Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Izboljšan robusten model z regijami za vizualno sledenje objektov

Alan Lukežič

Delo je pripravljeno v skladu s Pravilnikom o podeljevanju Prešernovih nagrad študentom, pod mentorstvom doc.dr. Mateja Kristana

Ljubljana, 2015

University of Ljubljana
Faculty of Computer and Information Science

Improved robust part-based model for visual object tracking

Alan Lukežič

The work has been prepared under mentorship of doc. dr. Matej Kristan in accordance with the regulations of Prešeren student awards.

Ljubljana, 2015

Povzetek

Magistrska naloga obravnava kratkotrajno vizualno sledenje objektov s pomočjo deformabilnih modelov z regijami. Ti modeli kažejo izjemen potencial pri naslavljanju nerigidnih deformacij in delnega zakrivanja objektov, vendar se pogosto odrežejo slabše od holističnih pristopov, ki objekt modelirajo z enim samim globalnim modelom izgleda. Običajno so vzrok za slabše delovanje veliko število prostih parametrov, ki jih sledilnik ocenjuje, in poenostavitve v topologiji konstelacije, ki so potrebne za delovanje v realnem času. Pogosto so tudi geometrijske in vizualne omejitve kombinirane neprincipelno. Za razliko od opisanih pristopov v nalogi predstavljamo generativni model, ki vizualne in geometrijske omejitve združuje v sistem vzmeti s konveksno energijsko funkcijo. Predlagamo tudi optimizacijsko metodo, ki učinkovito minimizira energijo polno povezanega sistema vzmeti. Predlagano metodo primerjamo z obstoječim optimizacijskim pristopom, ki ga naša metoda presega tako v smislu hitrosti kot tudi numerične stabilnosti. V nalogi predlagamo sledilnik z deli, ki kombinira dve stopnji podrobnosti vizualnega modela, in sicer grobo- in srednjenivojsko predstavitev tarče. Za lokalizacijo regij na srednje-nivojski predstavitvi uporabimo predlagano rešitev hitre optimizacije sistema vzmeti. Razvit sledilnik rigorozno primerjamo s trenutno najboljšimi metodami sledenja znotraj tekmovanja VOT2014, ter analiziramo sestavne dele sledilnika, saj primerjamo vpliv posameznih komponent na samo delovanje. Rezultati kažejo, da je predlagan sledilnik boljši tako od testiranih holističnih sledilnikov, kot tudi od najboljših objavljenih sledilnikov na VOT2014, ki temeljijo na regijah. Poleg tega sledilnik deluje v realnem času.

Ključne besede

računalniški vid, vizualno sledenje objektom, deformabilni modeli, korelacijski filtri, sistemi vzmeti

Abstract

This thesis addresses short-term visual object tracking by deformable parts models (DPM). The DPMs show a great potential in addressing non-rigid object deformations and self-occlusions, but according to recent benchmarks, they often lag behind the holistic approaches, which model an object with a single appearance model. The reason is that potentially large number of parameters in constellation needs to be estimated for target localization and simplifications of the constellation topology are often assumed to make the inference tractable. Furthermore, the visual model and geometric constraints are usually combined in an ad-hoc fashion. In contrast to related approaches, we present a generative model that jointly treats contributions of the visual and of the geometric model as a single physics-based spring system with a convex energy function. An efficient optimization method is proposed for this dual form that allows MAP inference of a fully-connected constellation model. The proposed optimization method is compared to the existing optimization approach and outperforms it in terms of stability and efficiency. In the thesis we propose a part-based tracker that combines two visual representations of the target, i.e., coarse and mid-level representation. The proposed optimization method is used for target localization on the mid-level representation. The resulting tracker is rigorously analyzed on a highly challenging VOT2014 benchmark, it outperforms the related part-based and holistic trackers including the winner of the VOT2014 challenge and runs in real-time. The design of the proposed tracker is analyzed by an analysis of each component of the tracker.

Keywords

computer vision, visual object tracking, deformable-part models, correlation filters, spring systems

Contents

Razširjeni Povzetek

Vizualno sledenje objektov je eden od temeljnih problemov računalniškega vida. Na kratko lahko povzamemo problem sledenja kot iskanje položaja objekta skozi sekvenco slik. Avtomatsko sledenje izkazuje svoj potencial v velikem številu aplikacij, saj se pogosto uporablja v nadzornih sistemih, urejanju videa ali vmesnikih za komunikacijo z računalnikom. Problem vizualnega sledenja je zahteven zaradi številnih sprememb, ki se lahko med sledenjem pojavijo. Objekt lahko spremeni svoj izgled (npr. se deformira) ali pa ga delno ali popolno zakrije drug objekt. Zaradi spremenjene razdalje sledečega objekta do kamere prihaja do sprememb v velikosti objekta na sliki. Objekt oziroma kamera se običajno tudi premikata po sceni, kar še dodatno otežuje sledenje. V nalogi obravnavamo kratkotrajno sledenje ene tarče z eno kamero. Sledilnik uvrščamo med kratkotrajne, saj mora napovedati pozicijo objekta za vsako sliko v sekvenci, medtem pa ne izvaja ponovne detekcije tarče. Poleg tega ima sledilnik še to omejitev, da je edina informacija, ki jo dobi na začetku, zgolj položaj objekta v prvi sliki. Prav tako sledilnik ne sme uporabljati informacije iz bodočih slik v sekvenci.

