{L}w, \tag{2.8}$$

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

$$\boldsymbol{F} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad , \quad \boldsymbol{L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.9}$$

kjer $W_{t-1} \sim G(\mu = 0, \boldsymbol{Q})$ in q predstavlja spektralno gostoto šuma. Pri obdelavi zaporedja slik je čas diskreten in velja $\Delta t = 1$.

Matrika $\boldsymbol{Q}$ predstavlja nekoreliran dinamični šum sistema. Če enačbe interpretiramo, model v času t: 1) predvidi položaj objekta na podlagi prejšnjega položaja in hitrosti, 2) trenutni hitrosti doda šum. NCV model torej predpostavlja korelirano hitrost in nekoreliran pospešek.

Za združevanje napovedi dinamičnega modela z meritvami uporabljamo rekurzivni Bayesov filter, imenovan Kalmanov filter [17]. Ta je primeren pri predpostavljanju linearne dinamike in Gaussove distribucije tako meritve kot napovedi. Kalmanov filter podajajo enačbe

$$\begin{aligned} \tilde{\boldsymbol{x}}_t &= \boldsymbol{\Phi}\hat{\boldsymbol{x}}_{t-1} + \Gamma\boldsymbol{\mu}_t && , && \tilde{\boldsymbol{P}}_t = \boldsymbol{\Phi}\boldsymbol{P}_{t-1}\boldsymbol{\Phi}^\mathsf{T} + \boldsymbol{Q}, \\ \boldsymbol{K} &= \tilde{\boldsymbol{P}}_t\boldsymbol{H}^\mathsf{T}\left(\boldsymbol{H}\tilde{\boldsymbol{P}}_t\boldsymbol{H}^\mathsf{T} + \boldsymbol{R}\right)^{-1} && , && \\ \hat{\boldsymbol{x}}_t &= \tilde{\boldsymbol{x}}_1 + \boldsymbol{K}\left(\boldsymbol{y}_t - \boldsymbol{H}\tilde{\boldsymbol{x}}_1\right) && , && \boldsymbol{P}_t = \left(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}\right)\tilde{\boldsymbol{P}}_1, \end{aligned} \tag{2.10}$$

kjer $\boldsymbol{P}_{t-1}$ predstavlja apriorno verjetnost, $\boldsymbol{P}_t$ posteriorno verjetnost, $\boldsymbol{y}_t$ meritev, $\boldsymbol{H}$ opazovalno matriko in $\boldsymbol{I}$ indentiteto.

Poglavje 3

Zasnova algoritma za sledenje

V tem poglavju podrobneje predstavimo algoritem za sledenje. V Podpoglavju 3.1 opredelimo afino preslikavo, s pomočjo katere v Podpoglavju 3.2 definiramo dolgoročni vizualni model. Način posodabljanja modela je predstavljen v Podpoglavju 3.3. V Podpoglavju 3.4 nato opišemo delovanje sledilnika: sledenje z detekcijo, kratkoročno sledenje z optičnim tokom, posodabljanje stanja sledilnika in na koncu sledilnik še grafično ponazorimo.

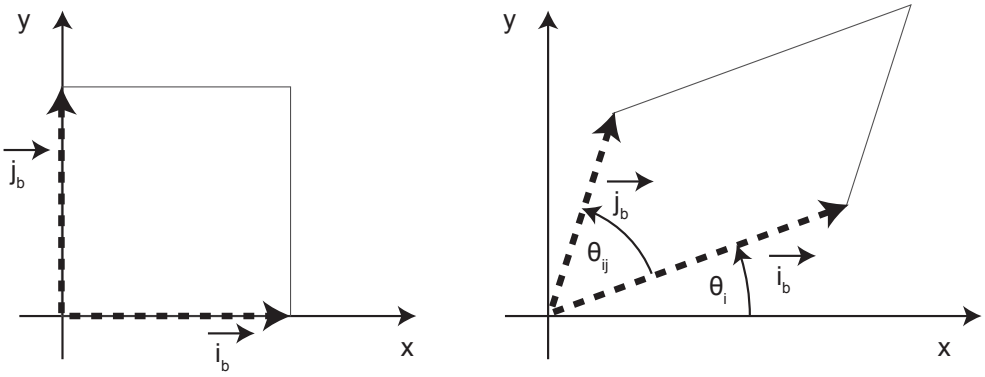
3.1 Afina preslikava

Afina preslikava je linearna preslikava, ki ohranja premice in ravnine. Ohranja vzporednost, ne pa nujno tudi kotov med premicami. Primer matrike za afino preslikavo $\boldsymbol{A}$ v dvorazsežnem prostoru podaja enačba

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \quad , \quad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}' = \boldsymbol{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y + a_{13} \\ a_{21}x + a_{22}y + a_{23} \\ 1 \end{bmatrix}. \tag{3.1}$$

Vektor $[a_{13}, a_{23}]^\intercal$ predstavlja premik, ostali parametri pa niso enostavno razložljivi in vzajemno določajo skaliranje in rotiranje baznih vektorjev. V literaturi se rotiranje baznih vektorjev včasih interpretira tudi kot eno rotacijo in zamik (ang. *skew*). Primer preslikave prikazuje Slika 3.1.

Matrika $\boldsymbol{A}$ ima 6 neznanih parametrov, zato jo lahko ocenimo z vsaj tremi nekolinearnimi točkami (vsaka točka prispeva x in y vrednost.) Če z $\boldsymbol{\chi}$ označimo



Slika 3.1: Afino preslikana bazna vektorja brez translacije

podatke, ki jih želimo z $\boldsymbol{A}$ preslikati v podatke $\boldsymbol{\chi}'$, lahko zapišemo $\boldsymbol{A}\boldsymbol{\chi} = \boldsymbol{\chi}'$ in definiramo funkcijo napake $\epsilon(\cdot)$ kot

$$\epsilon(a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{33}) = \sum_{i=1:n} ((a_{11}x_i + a_{12}y_i + a_{13} - x_i')^2 + (a_{21}x_i + a_{22}y_i + a_{23} - y_i')^2), \tag{3.2}$$

kjer je n število podatkov (korespondenc) in $n_{min} = 3$. Funkcijo ϵ se minimizira z metodo najmanjših kvadratov, ki je zelo občutljiva na šum, zato jo uporabimo v metodi RANSAC [10] (glej Podpoglavje 2.2.2).

3.2 Vizualni model

Vizualni model definiramo na podlagi lokalnih značilnic θ in afine preslikave $\mathbf{A}$. Uporabimo značilnice SURF [4]. V času t je definiran kot množica $M_t = \{w_{i=1:n^{(t)}}\}$, kjer $w_i = \left\{\boldsymbol{\eta}_i, \boldsymbol{\chi}^{(i)}_{j=1:m^{(i)}}\right\}$ in $\boldsymbol{\chi}_j = [\boldsymbol{\delta}_p, \boldsymbol{\delta}_c]^\intercal$. Vektor $\boldsymbol{\chi}_k$ izračunamo za vsako značilnico $\theta_k = \left\{x_k, y_k, r_k, \phi_k, \boldsymbol{\eta}^{(i)}_k\right\}$ (glej Podpoglavje 2.1.1 in Sliko 2.1), ki jo

dodajamo modelu, glede na trenutno ocenjeno preslikavo $\boldsymbol{A}_t$ po

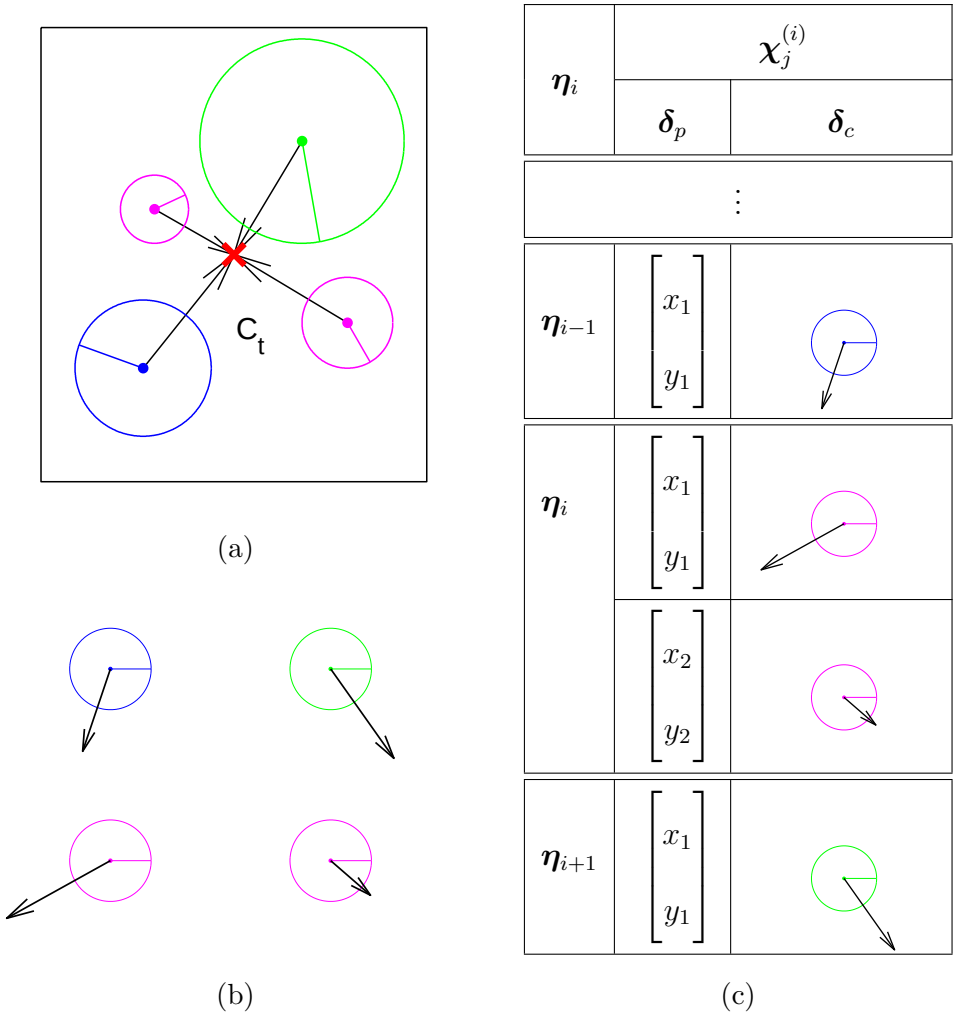
$$d = \frac{\sqrt{(c_y - y_k)^2 + (c_x - x_k)^2}}{r_k} \quad , \quad \gamma = atan2(c_y - y_k, c_x - x_k),$$

$$\boldsymbol{\delta}_p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A}_t^{-1} \begin{bmatrix} x_k \\ y_k \\ 1 \end{bmatrix} \quad , \quad \boldsymbol{\delta}_c = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = d \begin{bmatrix} \cos(\gamma - \phi_k) \\ \sin(\gamma - \phi_k) \end{bmatrix}, \tag{3.3}$$

kjer je $\boldsymbol{C} = [c_x, c_y]^\intercal$ središče objekta. Vektor $\boldsymbol{\delta}_c$ uporabimo pri GHT, vektor $\boldsymbol{\delta}_p$ pa pri prileganju afine transformacije. Grafično model ponazarja Slika 3.2c.

Za takšno zasnovo modela smo se odločili zaradi več razlogov. Dopuščamo, da se značilnica znotraj modela pojavlja dvoumno, na večih mestih hkrati, bodisi zaradi dveh podobnih struktur bodisi zaradi netoge notranje strukture objekta. Za značilnico, katere deskriptor tekom sledenja rahlo variira, tako ni potrebno hraniti vseh njenih opisov, kar zmanjša časovno zahtevnost algoritma.