V zadnjem času je kratkotrajno vizualno sledenje deležno precej pozornosti s strani raziskovalcev, saj je bilo organiziranih nekaj tekmovanj, ki primerjajo najboljše sledilnike [1, 2, 3]. Rezultati tekmovanj kažejo, da dve skupini sledilnikov dosegata precej dobre rezultate, in sicer holistični sledilniki ter tisti, ki sledijo s pomočjo regij. Oba pristopa sta prikazana v Sliki 1.

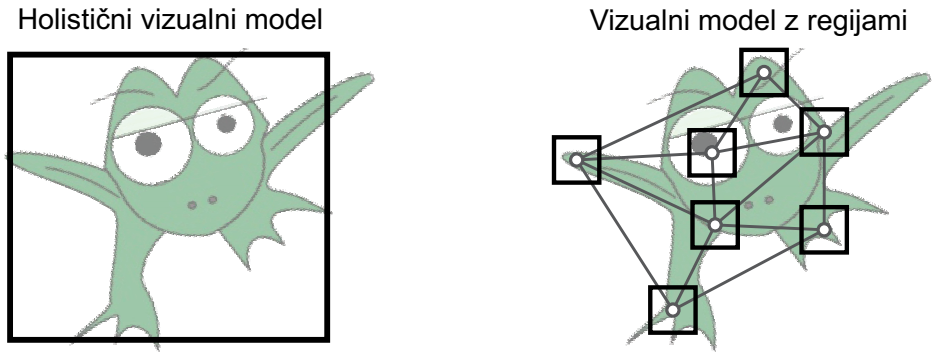

Figure 1: Razlika med holističnim vizualnim modelom (levo) z globalno predstavitvijo tarče in modelom z regijami (desno), ki pokriva tarčo z več povezanimi deli.

Holistični sledilniki predstavijo tarčo globalno, z eno regijo, čemur rečemo tudi holistični vizualni model. Rezultati kažejo, da je ta skupina sledilnikov sposobna zelo robustno slediti objektu, saj je število prostih parametrov, ki jih je potrebno oceniti, tipično majhno. Poleg tega vizualni model pokriva celoten objekt. Kljub temu imajo ti sledilniki v določenih situacijah težave. Ker se učijo izgleda celotne tarče, tipično niso sposobni zaznati zakrivanja, zato se vizualni model v teh primerih lahko pokvari, kar se odraža v slabših rezultatih sledenja ter celo odpovedi sledilnika. Holistični sledilniki običajno nenaravno naslavljajo spremembo velikosti objekta, saj bodisi sledijo z nespremenjeno velikostjo regije ali pa izčrpno preiskujejo čez različne skale v vsaki sliki [4, 5, 6]. Te probleme naslavljajo sledilniki z regijami, ki so opisani v nadaljevanju. Holistični sledilniki se, glede na to, kako formulirajo lokalizacijo objekta v sliki, delijo na dve skupini: generativne in diskriminativne. Generativni sledilniki učijo vizualni model izgleda tarče, na podlagi katerega iščejo najbolj podobno regijo v sliki. Primeri takih sledilnikov so predstavljeni v [7], kjer se za predstavitev tarče uporablja barvni histogram, ali pa podprostorske metode predstavljene v [8, 9, 10]. Nekateri avtorji predlagajo kombinacijo različnih značilnic, npr. [11, 7]. Generativni sledilniki imajo tudi določene pomanjkljivosti, saj za uspešno sledenje zahtevajo ve-

liko število učnih primerov. Ob večjih spremembah izgleda objekta pogosto odpovejo. Poleg tega pa ne uporabljajo informacije o ozadju v sliki, kar lahko izboljša rezultat sledenja. Zadnji problem naslavljajo diskriminativni sledilniki, ki sledenje predstavijo kot klasifikacijski problem, v katerem poizkušajo čim bolje ločiti objekt od ozadja. V to skupino spadajo sledilniki, ki delujejo s pomočjo metode podpornih vektorjev [12], Adaboost [13] in učenje z več instancami [14]. V zadnjem času so zelo dobre rezultate dosegli sledilniki, ki delujejo na podlagi strukturiranih podpornih vektorjev (Struck) [15] in korelacijskih filtrov [16, 4, 5, 17, 18]. Tem sledilnikom je skupno učenje z regresijo. Bolme et al. [16] je eden prvih, ki je sledenje predstavil s korelacijskim filtrom. Njegovo delo je bilo v nadaljevanju nadgrajeno v različnih sledilnikih [4, 5, 17, 18].