Za značilnico, ki se znotraj tarče ves čas nahaja na istem mestu, lahko hranimo več različnih vektorjev do središča tarče. To je željeno, saj se pri izračunu $\boldsymbol{\delta}_c$ zanašamo na oceno velikosti in rotacije značilke, ki ima omejeno natančnost. V primeru, ko so vektorji bolj ali manj enaki, z večjo gotovostjo glasujejo v Houghovem prostoru. Prav tako lahko za takšne značilnice hranimo več podobnih ali celo enakih vektorjev $\boldsymbol{\delta}_p$, s čimer imajo večjo težo pri ocenjevanju preslikave $\boldsymbol{A}_t$ z metodo MLESAC [28] v koraku detekcije (glej Podpoglavje 3.4.1).



(a) (b) (c)

Slika 3.2: Vizualni model na podlagi lokalnih značilnic. (a) Značilnice ob detekciji in vektorji do centra objekta C_t. (b) Vektorji, normalizirani glede na velikost in rotacijo značilnic. (c) Vizualni model po vnosu detektiranih značilnic.

3.3 Posodabljanje modela

Podpoglavje je razdeljeno na tri dele. Najprej predstavimo način filtriranja detektiranih značilnic. Nato opišemo postopka dodajanja in odstranjevanja značilnic

modelu.

3.3.1 Filtriranje značilnic

Filtriranje značilnic za dodajanje modelu in sledenje po metodi LK opravimo na podlagi globalne predloge, ki predstavlja trenutno prostorsko verjetnostno porazdelitev oblike tarče.

Predlogo definiramo kot kvadratno matriko $\boldsymbol{F}$ velikosti λ_{m_size}, kjer $\mathcal{Z}_f(\boldsymbol{F}) = [0, 1]$. Pri filtriranju jo najprej preslikamo v trenutni slikovni okvir na podlagi ocenjene $\boldsymbol{A}_t$ in jo označimo z $\boldsymbol{F'}$. Definirajmo še funckijo $\Upsilon(\boldsymbol{A}, \boldsymbol{F}, \lambda)$ kot

$$\begin{aligned} \Upsilon(\boldsymbol{A}, \boldsymbol{F}, \lambda) &= \boldsymbol{F} \odot \boldsymbol{J}, \text{ kjer} \\ \boldsymbol{J} &= \begin{cases} 1 & , \text{če } j \in P \\ 0 & \text{drugače} \end{cases} \end{aligned} \tag{3.4}$$

in je P paralelogram, ki ga določata vektorja $\boldsymbol{A}_t\lambda\boldsymbol{i}_b$ in $\boldsymbol{A}_t\lambda\boldsymbol{j}_b$.

Detektirane značilne točke $\boldsymbol{\psi}_i = [x_i, y_i]^\intercal$ dodamo množici za kratkoročno sledenje, če $\boldsymbol{F}_1(x_i, y_i) > \lambda_c$, kjer je $\boldsymbol{F}_1 = \Upsilon(\boldsymbol{A}_t, \boldsymbol{F'}, \lambda_{out})$ in $\lambda_{out} \geq 1$. Z $\boldsymbol{F}(n, m)$ označujemo element matrike v n-ti vrstici in m-tem stolpcu.

Pri izboru značilnic $\theta_i = \{x_i, y_i, r_i, \phi_i, \boldsymbol{\eta}_i\}$ za dodajanje modelu smo bolj selektivni. Poleg dovolj velike vrednosti na koordinati točke nas zanimajo vrednosti predloge na celotni regiji, ki pripada značilnici glede na njeno velikost r_i. Značilnico sprejmemo, če $\boldsymbol{F}_2(x_i, y_i) > \lambda_d$ in $k\sum_j \boldsymbol{F'}(x_j, y_j) > \lambda_a$, kjer $\boldsymbol{F}_2 = \Upsilon(\boldsymbol{A}_t, \boldsymbol{F'}, \lambda_{in})$, $\lambda_{in} \leq 1$, $k = \frac{1}{2\pi r_i}$ in $\forall j : (x_j - x_i)^2 + (y_j - y_i)^2 < r_i$.

3.3.2 Rast modela

Množico v času t detektiranih značilnic, ki smo jih uspešno filtrirali glede na globalno predlogo oblike, označimo z Ω_t. Deskriptorje značilnic $\theta_j \in \Omega_t$ s funckijo razdalje d (2.1) primerjamo z deskriptorji modela M_{t-1} in s tem znotraj Ω_t definiramo dve podmnožici Ω_k ter Ω_s, kjer

$$\begin{aligned} \Omega_k &= \left\{\theta_{j=1:n^{(k)}}\right\}, \forall\theta_j : min(d(\boldsymbol{\eta}_j, \boldsymbol{\eta}_{\boldsymbol{i=1:n}^{(t)}})) < \lambda_1, \\ \Omega_s &= \left\{\theta_{j=1:n^{(s)}}\right\}, \forall\theta_j : min(d(\boldsymbol{\eta}_j, \boldsymbol{\eta}_{\boldsymbol{i=1:n}^{(t)}})) < \lambda_2, \end{aligned}$$

$\boldsymbol{\eta}_{i=1:n^{(t-1)}}$ pa so deskriptorji modela M_{t-1}. Značilnice $\theta_j \in \Omega_t$ nato M_{t-1} dodajamo v dveh korakih. Najprej podamo postopek dodajanja novih vnosov v model, nato pa posodabljanje znanih vnosov $w_i \in M_{t-1}$.

Nove vnose v model definiramo s pomočjo množice $\Omega_u = \Omega_t \setminus \Omega_k$. Deskriptorje značilnic $\theta_j \in \Omega_u$ najprej razvrstimo v gruče po metodi voditeljev (ang. *k-means*). Pri tem iščemo minimalni k (število gruč), za katerega velja $\forall \boldsymbol{\eta}_j : d(\boldsymbol{\eta}_j, \boldsymbol{\eta}_j^{(r)}) < \lambda_k$, kjer $\boldsymbol{\eta}_j^{(r)}$ označuje središče gruče r, ki mu pripada značilnica θ_j, in $r = 1 : k$.

Za oceno minimalne vrednosti k_{min} deskriptorje $\boldsymbol{\eta}_j$ med seboj primerjamo s funkcijo d. Izračunamo število specifičnih deskriptorjev n_{spec} in k_{min} določimo po

$$n_{spec} = \sum_{j=1}^{n^{(u)}} \delta(j), \text{ kjer } \delta(j) = \begin{cases} 1 & \text{, če } min(d(\boldsymbol{\eta_j}, \boldsymbol{\eta_{i=1:n^{(u)}}})) > 2\lambda_k, \\ 0 & \text{drugače} \end{cases},$$

$$k_{min} = min\left(n^{(u)}, n_{spec} + 1\right). \tag{3.5}$$

Začetno vrednost k_{min} nato iterativno povečujemo, dokler ne dosežemo opredeljenega pogoja. Naenkrat lahko dodamo omejeno število deskriptorjev. Pri presežnem številu se odločimo za tiste, ki so glede na funkcijo d najbolj specifični. Za vsako gručo r izračunamo vnos $w_r = \left\{\boldsymbol{\eta_r}, \boldsymbol{\chi}_{j=1:m^{(r)}}^{(r)}\right\}$ tako, da po (3.3) izračunamo $\boldsymbol{\chi}_j$ za vsako značilnico gruče. Vnose nato dodamo v model in dobimo $M_t = M_{t-1} \cup \{w_{r=1:k}\}$.

Znane vnose $w_i \in M_{t-1}$ posodabljamo s pomočjo množice $\Omega_v = \Omega_k \cup \Omega_s$. Za vsako značilnico $\theta_j \in \Omega_v$ izračunamo $\boldsymbol{\chi}_j$ in ga dodamo ustreznemu vnosu w_i. Pri tem preverimo, če in kolikokrat se je $\boldsymbol{\delta}_p^{(j)}$ v w_i že pojavil. Frekvenco pojavljanja omejimo, da preprečimo prekomerno prileganje (ang. *overfitting*).

3.3.3 Čiščenje modela

Naj vizualni model v času t še enkrat zapišemo kot $M_t = \{w_{i=1:n^{(t)}}\}$, kjer $w_i = \left\{\boldsymbol{\eta}_i, \boldsymbol{\chi}_{j=1:m^{(i)}}^{(i)}\right\}$. Z dodajanjem novih značilnic povečujemo tako n kot m_i. Določimo pragova n_{max} in m_{max}, ki predstavljata največje dovoljeno število n in m_i. Kadar katera od vrednosti prag preseže, model oklestimo (ang. *prune*).

Najprej predstavimo klestenje množice M. Pri tem želimo ohraniti vnose w_i, ki zajemajo pojavnost tarče skozi daljše časovno obdobje, in odstraniti vnose, ki ne pripadajo objektu oz. zanj niso specifični, kot so razni vizualni artefakti (npr. posledica zajema slike, hipne spremembe razsvetljave ipd.). Vnose w_i zato

ohranjamo glede na m_i. Najprej odstranimo vnose, za katere velja $m_i = 1$, kar pomeni, da smo značilnico znotraj objekta opazili samo enkrat. Če po tem n še vedno presega n_{max}, ohranimo največ $n_{max}/2$ vnosov, in sicer tiste z največjimi m, po potrebi pa tiste z najmanjšim sprejemljivim m uniformno vzorčimo. Posebej je potrebno izpostaviti, da model klestimo pred dodajanjem novih značilnic, saj bi v nasprotnem primeru nove značilnice najprej odstranili. Množico znanih vnosov $\left\{\boldsymbol{\chi}^{(i)}_{j=1:m^{(i)}}\right\}$ za posamezen w_i klestimo z uniformnim vzorčenjem.

3.4 Sledilnik

V tem podpoglavju predstavimo delovanje sledilnika. Najprej definiramo stanje sledilnika. V Podpoglavju 3.4.1 predstavimo korake sledenja z detekcijo in v Podpoglavju 3.4.2 sledenje z optičnim tokom Lucas-Kanade. Nato v Podpoglavju 3.4.3 definiramo združevanje ocenjenih preslikav in posodabljanje stanja sledilnika.

Stanje sledilnika v času t predstavimo kot $S_t(M, \Psi, \boldsymbol{F}, \nu, \boldsymbol{A}, \rho_d, \boldsymbol{\rho}_t, \kappa)$, kjer so:

- M - vizualni model,
- Ψ - množica značilnih točk za sledenje po metodi LK,
- $\boldsymbol{F}$ - globalna predloga oblike,
- ν - dinamični model,
- $\boldsymbol{A}$ - afina preslikava,
- ρ_d - število uspešnih in neuspešnih detekcij,
- $\boldsymbol{\rho}_t$ - število uspešnih in neuspešnih sledenj,
- κ - binarni atribut, ki označuje stabilnost stanja.

3.4.1 Sledenje z detekcijo

V podpoglavju predstavimo način sledenja z detekcijo, kjer uporabljamo vizualni model M, dinamični model NCV, posplošeno Houghovo transformacijo in afino preslikavo. Metoda močno zavisi od stabilnosti sledilnika $\boldsymbol{\kappa}_{t-1}$, ki ga določimo glede na število uspešnih sledenj (3.9, 3.13), zato delovanje metode najprej predstavimo v stabilnem in nato v nestabilnem stanju.

V stabilnem stanju detekcija v času t poteka sledeče:

1. napoved položaja tarče s pomočjo dinamičnega modela,

2. določitev iskalne regije (ang. *region of interest*, ROI) na podlagi preslikave $\boldsymbol{A}_{t-1}$ in skalirnega faktorja λ_s,

3. detekcija značilnic θ_t znotraj ROI in primerjava z deskriptorji modela M_{t-1},

4. glasovanje prepoznanih značilnic v Houghovem prostoru,

5. s pomočjo maksimumov Houghovega prostora določimo pare točk za naslednji korak,

6. z metodo MLESAC [28] izbranim parom točk poiščemo afino preslikavo $\boldsymbol{A}_d$ in posodobimo $\rho_d^{(t)}$ ter $\boldsymbol{\kappa}_t$.