Sledilniki, ki sledijo s pomočjo regij, predstavijo tarčo z več regijami povezanimi v konstelacijo za zagotavljanje geometrijskih omejitev. Ti sledilniki tipično uspešno naslavljajo pojav zakrivanja med sledenjem, saj objekt, ki zakriva tarčo, vpliva le na določeno število regij in tako ne pokvari celotnega modela. Običajno so uspešni tudi pri obvladovanju deformacij objekta. Sledenje z regijami je predstavljeno v [19, 20], kjer je tarča predstavljena z velikim številom manjših regij, med sledenjem pa se iz konstelacije avtomatično odstranjujejo neinformativne regije ter dodajo nove. Podoben princip uporablja sledilnik LGT [21], ki za sledenje uporablja dve plasti, regijam na lokalni plasti pa je dodana lokalna povezanost, ki zagotavlja geometrijske omejitve. V magistrski nalogi predlagamo nov sledilnik, ki za razliko od obstoječih pristopov predstavi geometrijske in vizualne omejitve z enotnim, polno povezanim sistemom vzmeti. Vzmeti med seboj povezujejo regije, ki niso zelo majhne v primerjavi z velikostjo tarče, ampak vsaka predstavlja približno četrtino objekta.

Predlagani sledilnik je sestavljen iz dveh plasti, njegovo delovanje pa je prikazano na Sliki 2. Zaradi dvoplastne arhitekture je poimenovan dvoslojni deformabilni sledilnik z deli (angl., layered deformable parts tracker, LDP). Prva plast je groba reprezentacija tarče, ki skrbi za približno lokalizacijo

objekta, druga plast pa je srednjenivojska reprezentacija, katere naloga je upoštevanje deformacije objekta in prilagajanje njegove velikosti. Obe plasti sta opisani v nadaljevanju.

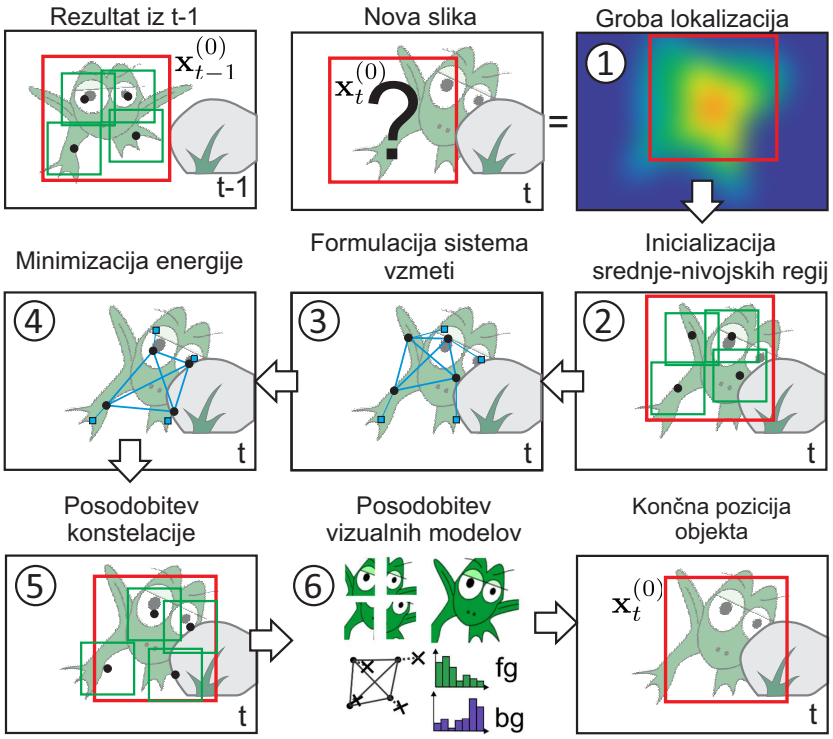

Figure 2: Vizualizacija sledenja s sledilnikom LDP.

Groba reprezentacija tarče je sestavljena iz dveh komponent, globalne vizualne predstavitve tarče in globalnega barvnega modela, ki je sestavljen iz barvnih histogramov objekta in ozadja. Oba modela sta uporabljena na vsakem koraku sledenja za ocenitev položaja središča objekta. Ocena je predstavljena kot položaj maksimalne vrednosti v verjetnostni mapi, dobljeni z modeloma. Globalni vizualni model je predstavljen kot korelacijski filter KCF [4], globalni barvni model pa kot barvni histogram tarče in barvni histogram njene neposredne okolice. Verjetnostna mapa je izračunana kot produkt rezultata obeh modelov.

Srednjenivojska reprezentacija tarče je sestavljena iz množice N_p srednje

velikih regij (v našem primeru $N_p = 4$), ki so povezane s sistemom vzmeti. Porazdelitev za stanje sistema vzmeti, ki je določeno s položajem regij $\mathbf{X}_t = \{\mathbf{x}_t^{(i)}\}_{i=1:N_p}$, je definirana kot

$$p(\mathbf{X}_t|\mathbf{Y}_t, \Theta) \propto p(\mathbf{Y}_t|\mathbf{X}_t, \Theta)p(\mathbf{X}_t, \Theta). \tag{1}$$

Porazdelitev $p(\mathbf{Y}_t|\mathbf{X}_t, \Theta)$ predstavlja vizualne omejitve, določene z meritvami $\mathbf{Y}_t$, porazdelitev $p(\mathbf{X}_t, \Theta)$ pa predstavlja geometrijske omejitve določene s parametri deformacijskega modela Θ, definiranega kot sistem vzmeti. Oba tipa omejitev sta modelirana z normalno porazdelitvijo, zato jih lahko obravnavamo v enem sistemu, in sicer kot produkt verjetnosti. Ta predpostavka nas pripelje do eksponentne verjetnosti $p(\mathbf{X}_t|\mathbf{Y}_t, \Theta) \propto \exp(-E)$, kjer je člen E modeliran kot funkcija

$$E = \frac{1}{2} \sum_{i=1:N_p} k_t^{(i)} \left\| d_t^{(i)} \right\|^2 + \sum_{i,j \in \mathcal{L}} k_t^{(i,j)} (\mu_t^{(i,j)} - \left\| d_t^{(i,j)} \right\|)^2, \tag{2}$$

ki predstavlja energijsko funkcijo sistema vzmeti. Geometrijske omejitve so definirane kot dinamična vozlišča (krogi na Sliki 3), vizualne omejitve pa kot statična vozlišča v sistemu vzmeti (črni kvadrati na Sliki 3). Leva (zunanja) vsota v (2) predstavlja vizualne omejitve (statične vzmeti), desna (notranja) vsota pa geometrijske omejitve (dinamične vzmeti). Oznaki $k_t^{(i)}$ in $d_t^{(i)}$ predstavljata prožnostni koeficient in dolžino statične vzmeti. Dinamična vzmet, ki povezuje i-to in j-to dinamično vozlišče je določena s prožnostnim koeficientom $k_t^{(i,j)}$, nominalno dolžino $\mu_t^{(i,j)}$ in trenutno dolžino vzmeti $d_t^{(i,j)}$.

Za minimizacijo energijske funkcije iz (2) se običajno uporabljajo metode, ki temeljijo na odvodu funkcije, kot je na primer metoda konjugiranih gradientov (CGD). Ker pa za enodimenzionalni sistem vzmeti obstaja direktna oblika rešitve, se lahko ta lastnost uporabi pri optimizaciji dvodimenzionalnega sistema in tako doseže hitrejše delovanje. Predlagamo iterativno direktno metodo za optimizacijo sistema vzmeti. Metoda v vsaki iteraciji *razbije* sistem na dva enodimenzionalna sistema vzmeti in ju reši z direktno metodo. V naslednjem koraku se sistema združita v enega dvodimezionalnega. Ker je energijska funkcija sistema vzmeti konveksna, algoritem v končnem številu korakov deseže minimum.

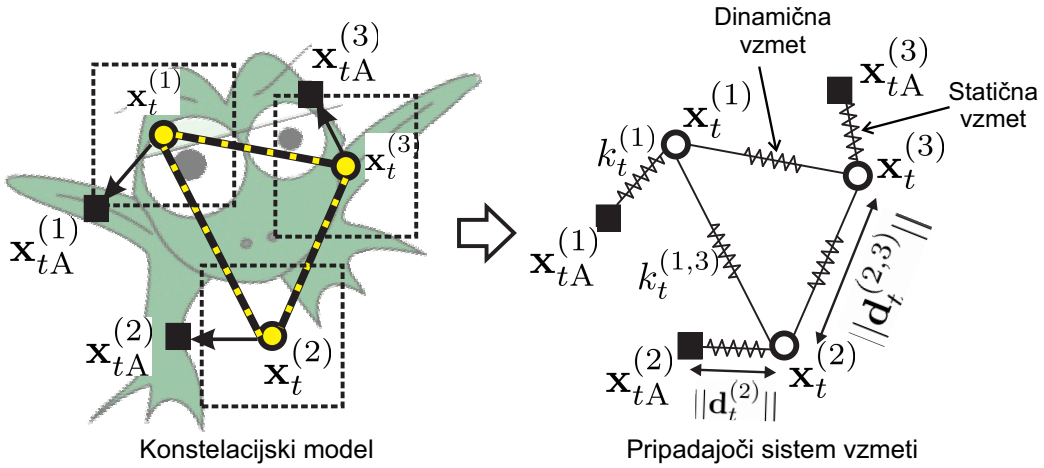

Figure 3: Primer konstelacijskega modela s pravokotnimi regijami in puščicami, ki kažejo proti vizualno najbolj podobnim pozicijam (levo), ter pripadajoč sistem vzmeti (desno).