V nadaljevanju podrobneje opišemo posamezne korake. V koraku 4 značilnico θ_k prepoznamo, če $min(d(\boldsymbol{\eta}_k, \boldsymbol{\eta}_{i=1:n^{(t)}})) < \lambda_{sim}$. Vsaki prepoznani značilnici nato pripišemo vse vektorje $\boldsymbol{\chi}_j^{(i)} = [x, y, x_c, y_c]^\intercal$ ustreznega vnosa w_i in po enačbi

$$\boldsymbol{C}^{(k,j)} = \begin{bmatrix} c_x \\ c_y \end{bmatrix} = r_k \begin{bmatrix} \cos\Phi_k & -\sin\Phi_k \\ \sin\Phi_k & \cos\Phi_k \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix}^{(j)} + \begin{bmatrix} x_k \\ y_k \end{bmatrix} \tag{3.6}$$

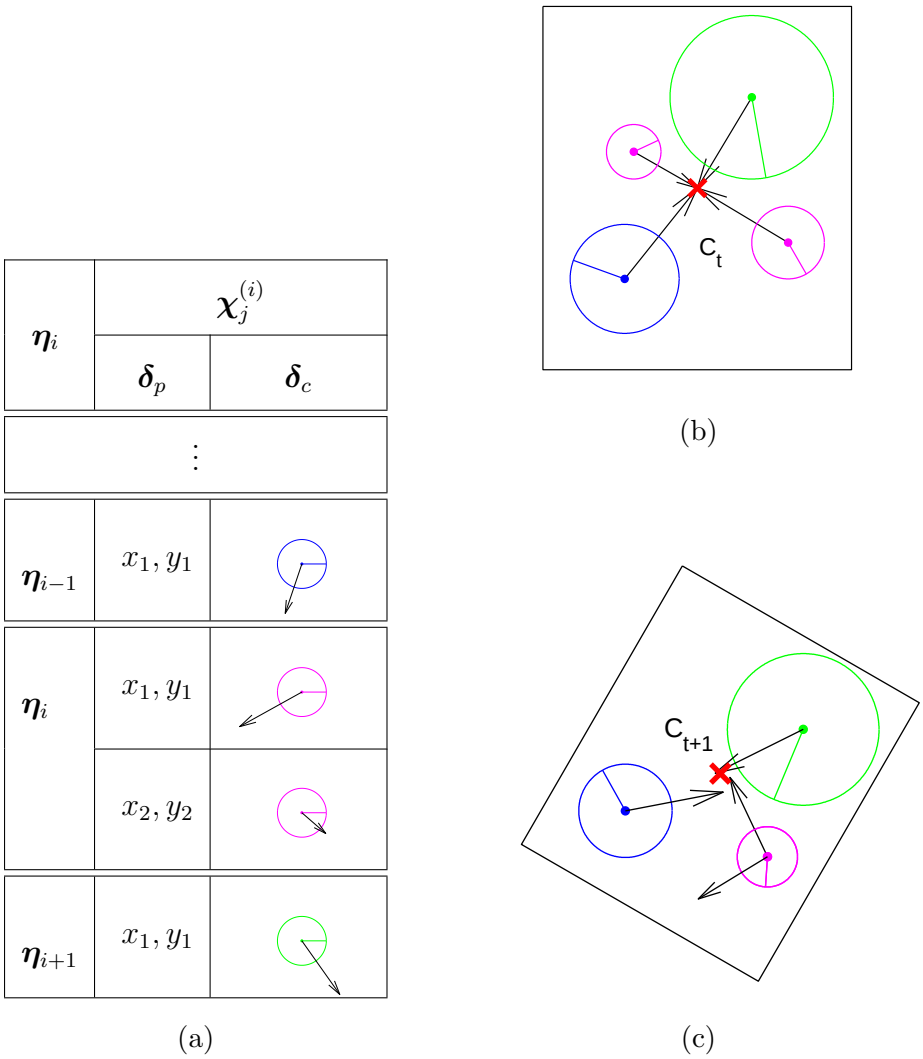
za vsakega izračunamo pričakovan položaj centra objekta $\boldsymbol{C}^{(k,j)}$. S tem za vsako θ_k definiramo matriko $\boldsymbol{V}_k$, kot prikazuje enačba

$$\boldsymbol{V}_k = \begin{bmatrix} x_k & y_k & \boldsymbol{\delta}_p^{(1)} & c_x^{(k,1)} & c_y^{(k,1)} \\ x_k & y_k & \boldsymbol{\delta}_p^{(2)} & c_x^{(k,2)} & c_y^{(k,2)} \\ \vdots & & & & \\ x_k & y_k & \boldsymbol{\delta}_p^{(m)} & c_x^{(k,m)} & c_y^{(k,m)} \end{bmatrix}. \tag{3.7}$$

Vrednosti $\boldsymbol{C}^{(k,j)}$ kvantificiramo kot $\left\lfloor \boldsymbol{C}^{(k,j)}/\delta_{cell} \right\rfloor$ in povečamo vrednost ustreznim celicam Houghovega prostora. Postopek prikazuje Slika 3.3.

Glasovalno shemo zgladimo z Gaussovim jedrom velikosti σ in zavržemo celice, ki niso lokalni maksimumi. V nadaljevanju se omejimo na preiskovanje največ petih potencialnih centrov znotraj ROI z največjo vrednostjo.

V koraku 5 za vsako morebitno središče $\boldsymbol{C}_p = [x_p, y_p]^\intercal$ izberemo pare točk, na podlagi katerih bo ocenjena $\boldsymbol{A}_t$. Množico parov označimo z Γ_p. Najprej izračunamo standardni odklon $\boldsymbol{\sigma}_c$ oglišč preslikave $\boldsymbol{A}_{t-1}$ v x in y smeri. Za vsako značilnico na podlagi $\boldsymbol{V}_k$ množici Γ_p dodamo pare točk $(\boldsymbol{\delta}_p^{(j)}, \boldsymbol{\delta}_k = [x_k, y_k]^\intercal)$, za katere velja $(c_x^{(j)} - x_p < \sigma_x/3) \wedge (c_y^{(j)} - y_p < \sigma_y/3)$.



Slika 3.3: Postopek pripisovanja vektorjev $\boldsymbol{\delta}_c$ značilnicam ob detekciji. (a) Model M_t. (b) Objekt v $t-1$. (c) Značilnicam pripišemo vse vektorje ustreznega deskriptorja modela, ti pa nato predvidevajo središče objekta.

V koraku 6 poskušamo za vsako množico Γ_p z metodo MLESAC [28] prilagoditi afino preslikavo $\boldsymbol{A}_p$. Znotraj iteracij metode vsako generirano preslikavo $\boldsymbol{A}_k$ primerjamo z $\boldsymbol{A}_{t-1}$ s pomočjo baznih vektorjev, ki jih preslikavi definirata (glej

Sliko 3.1). Zavržemo preslikave, za katere ne velja pogoj

$$\frac{\left\|\boldsymbol{i}_b^{(k)} - \boldsymbol{i}_b^{(t-1)}\right\|}{\left\|\boldsymbol{i}_b^{(t-1)}\right\|} < \lambda_{smax} \quad \wedge \quad \frac{\|\boldsymbol{j}_b^{(k)} - \boldsymbol{j}_b^{(t-1)}\|}{\|\boldsymbol{j}_b^{(t-1)}\|} < \lambda_{smax}$$
$$\wedge \quad |\theta_i^{(k)} - \theta_i^{(t-1)}| < \lambda_{rmax} \quad \wedge \quad \theta_{ij}^{(k)} < \lambda_{skew}. \tag{3.8}$$

S tem definiramo implicitni dinamični model za velikost in rotacijo tarče.

Rezultat detekcije $\boldsymbol{A}_d$ je $\boldsymbol{A}_p$, ki ima znotraj Γ_p največjo podporo. Če za noben p ni bilo mogoče najti preslikave, za katero velja (3.8), potem $\rho_d^{(t)} = 0$. V nasprotnem primeru velja $\rho_d^{(t)} = \rho_d^{(t-1)} + 1$.

V nestabilnem stanju namesto koraka 2 in 3 ROI določimo kot celotno slikovno polje. Prav tako v koraku 5 ne moremo izračunati $\boldsymbol{\sigma}_c$, ki določa prag za dodajanje parov točk množici Γ_p, na podlagi $\boldsymbol{A}_{t-1}$. $\boldsymbol{\sigma}_c$ zato ocenimo na podlagi preslikave, ki jo dobimo po metodi najmanjših kvadratov iz značilnic, ki so glasovale v celico trenutno preiskovanega morebitnega centra.

V koraku 6 generiranih $\boldsymbol{A}_k$ ne moremo vrednotiti glede na $\boldsymbol{A}_{t-1}$, če objekta v $t-1$ nismo detektirali. Če smo ga detektirali (stanje je lahko nestabilno kljub detekciji v $t-1$), pa $\boldsymbol{A}_k$ vseeno omejujemo glede na (3.8), s čimer preverjamo doslednost morebitne ponovne detekcije.

Če smo tarčo uspešno detektirali, potem $\rho_d^{(t)} = \rho_d^{(t-1)} + 1$. V nasprotnem primeru velja $\rho_d^{(t)} = 0$. V nestabilnem stanju se z uspešno detekcijo lahko zgodi, da stanje preide v stabilno. Pogoj določimo z enačbo

$$\rho_d^{(t)} > 2 \Rightarrow \kappa_t = 1. \tag{3.9}$$

Do ponovnega prepoznavanja tarče in ponovne vzpostavitve stabilnosti torej pride takrat, ko imamo v treh zaporednih slikah dosledno oceno preslikave na podlagi preiskovanja petih najverjetnejših položajev v celotnem slikovnem okvirju.

3.4.2 Kratkoročno sledenje z optičnim tokom

V času t za vsako točko $\boldsymbol{\psi} \in \Psi_{t-1}$ ocenimo optični tok po metodi LK, ki vključuje širjenje toka skozi Gaussovo piramido. Za vsako točko za enačbo optičnega toka (2.7) izračunamo 2×2 normalno matriko [6]. Za le-to se nato izračuna najmanjšo lastno vrednost in jo deli s številom pikslov v regiji. Če je vrednost manjša od λ, se za točko optičnega toka sploh ne oceni.

Točke, za katere smo ocenili optični tok, označimo s $\boldsymbol{\psi}_i$ in njihove koordinate v času t s $\boldsymbol{\psi}_j$. Za vsako $\boldsymbol{\psi}_i$ izračunamo $\boldsymbol{\delta}_i = \boldsymbol{A}_{t-1}^{-1}\boldsymbol{\psi}_i$. Na podlagi parov $(\boldsymbol{\delta}_i, \boldsymbol{\psi}_j)$ preslikavo $\boldsymbol{A}_s^{(t)}$ ocenimo z metodo MLESAC [28] enako kot v stabilnem stanju sledenja z detekcijo (glej Podpoglavje 3.4.1), kjer postavimo pogoj (3.8).

Ocenimo tudi verjetnost najdene preslikave $p(\boldsymbol{A}_s^{(t)}) = min(1, {}^{n_{in}}/_{\lambda_o n})$, kjer je n_{in} število točk, ki preslikavo podpirajo, in n število $\boldsymbol{\psi}_i$. Zaradi lastnosti metode MLESAC pričakujemo robustnost tudi v prisotnosti velikega šuma. Oceni preslikave tako zaupamo glede na pričakovan odstotek osamelcev (ang. *outliers*) λ_o. Če preslikave ne najdemo, je $p(\boldsymbol{A}_s^{(t)}) = 0$.