Delovanje sledilnika je analizirano in primerjano s trenutno najboljšimi obstoječimi metodami za vizualno sledenje na sistemu VOT2014 [3]. Sistem je sestavljen iz zbirke 25 sekvenc, ki vsebujejo scene iz različnih domen. Na vsaki sliki iz zbirke je izračunano prekrivanje regije, ki jo napove sledilnik z anotirano regijo. Ko prekrivanja ni več, sistem zabeleži odpoved sledilnika, sledenje pa se nadaljuje čez pet slik. Delovanje sledilnika je ovrednoteno na podlagi dveh mer, robustnosti in natančnosti. Robustnost predstavlja število odpovedi sledilnika na neki sekvenci, natančnost pa povprečno prekrivanje regije, ki jo napove sledilnik z anotiranimi regijami. Sestavni deli sledilnika in njihov vpliv na sledenje so analizirani tako, da so različne variante sledilnika LDP primerjane znotraj sistema VOT2014. Predlagan sledilnik s štirimi regijami je označen z oznako LDP, sledilnika s tremi oziroma devetimi regijami sta označena z LDP-3 oziroma LDP-9. Sledilnik brez segmentacije na grobi predstavitvi je označen z LDP-nosegm, sledilnik LDP-nomid je sestavljen brez srednjenivojske predstavitve, LDP-nosprings je sledilnik brez sistema vzmeti ter LDP-root-only predstavlja sledilnik z zgolj vizualnim modelom na grobi predstavitvi (torej brez segmentacije in brez srednjenivojske pred-

stavitve). Sledilnika LDP-star in LDP-pairwise sta sledilnika s spremenjeno topologijo sistema vzmeti. Zvezdasta topologija je označena z oznako LDP-star, medtem ko LDP-pairwise predstavlja sledilnik s topologijo, ki sta ji odvzeti diagonalni vzmeti. Primerjava rezultatov je prikazana na AR-grafu, na Sliki 4. Na abscisni osi je podano povprečno število odpovedi sledilnika (robustnost), na ordinatni osi pa povprečno prekrivanje regije z anotiranimi vrednostmi (natančnost). Vrednosti so uteženo povprečene čez sekvence, utež pa je proporcionalna dolžini posamezne sekvence. Idealna pozicija v grafu je zgornji levi vogal, ki predstavlja nič odpovedi in 100 % povprečno prekrivanje. Rezultati kažejo, da se sledilnik LDP z vsemi komponentami odreže najbolje od vseh variant.

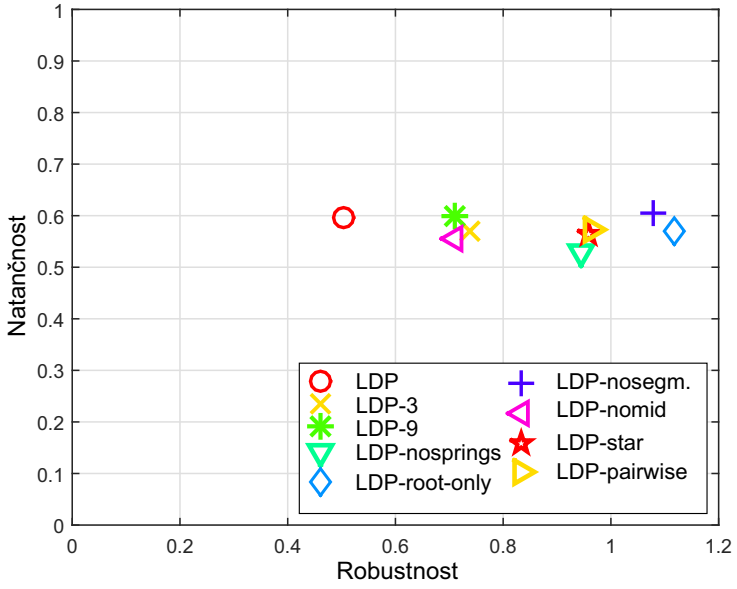

Figure 4: Primerjava različnih variant sledilnika LDP. Na grafu sta prikazana robustnost (povprečne število odpovedi) in natančnost.

Primerjava predlaganega sledilnika z obstoječimi metodami za vizualno sledenje je prav tako izvedena na sistemu VOT2014. Rezultati so, podobno kot rezultati analize sestavnih delov sledilnika, prikazani na AR-grafu na Sliki 5. Opaziti je mogoče, da je sledilnik LDP najbližje točki idealnega sle-

denja, saj je po natančnosti primerljiv z najnatančnejšimi sledilniki, medtem ko je bistveno boljši v robustnosti.

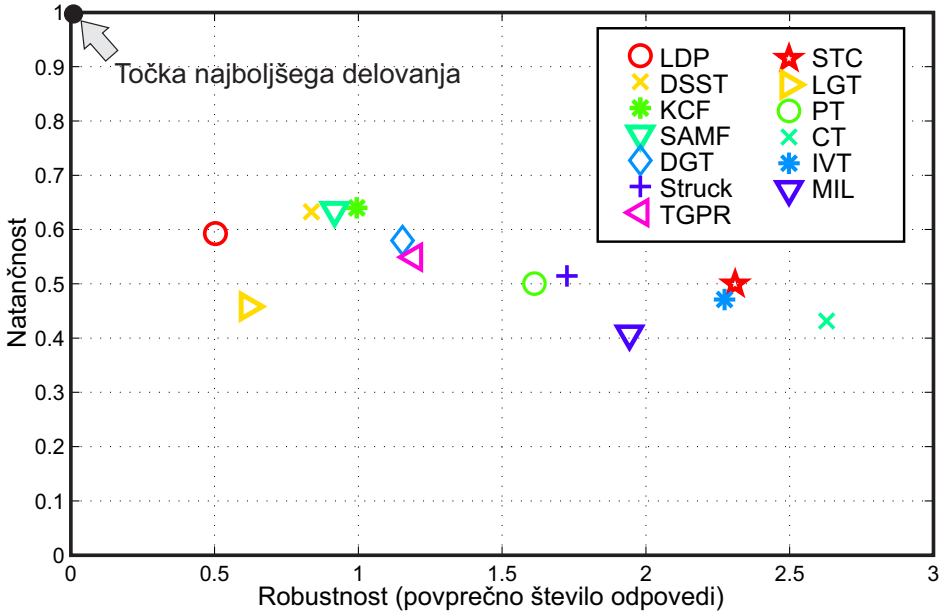

Figure 5: Graf prikazuje vsak sledilnik kot točko v prostoru natačnost/število odpovedi. Vrednosti so povprečene preko sekvenc.

VOT2014 nudi tudi analizo sledilnikov in razvrstitev v obliki tekmovanja. Rezultat primerjave sledilnika LDP z ostalimi je prikazan v Tabeli 1, kjer številke prikazujejo povprečne razvrstitve po sekvencah (prvi stolpec) oziroma po vizualnih atributih (drugi stolpec). Vizualni atributi predstavljajo sestavni del vsake sekvence v podatkovni zbirki tako, da je za vsako sliko podano, kateri atributi so prisotni. Rezultati kažejo, da sledilnik LDP v povprečju doseže najboljšo uvrstitev med primerjanimi.

	sekvence			viz. atributi			skupaj
sledilnik	A	R	Av	A	R	Av	Av
LDP	2.60	**1.80**	**2.20**	2.00	*1.67*	*1.83*	**2.02**
KCF	**1.96**	3.16	*2.56*	**1.17**	2.50	*1.83*	*2.20*
DSST	2.40	2.96	2.68	**1.17**	2.33	**1.75**	2.21
SAMF	*2.24*	3.16	2.70	*1.33*	2.50	1.91	2.31
DGT	3.88	*2.48*	3.18	3.50	2.50	3.00	3.09
TGPR	3.76	4.8	4.28	5.00	4.33	4.66	4.47
LGT	6.20	3.40	4.80	7.33	**1.33**	4.33	4.56
Struck	4.64	5.72	5.18	4.67	4.83	4.75	4.96
PT	5.48	5.20	5.34	5.33	4.67	5.00	5.17
STC	5.76	6.96	6.36	6.17	9.33	7.75	7.05
IVT	6.32	7.72	7.02	6.83	9.17	8.00	7.51
MIL	8.32	5.92	7.12	10.17	6.50	8.33	7.73
CT	7.60	7.08	7.34	9.17	7.50	8.33	7.84

Table 1: Razvrstitve sledilnikov glede na natančnost (A), robustnost (R) in povprečje obeh vrednosti (Av). Manjša vrednost razvrstitve pomeni boljšo uvrstitev in s tem boljše delovanje. Zadnji stolpec predstavlja povprečje obeh vidikov razvrščanja sledilnikov (po sekvencah in po vizualnih atributih). Najboljši rezultat v vsakem stolpcu je obarvan rdeče, drugi modro in tretji zeleno.

Predlagana optimizacijska metoda IDA za minimizacijo energijske funkcije sistema vzmeti je primerjana z metodo konjugiranih gradientov (CGD) po številu iteracij potrebnih za konvergenco, energiji, ki ostane v sistemu po optimizaciji, in po času potrebnem za optimizacijo. Rezultati obeh metod so prikazani v Tabeli 2 in so dobljeni tako, da so povprečeni preko sto tisoč različnih, naključno generiranih sistemov vzmeti. Primerjava prikazuje, da metoda IDA dosega primerljivo končno energijo sistema s približno pol manj iteracijami kot metoda CGD. Zadnji stolpec v Tabeli 2 prikazuje, da metoda IDA potrebuje povprečno 3ms za optimizacijo, medtem ko je metoda CGD

več kot trikrat počasnejša.