3.4.3 Posodabljanje stanja sledilnika

Spomnimo, da stanje sledilnika definiramo kot $S_t(M, \Psi, \boldsymbol{F}, \nu, \boldsymbol{A}, \rho_d, \boldsymbol{\rho}_t, \kappa)$. Posodabljanje preslikave $\boldsymbol{A}$ določimo kot

$$\boldsymbol{A} = \begin{cases} w_d\boldsymbol{A}_d + w_s\boldsymbol{A}_s & \text{, če } \rho_d^{(t-1)} > 0 \wedge \rho_d > 0 \wedge p(\boldsymbol{A}_s) > .5, \\ \boldsymbol{A}_d & \text{, če } \rho_d^{(t-1)} = 0 \wedge \rho_d = 1, \\ \boldsymbol{A}_s & \text{, če } \rho_d = 0 \wedge p(\boldsymbol{A}_s) > .5, \\ \boldsymbol{A}_p & \text{drugače} \end{cases}, \tag{3.10}$$

kjer je $[w_d, w_t]^\intercal = [1, p(\boldsymbol{A}_s)]^\intercal / (1 + \boldsymbol{A}_s)$, $\boldsymbol{A}_p$ pa določen kot

$$\boldsymbol{A}_p = \begin{bmatrix} a_{11}^{(t-1)} & a_{12}^{(t-1)} & x_k \\ a_{21}^{(t-1)} & a_{22}^{(t-1)} & y_k \end{bmatrix}, \tag{3.11}$$

kjer sta x_k in y_k s Kalmanovim filtrom posodobljena x in y dinamičnega modela.

V primeru uspešnega sledenja tako z detekcijo kot s pomočjo optičnega toka ocenjeni preslikavi združimo z uteževanjem. V primeru detekcije po v času $t-1$ neuspeli detekciji, zaupamo zgolj $\boldsymbol{A}_d$. V primeru neuspele detekcije se zanašamo na $\boldsymbol{A}_s$.

Posodabljanje števila uspešnih in neuspešnih sledenj $\boldsymbol{\rho}_t$ in indikatorja stabilnosti stanja κ določata enačbi

$$\boldsymbol{\rho}_t = \begin{cases} \boldsymbol{\rho}_t^{(t-1)} + \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal & \text{, če } \boldsymbol{A}_t \neq \boldsymbol{A}_{t-1} \\ \begin{bmatrix} 0 & \boldsymbol{\rho}_t^{(t-1)}(2) + 1 \end{bmatrix}^\intercal & \text{drugače} \end{cases}, \tag{3.12}$$

$$\kappa = \kappa_{t-1} \wedge \boldsymbol{\rho}_t(2) < 3 \vee \rho_d > 2. \tag{3.13}$$

Postopek posodabljanja števila uspešnih detekcij ρ_d smo podali v Podpoglavju 3.4.1.

M, Ψ,$\boldsymbol{F}$ in ν nato posodabljamo glede na posodobljena $\boldsymbol{A}_t$ in κ_t. Dinamični model ν določimo kot

$$\nu = \begin{cases} K(\nu_{t-1}, \boldsymbol{C}) & \text{, če } \kappa \\ \nu_0 & \text{, če } \neg\kappa \wedge \rho_d = 1, \end{cases} , \tag{3.14}$$

kjer so K Kalmanov filter, $\boldsymbol{C}$ središče objekta, kot ga določa $\boldsymbol{A}$ in ν_0 ponastavljen dinamični model glede na $\boldsymbol{C}$.

Posodabljanje globalne predloge oblike $\boldsymbol{F}$, dodajanje značilnic modelu in značilnih točk množici Ψ tvorijo povratno zanko. V t za vsako prepoznano značilnico in vsako značilno točko, za katero smo ocenili optični tok (tudi, če niso sodelovale pri oceni $\boldsymbol{A}_t$) preverimo, če napaka $\epsilon(\boldsymbol{A}_t) < \lambda_i$ (3.2), kjer je λ_i večja od napake, ki jo dovolimo pri ocenjevanju $\boldsymbol{A}_t$. $\boldsymbol{F}_t$ nato določimo v štirih korakih:

1. Določimo $\boldsymbol{T}_d = \boldsymbol{0}^{h\times w}$, kjer sta h in w višina in širina slike. Vsaki značilnici $\theta_k = \{x_k, y_k, r_k, \phi_k, \boldsymbol{\eta}_k\}$, katere ϵ ustreza pogoju, pripišemo Gaussovo jedro $K(\theta_k)$ velikosti $2r_k$ in $\sigma = r_k/2$ in velja $max(K) = 1$. Določimo $\boldsymbol{T}(x_k+i, y_k+j) = min(1, K(r_k+i, r_k+j) + \boldsymbol{T}(x_k+i, y_k+j))$, za vsak $i, j \in [-rk, rk]$.

2. Določimo $\boldsymbol{T}_f = \boldsymbol{0}^{h\times w}$ in za vsako značilno točko $\boldsymbol{\psi}_k = [x, y]^\intercal$ posodobimo $\boldsymbol{T}_f$ enako kot to storimo za $\boldsymbol{T}_d$ v koraku 1 s to razliko, da je r_k konstanta.

3. Izračunamo $\boldsymbol{F}'_t = min(1, \boldsymbol{F}_{t-1} + \lambda_1\beta\boldsymbol{T}_d + \lambda_2\boldsymbol{T}_f \odot (\boldsymbol{F}_{t-1} + \beta\boldsymbol{T}_d))$, kjer $\lambda_1 \propto \delta_d$, $\lambda_1 \in [.3, 1]$ in $\lambda_2 \propto \boldsymbol{\delta}_t(1)$, $\lambda_2 \in [.3, 1]$.

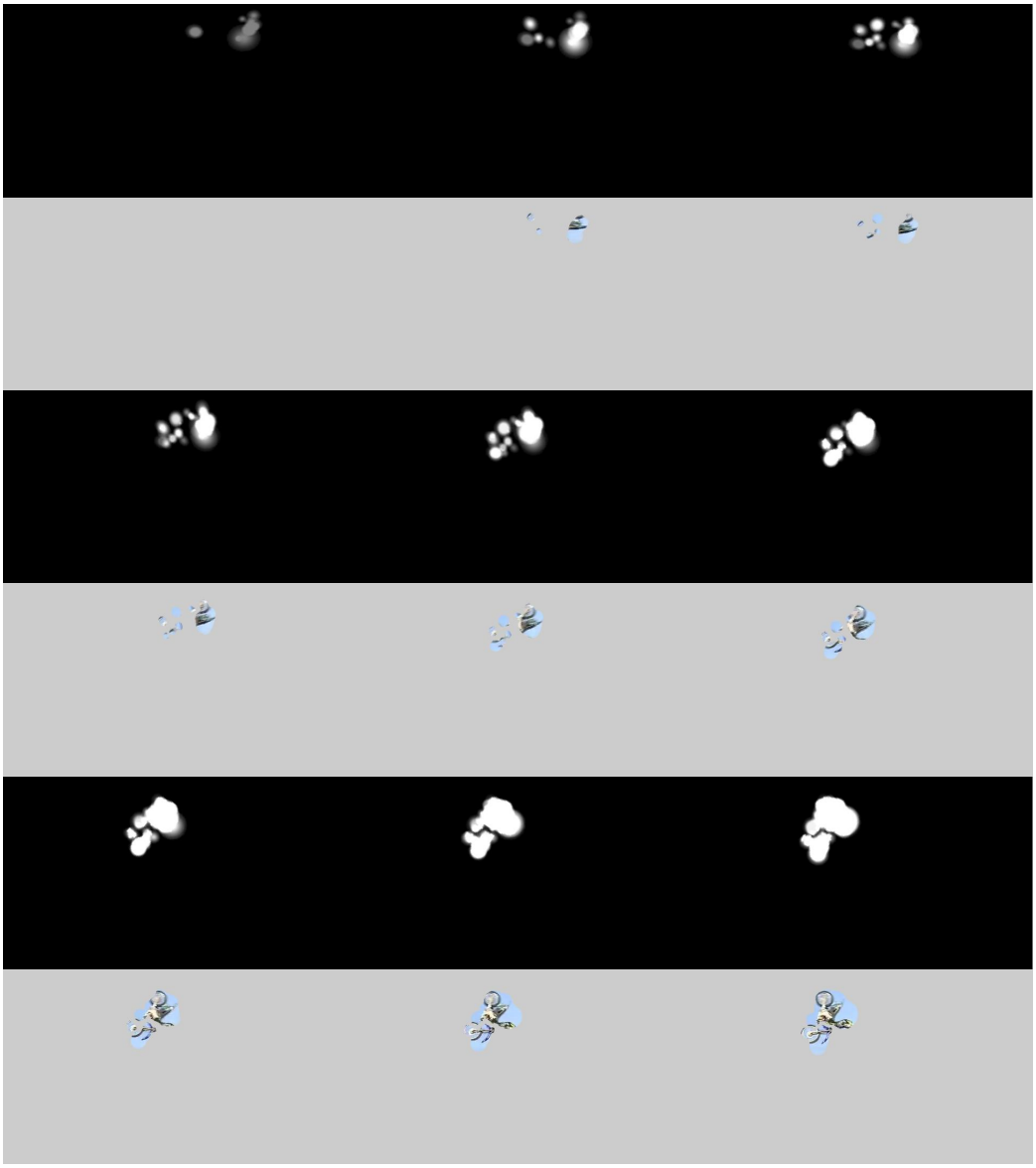
4. Izračunamo $\boldsymbol{F}_t = \alpha\boldsymbol{F}'$.

Povedano z besedami, na mestih, kjer se nahajajo točke, ki podpirajo preslikavo $\boldsymbol{A}_t$, $\boldsymbol{F}$ povečamo vrednost. Ko je bilo sledenje neuspešno v 5 zaporednih časovnih intervalih, je $\boldsymbol{F}_t = \boldsymbol{0}$. V koraku inicializacije $\boldsymbol{F}_0$ dobimo z algoritmom GrabCut [25], ki regijo objekta segmentira na podlagi primerjave barvne informacije regije z njeno okolico. Primer rezultatov posodabljanja po začetni segmentaciji prikazuje Slika 3.4 in po ponovni detekciji Slika 3.5.