	Število iteracij			Energija			Čas
Metoda	Povpr.	St. dev	Mediana	Povpr.	St. dev	Mediana	Povpr. [ms]
IDA	12.75	9.32	10	1.22	3.34	0.83	2.92
CGD	28.01	9.90	28	1.28	3.34	0.83	9.72

Table 2: Primerjava predlagane iterativne optimizacijske metode (IDA) in metode konjugiranih gradientov (CGD).

Aplikacije in diseminacija

Tematiko, ki smo jo obravnavali v magistrski nalogi smo predstavili na več mednarodnih znanstvenih konferencah. Preliminare rezultate sledenja smo objavili v članku rezultatov primerjave sledilnikov na izzivu VOT Challenge 2014 [3], ki je bil predstavljen na delavnici v okviru ene od glavnih konferenc računalniškega vida European Conference on Computer Vision, ECCV2014. Predlagan vizualni sledilnik je bil predstavljen na konferenci Computer Vision Winter Workshop v Avstriji, v okviru vabljenega predavanja [22]. Sledilnik smo primerjali s trenutno najboljšimi obstoječimi metodami za vizualno sledenje na sistemu iz tekmovanja VOT Challenge 2014 [3]. Ker je bil sledilnik razvit po koncu tekmovanja, se tega ni mogel udeležiti uradno, vendar so vsi delni rezultati s tekmovanja javno dostopni. Naša analiza je pokazala, da naš sledilnik prekaša najboljše sledilnike na tem tekmovanju, na primer po robustnosti je za 40% boljši od zmagovalca tekmovanja VOT2014. Učinkovito optimizacijo sistema vzmeti, ki je opisana v magistrski nalogi, smo predstavili na konferenci ERK2015 [25].

Zaradi zelo dobrih rezultatov na standardnih zbirkah smo sledilnik preizkusili še na dveh zelo zahtevnih in raznolikih realnih problemih. Prvi projekt se je imenoval DroneTrack, njegov namen pa je bil omogočiti kvadrokopterju avtonomno zasledovanje objekta, ko se ta giblje skozi prostor zgolj z uporabo

kamere. Taka aplikacija je izjemno zahtevna za vizualno sledenje, saj zahteva zelo robusten sledilnik in delovanje v realnem času. Zato smo sledilnik reimplementirali v programskem jeziku C++ in dodali pohitritve ter poenostavitve, kar je omogočalo delovanje s hitrostjo 50 slik na sekundo na standardnem prenosniku. Na Sliki 6 levo je prikazan pogled iz kamere na kvadrokopterju med sledenjem osebe, shema sistema pa je prikazana desno. Predstavitveni video, ki je nastal kot rezultat projekta in je bil predstavljen na noči raziskovalcev septembra 2015 [23], je bil objavljen tudi na spletnem portalu Frekvenca X na Valu202 [24].

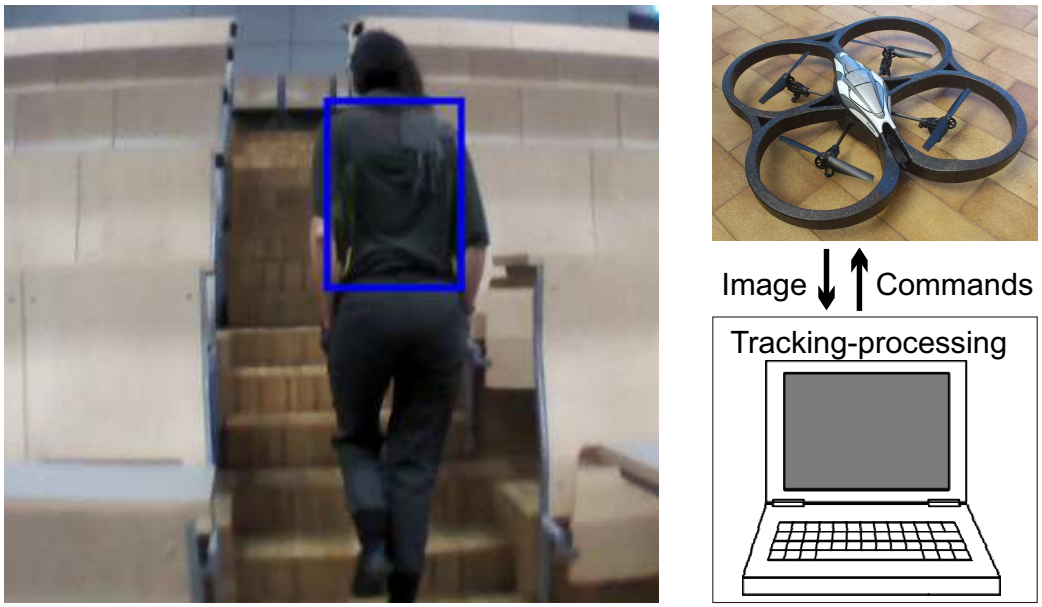

Figure 6: Sistem za vizualno sledenje objektov s kvadrokopterjem. Na desni strani je prikazana komunikacija med kvadrokopterjem in računalnikom. Kvadrokopter računalniku pošilja zaporedne slike, ki jih zajame s kamero, na računalniku se izvaja vizualno sledenje in izračun odzivov kvadrokopterja glede na položaj objekta v sliki. Ukazi za krmiljenje se nato pošljejo nazaj na kvadrokopter. Na levi strani je prikazana slika iz kvadrokopterja in rezultat vizualnega sledilnika.