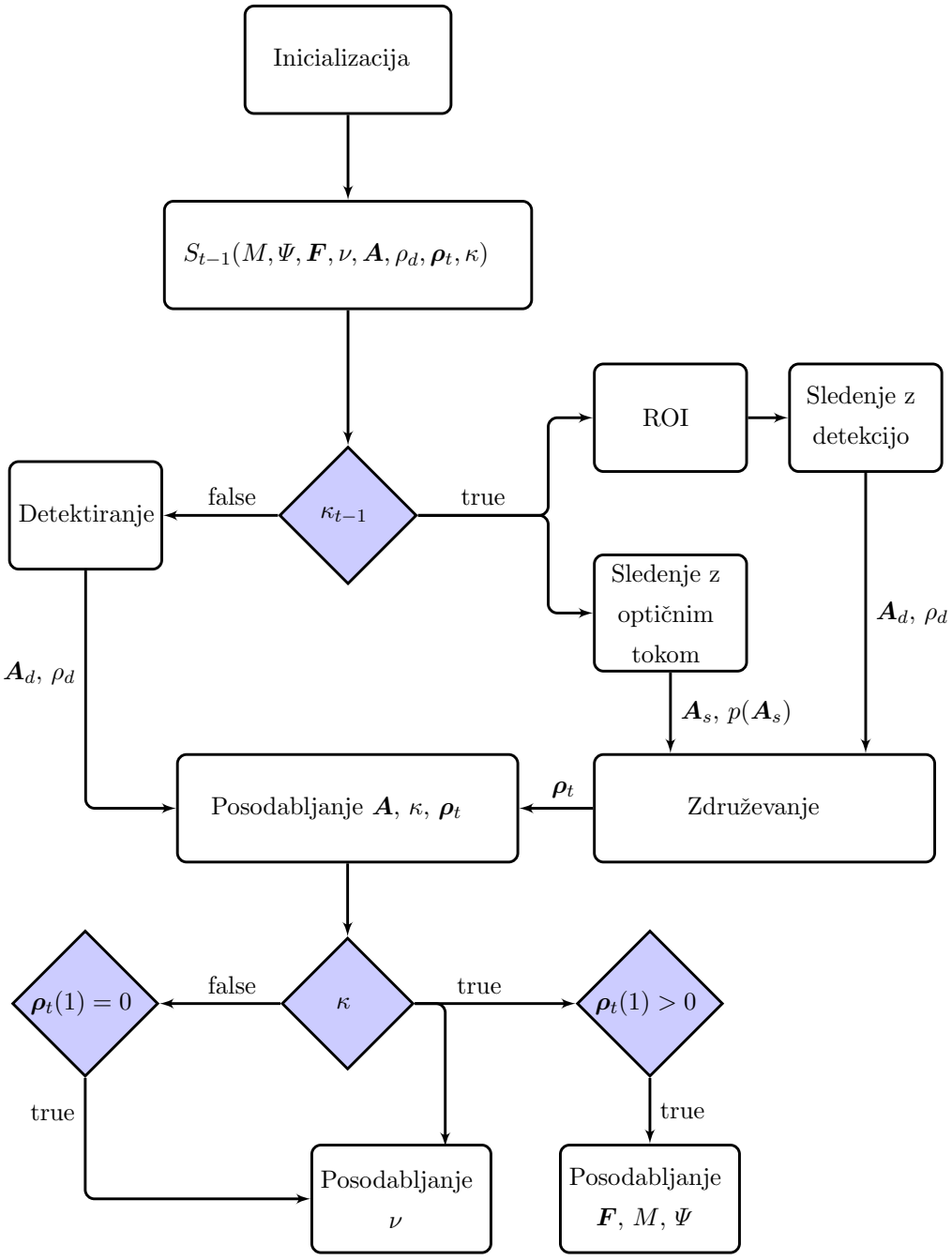
$\boldsymbol{F}'$ uporabimo za filtriranje značilnih točk in značilnic, ki smo ga podali v Podpoglavju 3.3. Spomnimo, da dodajamo točke na mestih, kjer je vrednost predloge $\boldsymbol{F}'$ dovolj velika. Filtriramo samo ob uspešnem sledenju v stabilnem stanju.

Slika 3.4: Spreminjanje globalne predloge oblike na podlagi doslednih značilnih točk po začetni segmentaciji in pripadajoče maskirane slike. $\Delta_t = 2$.

Posodabljanja modela smo podali v Podpoglavju 3.3. Ob neuspešnem sledenju je $\Psi = \varnothing$. Celoten algoritem ponazarja Slika 3.6.



Slika 3.5: Spreminjanje globalne predloge oblike na podlagi doslednih značilnih točk in značilnic po ponovni detekciji. $\Delta_t = 2$.



Slika 3.6: Shema delovanja sledilnika.

Poglavje 4

Testiranje in analiza rezultatov

V poglavju predstavimo način testiranja sledilnika, rezultate analiziramo in jih primerjamo s sorodnimi metodami. Sledilnik smo testirali na javno dostopnih zbirkah, in sicer na zbirkah LTDT 2014 (ang. *Long-Term Dectection and Tracking*) [7] ter VOT Challenge 2014 (ang. *Visual Object Tracking*) [24]. Parametri sledilnika so bili na vseh sekvencah enaki, predstavimo jih v Podpoglavju 4.1 skupaj z načinom izračuna izbranih mer primerjanja. V Podpoglavju 4.2 predstavimo testiranje na sekvencah LTDT in v Podpoglavju 4.3 na sekvencah VOT.

Sledilnik je implementiran v okolju *Matlab* s pomočjo knjižnice *OpenCV* in v povprečju deluje s frekvenco 7.1 slik na sekundo (fps). Pri računanju frekvence smo sekvence, kjer sledilnik odpove že na začetku, izpustili, saj je tam hitrost velika zgolj zaradi majhnega modela in tako nerealna.

4.1 Parametri sledilnika in računanje mer

Pri testiranju uporabljene parametre sledilnika predstavlja Tabela 4.1 v vrstnem redu, kot so navedeni skozi poglavja.

parameter	**vrednost**	**parameter**	**vrednost**
λ_c	.6	δ_{cell}	5
λ_d	.9	λ_{smax}	.2
λ_a	.7	λ_{rmax}	30°
λ_k	.1	λ_{skew}	10°
n_{max}	500	λ_0	.6°
m_{max}	20	β	.6
λ_s	3	α	.95
λ_{sim}	.3		

Tabela 4.1: Parametri sledilnika, uporabljeni pri testiranju.

Za vrednotenje sledilnika smo na vseh sekvencah uporabili naslednje mere: točnost A_{cc} (ang. *accuracy*), natančnost P (ang. *precision*), občutljivost R (*recall*), odstotek sledenj S_r (*ang. sucess rate*) in mero F, kot kažejo enačbe

$$A_{cc} = \frac{O \cap O_g t}{O \cup O_g t} \quad , \quad P = \frac{tp}{tp + fp} \quad , \quad R = \frac{tp}{fn},$$
$$S_r = \frac{n_t}{n} \quad , \quad F = \frac{2 * P * R}{P + R}, \tag{4.1}$$

kjer sta O ter O_{gt} območji, ki ga tarči pripišeta sledilnik in človek, n_t število okvirjev, v katerih je sledilnik ocenil položaj, in n število vseh okvirjev. Število pravilnih sledenj tp smo računali glede na A_{cc}, in sicer smo za pravilno prepoznali sledenje, katerege $A_{cc} > 0$. Število napačnih sledenj fp je enako številu okvirjev, kjer je sledilnik menil, da je tarčo uspešno našel, a je $A_{cc} = 0$. fn je število okvirjev, kjer sledilnik tarče kljub prisotnosti ni zaznal.

4.2 Sekvence LTDT 2014

LTDT [7] podaja 6 sekvenc, ki so namenjene testiranju in primerjavi dolgoročnih sledilnikov in zato daljše (od 2665 pa tudi do 29697 zaporednih slik), ter vsebujejo situcije, ki predstavljajo največje izzive dolgoročnega sledenja (predvsem zakritost in daljša odsotnost tarče). Tarče same po sebi niso zahtevne, večinoma so toge. Na sekvencah preverjamo zadane cilje sledilnika: 1) prilagajanje vizualnega

modela spremembam videza tarče, 2) prepoznavanje odsotnosti tarče oz. odpovedi sledenja, 3) ponovna detekcija tarče po njenem ponovnem vstopu. Za zadnjo točko je najpomembnejše to, da ne pride do napačne ponovne detekcije, kjer bi model posodabljali na podlagi drugega realnega objekta, če pa do tega pride, pa bi želeli, da napačna tarča ne prenasiti modela, da lahko kasneje pravo tarčo ponovno detektiramo.

Za primerjavo imamo na voljo rezultate dveh trenutno najnaprednejših (ang. *state-of-the-art*, SOTA) dolgoročnih sledilnikov: PREDATOR [16] in ALIEN [23], ki so podani z vertikalno orientiranimi pravokotniki. V ta namen regije našega sledilnika ustrezno prilagodimo. Rezultate prikazujemo samo za 5 sekvenc, saj na najdaljši že v samem začetku sledilnik odpove, ker nobena značilnica ne ustreza pogoju za dodajanje.

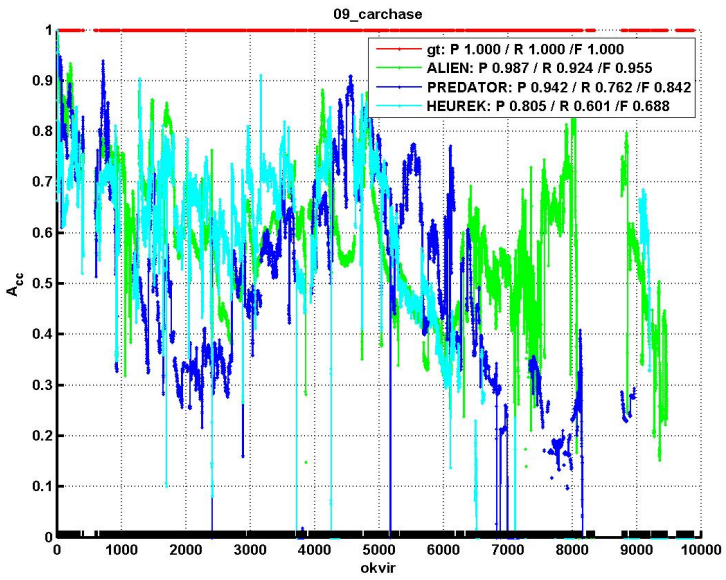
Sledilnik na tem mestu poimenujemo HEUREK. Imena ne poiskušamo prisiliti v kratico, temveč z njim namigujemo na potencial sledilnika, saj upamo, da v prihodnosti zraste v HEUREKO.

4.2.1 Rezultati

Za boljšo interpretacijo predstavimo grafično primerjavo časovne odvisnosti mere A_{cc}, ki jo tudi v obliki histograma.

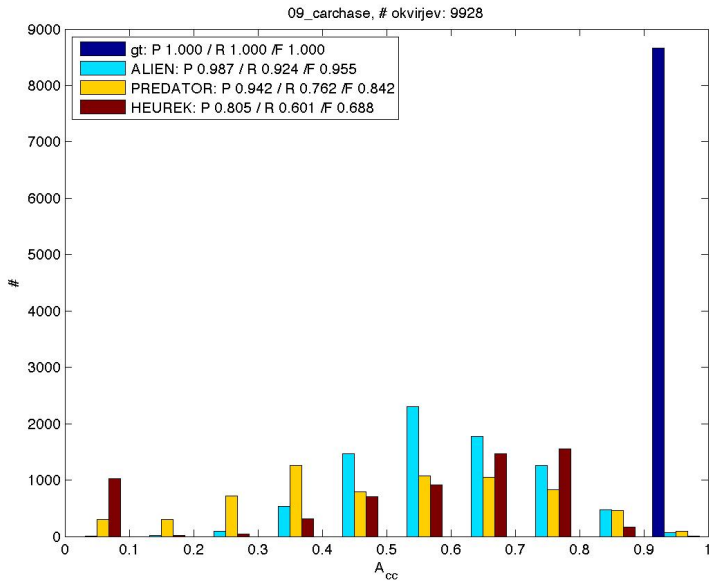
Sekvenca *09_carchase* (Slika 4.1) vsebuje slike helikopterskega posnetka sledenja avtomobilu na avtocesti. Problematična so zakritja, saj je nad avtocesto veliko nadvozov (takrat tarče ne vidimo), ter veliko podobnih objektov pri ponovni detekciji. Približno od 7000-tega okvirja (Slika 4.2) naprej je čez sliko prisoten grafični napis, kjer sama tipografija vsebuje pester nabor značilnic, ki jih nato sledilnik prepozna za tarčo. Kljub relativno dolgem napačnemu sledenju se proti koncu zgodi pravilna ponovna detekcija, kar kaže, da se model zmotni tarči ni preveč prilagodil. Glede na izbrane mere HEUREK zaostaja za sledilnikoma ALIEN in PREDATOR. Glede na distribucijo A_{cc} (Slika 4.3) lahko opazimo, da je za to kriva prav napačna detekcija. Vidimo, da število sledenih okvirjev v nasprotju z ostalima sledilnikoma narašča v odvisnosti od A_{cc} skoraj linearno, da pa je prisoten vrh pri najmanjši točnosti, ki nakazuje napačno detekcijo. Padec števila okvirjev pri največji natančnosti lahko razumemo kot posledico približka ocenjene regije v vertikalno orientiran pravokotnik.

Slika 4.1: Primeri okvirjev sekvence *09_carchase*.



Slika 4.2: Primerjava časovne odvisnosti mere A_{cc} sledilnikov na sekvenci *09_carchase*.

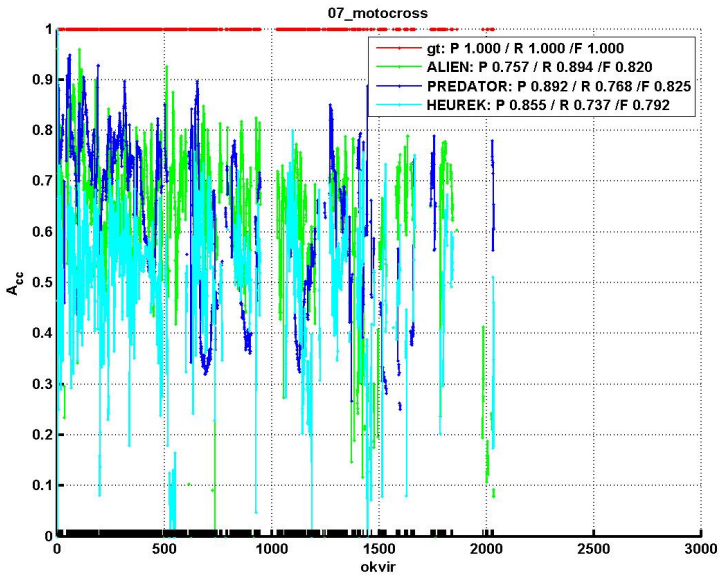
Sekvenca *07_motocross* (Slika 4.4) je za naš sledilnik zelo problematična ob inicializacija, kjer metoda GrabCut [25] ne ustvari dovolj velik predloge oblike, da



Slika 4.3: Primerjava histogramov mere A_{cc} sledilnikov na sekvenci *09_carchase*.

bi modelu ali Ψ sploh lahko dodali kakšno točko. Izziv predstavljata tudi netoga tarča, megljenje in rezanje posnetkov.

Slika 4.4: Primeri okvirjev sekvence *07_motocross*.

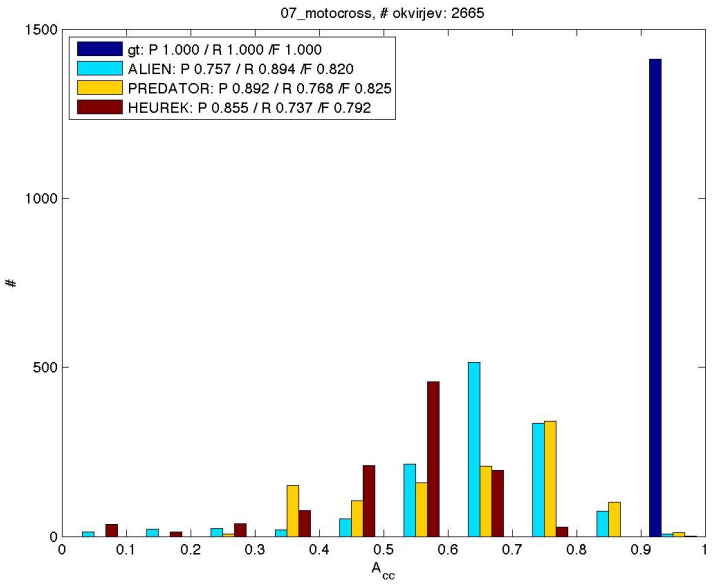


Slika 4.5: Primerjava časovne odvisnosti mere A_{cc} sledilnikov na sekvenci *07_motocross*.

Na sekvenci *08_volkswagen* (Slika 4.7) se sledi avtomobilu iz avtomobila za njim. Tudi ta sekvenca je za sledilnik problematična ob inicializaciji, saj je tarča majhna in relativno nestrukturirana. Glavni razlog za slabšo oceno je zopet napačna detekcija okrog okvirja št. 7000 in 8000, kot kaže Slika 4.8, kjer je točnost enaka 0. To kaže tudi vrh v histogramu na Sliki 4.9. Na koncu pride do ponovne pravilne detekcije, kar zopet kaže, da se sledilnik napačni tarči ni preveč prilagodil.

Kot pove ime, sekvenca *sitcom* vsebuje posnetke komične serije, kjer se sledi obrazu igralca. Sekvenca je v primerjavi z drugimi posebna najbolj po tem, da vsebuje reze (snemana je z več kamerami), zato ne moremo pričakovati kontinuitete gibanja. Na Sliki 4.10 se lepo vidi omejitev predpostavljanja linearnosti, saj naš sledilnik sledi obrazu in ne glavi. Za tovrstne posnetke bi bilo bolje uporabljati vnaprejšnje učenje.

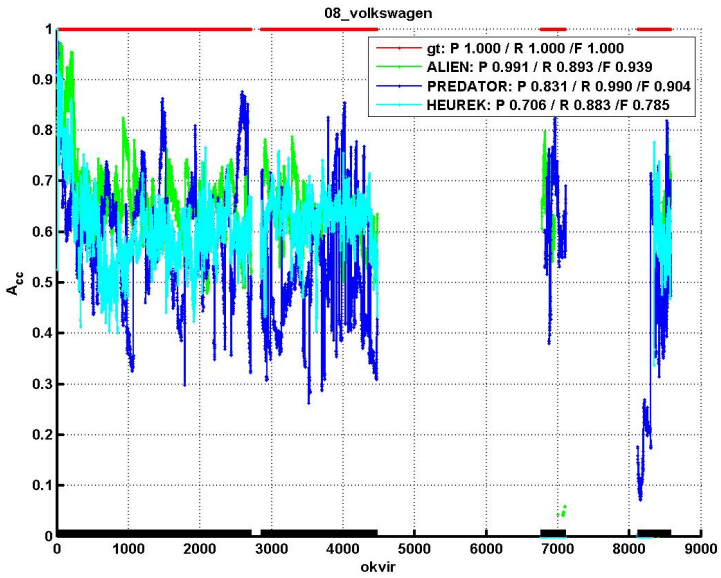
Ta in sekvenca *NissanSkylineChaseCropped* sta edini, kjer HEUREK glede na F-mero za las prekaša sledilnik ALIEN. To pa je verjetno varljivo. Podatkov za sledilnik PREDATOR pri teh sekvencah nimamo podanih, anotacije pa se ne zdijo človeške, zato sumimo, da tu zlati standard predstavljajo kar njegovi rezultati.



Slika 4.6: Primerjava histogramov mere A_{cc} sledilnikov na sekvenci *07_motocross*.

Slika 4.7: Primeri okvirjev sekvence *07_volkswagen*.

Rahlo večjo občutljivost si lahko razlagamo tudi s tem, da ALIEN za ponovno detekcijo uporablja naključno iskanje [23], medtem ko HEUREK prostor preiskuje



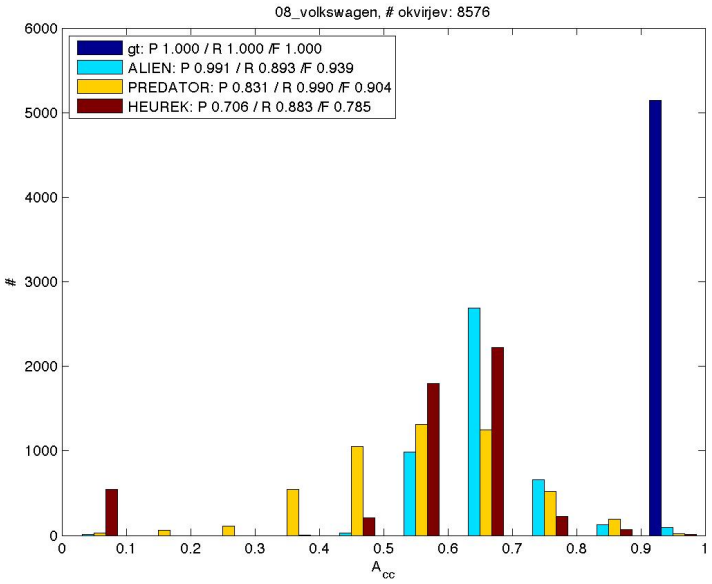
Slika 4.8: Primerjava časovne odvisnosti mere A_{cc} sledilnikov na sekvenci *08_volkswagen*.

s pomočjo GHT (glej Podpoglavje 3.4.1).

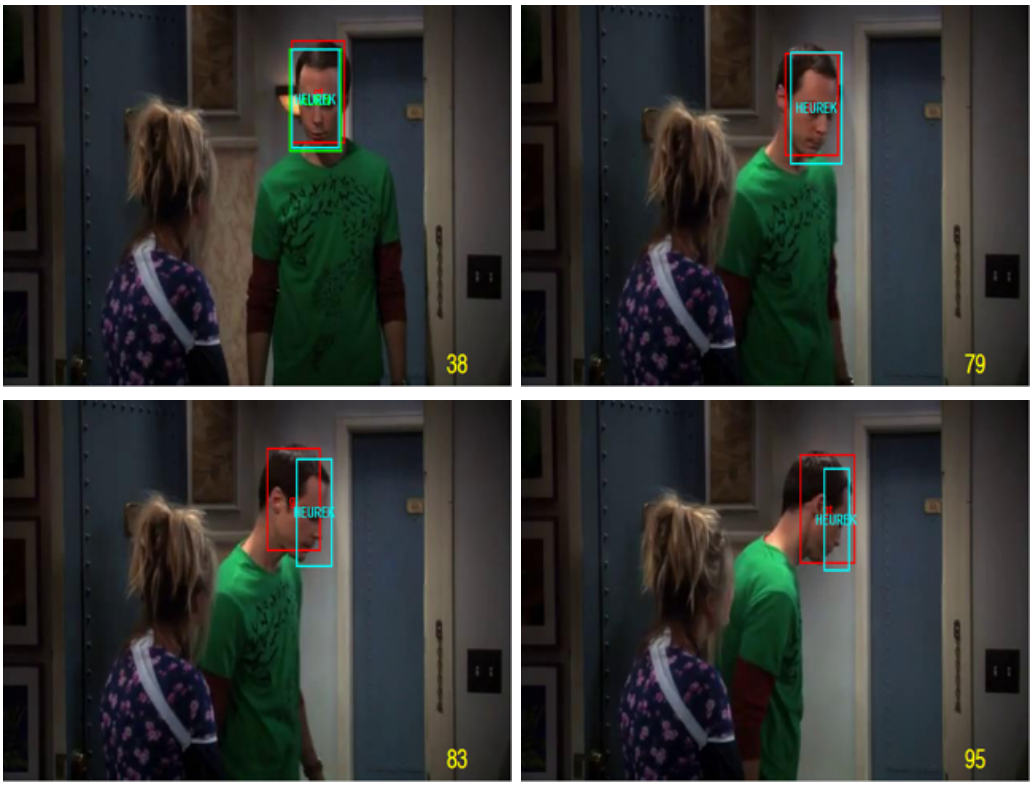
Podobni krivulji časovne odvisnosti A_{cc} na Sliki 4.11 lepo prikazujeta sorodnost delovanja našega sledilnika in sledilnika ALIEN.

Tudi v sekvenci *NissanSkylineChaseCropped* (Slika 4.13) se sledi avtomobilu. Iz grafa časovne odvisnosti A_{cc} (Slika 4.14) je zopet razvidno sorodno delovanje sledilnikov HEUREK in ALIEN, iz distribucije pa, da smo tudi tu za tarčo prepoznali nepravi objekt.

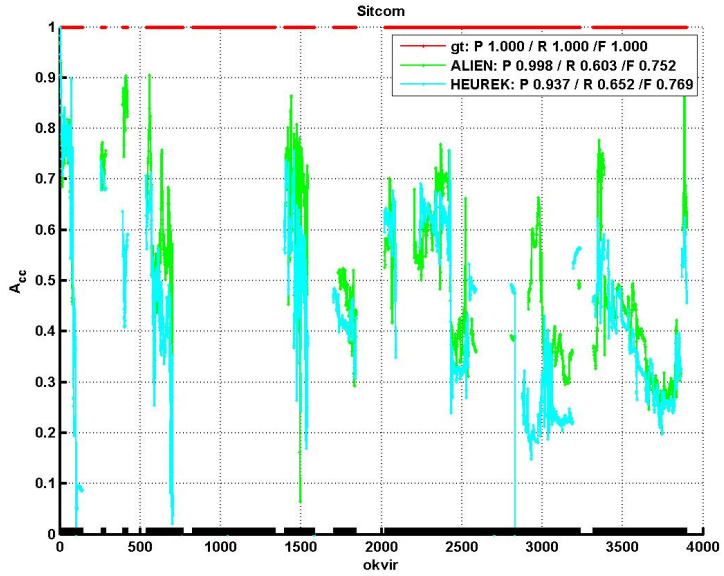
Rezultati kažejo, da smo zadovoljili začetne cilje. Sledilnik se prilagaja spreminjanju videza, prepozna odsotnost tarče in jo zna ponovno zaznati. Ugotavljamo, da je glavni razlog za slabše rezultate napačna ponovna detekcija. V nasprotju s primerjanima sledilnikoma HEUREK za klasificiranje ne uporablja negativnih primerov. Kljub napačnim detekcijam le-te ne prevladajo nad sledenjem. Zaradi problemo z inicializacijo bi morali na drugačen način nasloviti segmentiranje področja tarče.



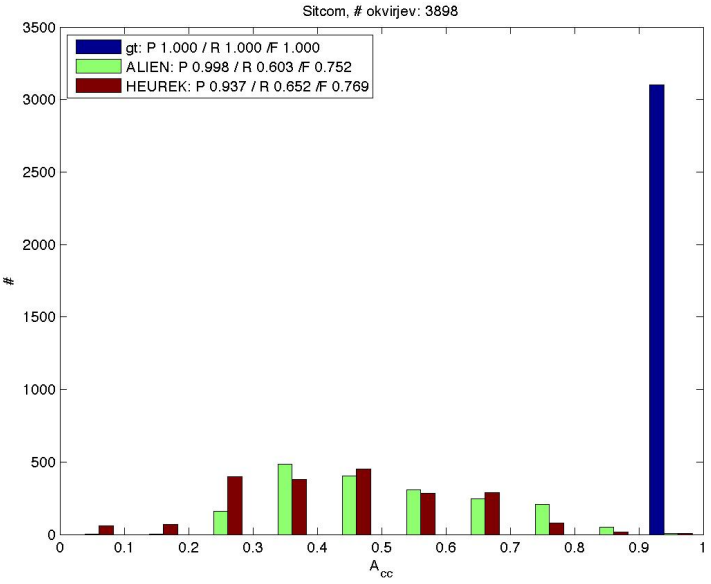
Slika 4.9: Primerjava histogramov mere A_{cc} sledilnikov na sekvenci *08_volkswagen*.



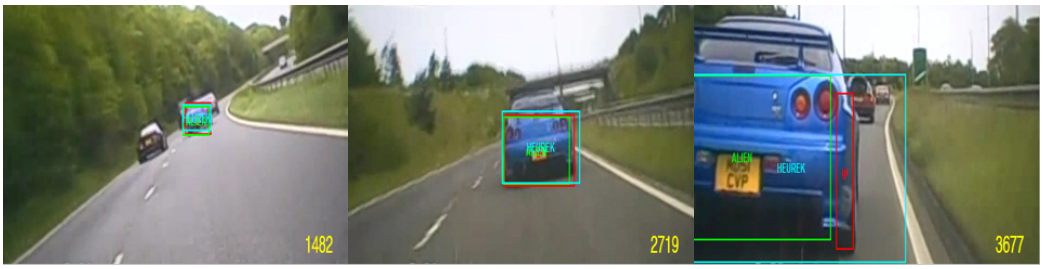
Slika 4.10: Obnašanje sledilnika na sekvenci *sitcom*. Sledilnik obraz razume kot ravnino.



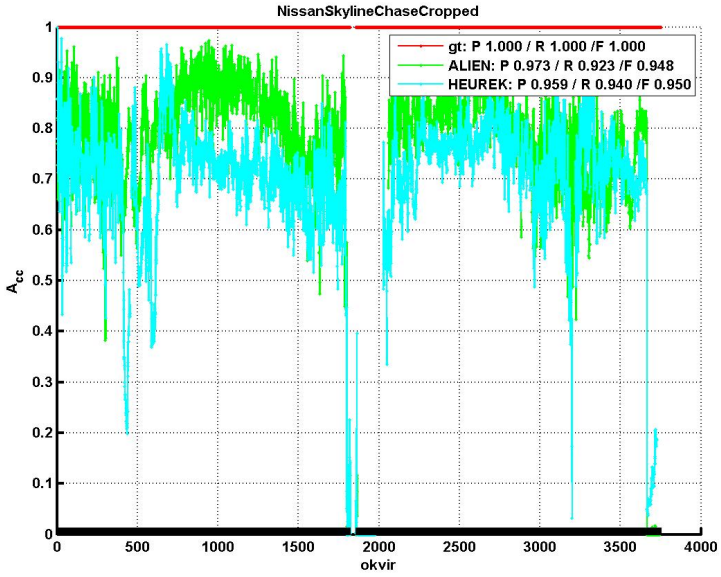
Slika 4.11: Primerjava časovne odvisnosti mere A_{cc} sledilnikov na sekvenci *sitcom*.



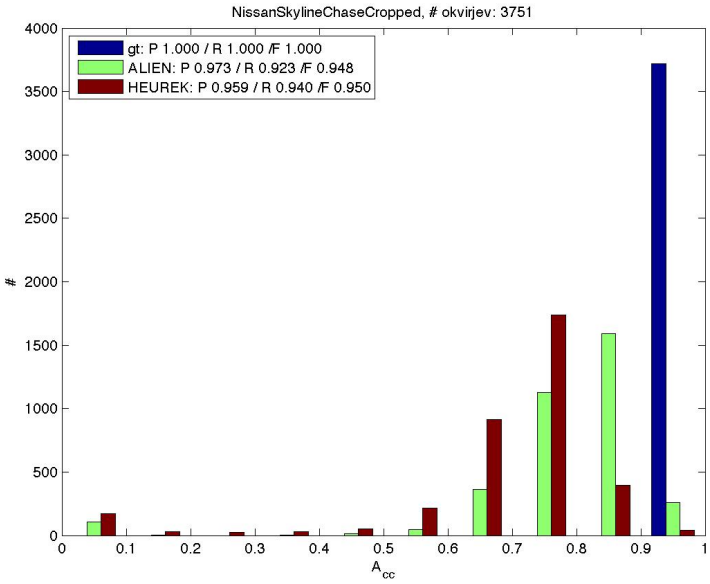
Slika 4.12: Primerjava histogramov mere A_{cc} sledilnikov na sekvenci *sitcom*.



Slika 4.13: Primeri okvirjev sekvence *NissanSkylineChaseCropped*. Na desni je videti vprašljivost zlatega standarda, označenega z rdečo.



Slika 4.14: Primerjava časovne odvisnosti mere A_{cc} sledilnikov na sekvenci *NissanSkylineChaseCropped*.



Slika 4.15: Primerjava histogramov mere A_{cc} sledilnikov na sekvenci *Nissan-SkylineChaseCropped*.

4.3 Sekvence VOT 2014

Sekvence VOT so namenjene vrednotenju kratkoročnih sledilnikov brez vnaprejšnjega učenja, kjer so lahko uspešni tudi sledilniki, ki vizualni model gradijo samo ob inicializaciji [18]. Sekvence poskušajo zajeti raznolike problematike, kot so hitri premiki kamere, sprememba osvetljave, velikosti in orientacije tarče, prisotnost šuma in megljenja. Anotirane so z orientiranimi pravokotniki in tako omogočajo boljše preverjanje kvalitete ocene afine preslikave. Tarče so za naš sledilnik bolj problematične: nekatere so premalo strukturirane, druge netoge. Sekvence povzema Slika 4.16.

Slika 4.16: Sekvence VOT [18].

4.3.1 Rezultati

Zaradi relativne kratkosti sekvenc smo si privoščili zagon sledilnika na vsaki sekvenci petkrat in izračunali povprečje rezultatov, ki so prikazani v Tabeli 4.2.

	točnost	natančnost	občutljivost	% sledenj	F-mera
ball	0.7302	1.0000	0.3618	0.3618	0.5271
bicycle	0.7008	0.9396	0.6489	0.6963	0.7657
car	0.8261	0.9871	0.9562	0.9689	0.9713
david	0.8093	1.0000	1.0000	1.0000	1.0000
drunk	0.6084	1.0000	0.9161	0.9161	0.9526
hand1	0.4723	0.5054	0.3984	0.8165	0.4305
hand2	0.7571	0.9285	0.1925	0.2632	0.2487
jogging	0.6877	0.9743	0.8660	0.8869	0.9162
motocross	0.8106	1.0000	0.8160	0.8160	0.8955
polarbear	0.6707	0.9981	0.8443	0.8454	0.9060
skating	0.5938	0.6427	0.3003	0.4331	0.4028
sphere	0.8751	1.0000	0.0190	0.0190	0.0367
sunshade	0.8285	1.0000	0.4234	0.4234	0.5942
surfing	0.6489	1.0000	1.0000	1.0000	1.0000
torus	0.6898	1.0000	0.1422	0.1422	0.2470
trellis	0.6930	0.8567	0.3461	0.4563	0.4546
tunnel	0.5039	1.0000	1.0000	1.0000	1.0000
woman	0.5093	0.9565	0.4366	0.4594	0.5645
basketball	-	-	-	-	-
bolt	-	-	-	-	-
diving	-	-	-	-	-
fernando	-	-	-	-	-
fish1	-	-	-	-	-
fish2	-	-	-	-	-
gymnastics	-	-	-	-	-

Tabela 4.2: Rezultati testiranja sledilnika na sekvencah VOT.

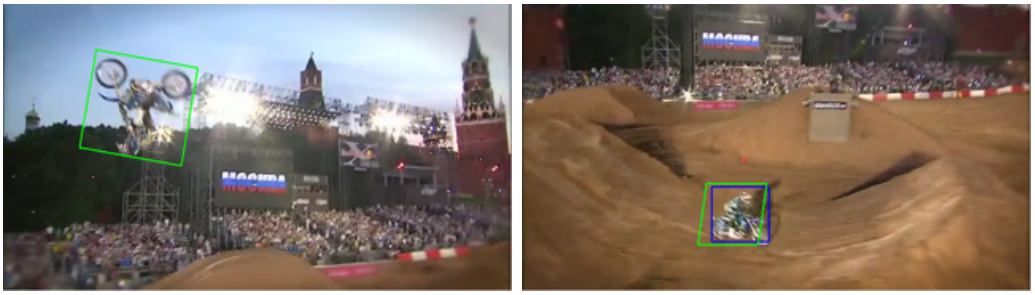
Sledilnik odpove pri sledenju netogih objektov in v trenutkih, ko je prisotno megljenje, saj tam ni moč oceniti optičnega toka, prav tako pa ne prepoznamo

značilnic. Problem bi lahko rešili, če bi pri problemih kratkoročnega sledenja dopustili dodajanje novih točk za sledenje tudi v prvem okvirju, v katerem sledenje odpove, in bi zaupali dinamičnemu modelu. Zaradi nagnjenosti k dolgoročnosti tega ne počnemo.

Za sekvence, kjer sledilnik popolnoma odpove, rezultati niso prikazani. Kljub temu podajamo nekaj tovrstnih primerov. Ena takšnih je sekvenca *sphere*, kjer je točnost sicer visoka, a jo sledilnik prepozna le v 4 okvirjih. Tu je problematično megljenje. Sekvenci *hand1* ter *hand2* sta problematični zaradi megljenja, netogosti tarče in slabe teksturiranosti.

Lastnosti sledilnika lepo kažeta sekvenci *motocross* in *ball*. Prvo so na izzivu VOT rangirali kot najbolj zahtevno glede na rezultate sodelujočih sledilnikov [18], HEUREK pa ima tu visoko tako točnost kot F-mero. Sekvenca naj bi bila zahtevna predvsem zaradi spreminjanja velikosti in rotacije, ki ga naš sledilnik upošteva. Nekaj okvirjev prikazuje Slika 4.17.

Sekvenca *ball* kaže omejitve afine preslikave (predpostavka planarnosti) in omejitev sledilnika pri celostnem dojemanju tarče. Žoga je seveda okrogla, ko pa se vrti izven ravnine slike, sledilnik upošteva le znan videz (glej Sliko 4.18). Da bi ji lahko uspešno sledil, se mora njenega videza učiti dalj časa.

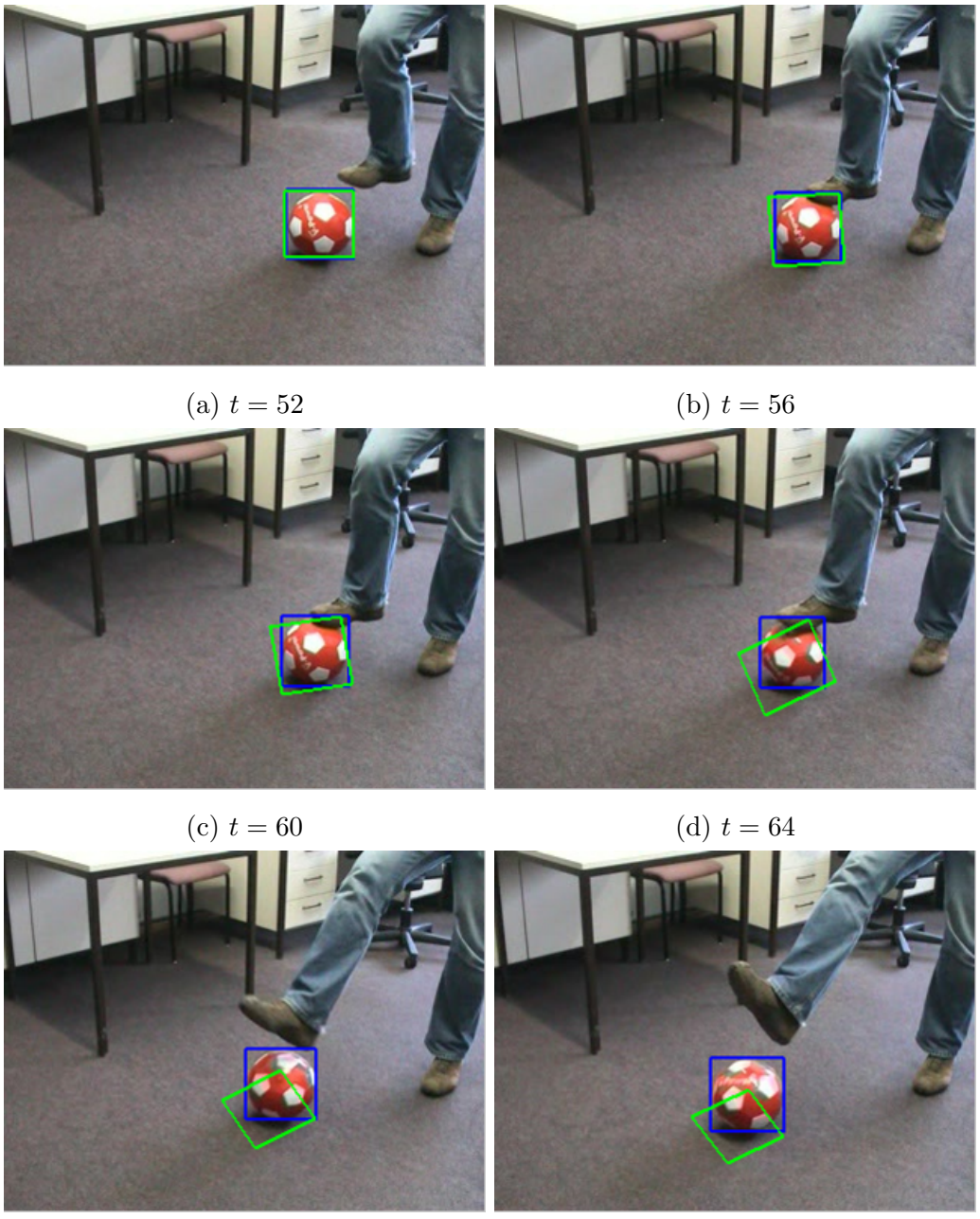


(a) $t = 1$, inicializacija (b) $t = 59$

(c) $t = 70$ (d) $t = 71$, odpoved sledenja

(e) $t = 84$, ponovna detekcija (f) $t = 160$

Slika 4.17: Obnašanje sledilnika na sekvenci *motocross*. Modra regija je ročna anotacija, zelena pa rezultat sledilnika.



(a) $t = 52$ (b) $t = 56$

(c) $t = 60$ (d) $t = 64$

(e) $t = 68$ (f) $t = 72$

Slika 4.18: Obnašanje sledilnika na sekvenci *ball*. Sledilnik poizkuša žogo razumeti kot ravnino.

Poglavje 5

Sklepne ugotovitve

Kljub zadanim ciljem razviti sledilnik pušča veliko odprtih možnosti za izboljšave. Iz rezultatov in empiričnega opazovanja lahko sklepamo na glavne pomanjkljivosti sledilnika: 1) predpostavljanje globalne planarnosti, 2) odvisnost od rezultatov metode GrabCut [25] ob inicializaciji, ko rezultat segmentacije ne zadosti pogoju za dodajanje značilnih točk, 3) večja verjetnost napačne ponovne detekcije od konkurenčnih metod 4) občutljivost na velike vizualne spremembe in 5) občasna občutljivost na zakrivanje.

Zadnji dve pomanjkljivosti sta povezani. Problem zakrivanja namreč rešujemo implicitno z globalno predlogo oblike, kjer za dodajanje novih značilnic postavimo velik prag in nizko mero zaupanja pri posodabljanju le-te. Na mestih, kjer nastopijo večje vizualne spremembe, v koraku detekcije in sledenja z optičnim tokom tako oblike ne afirmiramo in posledično točk iz področja ne prepoznavamo kot tarčo. Problem zakrivanja bi lahko rešili tudi drugače. Uporabili bi predlogo za okolico objekta in z optičnim tokom sledili točkam izven regije tarče, glede na predlogo oblike pa tudi tistim znotraj nje. Pri filtriranju značilnih točk bi tako upoštevali vizualni kontekst, ne da bi s tem zavrgli informacijo o obliki, ki se skriva za zakrivalnim elementom. Časovne zahtevnosti takšna metoda ne bi bistveno povečala.

Predpostavko planarnosti bi lahko reševali z metodami ocenjevanja nelinearnih transformacij, kjer bi v prvem koraku še vedno lahko uporabili posplošeno Houghovo transformacijo. Premisliti bi bilo potrebno o alternativni metodi segmentiranja tarče ob inicializaciji in preveriti, če je glede na nadaljnje posodabljanje

globalne predloge oblike začetna segmentacija sploh potrebna.

Za zmanjšanje števila napačnih ponovnih detekcij bi lahko uporabili metodo, kjer hranimo tudi značilnice ozadja in detektirane značilnice, ki so ozadju podobne, ne sodelujejo v nadaljnji oceni stanja tarče, tako kot to počne sledilnik ALIEN [23], s to razliko, da bi bil ta korak potreben le ob iskanju tarče. Problem bi lahko reševali tudi tako, da bi pri ponovni detekciji značilnice zavrgi na podlagi podpore, ki si jo je skozi čas pridobila v modelu, podobno kot počnemo pri klestenju modela.

Inherentna pomanjkljivost sledilnikov, ki model gradijo na podlagi značilnih točk, je neuspešnost sledenja neteksturiranih tarč. Problem bi lahko naslovili z uporabo množice modelov različnih vrst, predvsem takih, ki uporabljajo barvno informacijo.

Literatura

[1] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. Kaze features. In *Computer Vision – ECCV 2012*, volume 7577 of *Lecture Notes in Computer Science*, pages 214–227. Springer Berlin Heidelberg, 2012.

[2] B. Babenko, Ming-Hsuan Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1619–1632, Aug 2011.

[3] D. H. Ballard. Readings in computer vision: Issues, problems, principles, and paradigms. pages 714–725. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, pages 404–417, Berlin, Heidelberg, 2006. Springer-Verlag.

[5] M. Betke, J. Gips, and P. Fleming. The camera mouse: visual tracking of body features to provide computer access for people with severe disabilities. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 10(1):1–10, March 2002.

[6] J. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm. Technical report, Intel Corporation Microprocessor Research Labs, 2000.

[7] O. Camps, R. Cucchiara, A. Del Bimbo, J. Matas, F. Pernici, and S. Sclaroff. Long-term detection and tracking. http://www.micc.unifi.it/LTDT2014/, 2014.

[8] AI Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *Visualization and Computer Graphics, IEEE Transactions on*, 12(4):615–628, July 2006.

[9] L. Čehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(4):941–953, 2013.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[11] R.W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal, The*, 29(2):147–160, April 1950.

[12] S. Hare, A. Saffari, and P. H. S. Torr. Efficient online structured output learning for keypoint-based object tracking. In *Computer Vision and Pattern Recognition*, pages 1894–1901. IEEE, June 2012.

[13] P. V. C. Hough. Machine Analysis of Bubble Chamber Pictures. In *International Conference on High Energy Accelerators and Instrumentation*, CERN, 1959.

[14] Weiming Hu, Tieniu Tan, Liang Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, Aug 2004.

[15] Weiming Hu, Tieniu Tan, Liang Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, Aug 2004.

[16] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, July 2012.

[17] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME – Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[18] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Fernandez, T. Vojir, G. Nebehay, et al. The visual object tracking VOT2014 challenge results. *VOT2014 workshop, ECCV*, 2014.

[19] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555, Nov 2011.

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[21] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. pages 674–679, 1981.

[22] M. E. Maresca and A. Petrosino. Matrioska: A multi-level approach to fast tracking by learning. In Alfredo Petrosino, editor, *Image Analysis and Processing – ICIAP 2013*, volume 8157 of *Lecture Notes in Computer Science*, pages 419–428. Springer Berlin Heidelberg, 2013.

[23] F. Pernici and A. Del Bimbo. Object tracking by oversampling local features. *Pattern Analysis and Machine Intelligence*, 2013.

[24] R. Pflugfelder, M. Kristan, A. Leonardis, and J. Matas. Visual object tracking challenge. http://www.votchallenge.net/vot2014, May 2014.

[25] C. Rother, V. Kolmogorov, and Blake A. "grabcut" – interactive foreground extraction using iterated graph cuts. *ACM TRANS. GRAPH*, pages 309–314, 2004.

[26] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, Jun 1994.

[27] D. J. Tan, S. Holzer, N. Navab, S. Ilic, and AG Siemens. Deformable template tracking in 1ms. *British Machine Vision Conference*, 2014.

[28] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:2000, 2000.

[29] P. Vandergheynst, R. Ortiz, and A. Alahi. Freak: Fast retina keypoint. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:510–517, 2012.

[30] T. Zhang, K. Jia, C. Xu, Y. Ma, and N. Ahuja. Robust visual tracking via consistent low-rank sparse learning. *International Journal of Computer Vision*, pages 1–20, 2014.