V sklopu raziskovalnega projekta ARRS L2-6765 [26] smo sledilnik aplicirali na problem sledenja prometnih znakov v mobilnem kartirnem sistemu. Ta problem je bil še posebej zahteven saj znak med sledenjem občutno spremeni svojo velikost (do 5% začetne velikosti). Sledilnik se je odrezal odlično, kvalitativni rezultati sledenja prometnih znakov v implementiranem sistemu pa so prikazani na Sliki 7.

Rezultati eksperimentalne evaluacije na standardnih podatkovnih zbirkah potrjujejo, da se sledilnik umešča med najboljše sledilnike na področju sledenja in celo prekaša zmagovalca zadnjega tekmovanja VOT2014. Rezultati aplikacije na dveh zahtevnih domenah pa potrjujejo visoko praktično učinkovitost razvitega sledilnika. Sledilnik tako predstavlja znanstveni napredek na področju sledenja, hkrati pa ga odlikuje praktična uporabnost. V pripravi je članek z opisom in evalvacijo sledilnika, ki ga nameravamo poslati na glavno revijo s področja računalniškega vida.

Figure 7: Primer vizualnega sledenja prometnih znakov iz kamere na avtomobilu v mobilnem kartirnem sistemu. Opaziti je mogoče, da sledilnik zelo dobro naslavlja tudi spremembo skale, saj se velikost prometnega znaka preko sekvence signifikantno zmanjšuje.

Chapter 1

Introduction

Visual object tracking is one of the fundamental problems in computer vision. Knowing the position of the object in current (i-th) frame, the task is to determine the location of the object in the next frame, as shown in Figure 1.1. Estimating the trajectory of an object through the sequence of frames appears in many real-world problems, therefore it is interesting and highly useful for numerous applications e.g., surveillance systems, gesture recognition, video editing or interfaces for computer-human interaction. The problem of visual tracking is challenging for numerous reasons: the object can change its appearance (e.g., deformation), it can be occluded by another object or by itself (self-occlusion) and it can change its size due to the changed distance from the camera. Furthermore, camera or object often move and also the illumination of the scene can change significantly during tracking. Visual tracking includes tracking of one (*single-target*) or many targets (*multi-target*). It differs also by the modalities used, typically by tracking with one (*single-camera*) or many cameras (*multi-camera*). Visual trackers are devided into two groups by the intention of tracking. *Short-term* trackers do not perform reinitialization of the target when it is lost and output the target position at every frame. On the other hand, *long-term* trackers automatically detect when target is lost or outside the frame and perform re-detection of the target. This thesis addresses short-term, single-camera,

Figure 1.1: Main task in visual tracking: given two sequential frames from a video and the position of an object in t-th frame, the position of an object in frame $t + 1$ is estimated.

single-target, model-free, causal visual tracking. The *model-free* property means that except from the position of the object in the first frame, no prior knowledge of the object is given. The *causality* of the tracker represents that no future frames or frames prior the initialization are used.

Short-term single-target visual tracking has received a significant attention over the last decade and recently several papers reporting experimental comparison of trackers on a common testing ground have been published [1, 27, 2, 3]. Results show that tracking quality depends highly on the expressiveness of the feature space in the target appearance model and the inference algorithm that converts the features into a score of target presence. Most of the popular trackers apply holistic visual models which capture the target appearance by a single patch. In combination with efficient machine-learning and signal processing techniques from online classification and regression, these trackers exhibited top performance across all benchmarks [15, 14, 13, 16]. Most of these approaches localize the target by sliding windows and some apply a greedy search over nearby scales [5, 17, 4, 6] to address the scale change as well.

Nevertheless, a single patch often poorly approximates the targets that undergo significant deformation, self-occlusion and partial occlusions, leading to drift and eventual failure. These situations are conceptually better ad-

dressed by part-based models that decompose the target into a constellation of parts. These types of trackers show a great potential in tracking non-rigid objects, but the performance often falls behind the holistic models [3], which is due to the large number of free parameters estimated in the deformation model. Most part-based trackers use very small parts, apply low-level features for the visual models, e.g., histograms [21, 28] or keypoints [29, 30] and increase their discrimination power by increasing the number of parts. The target is localized by optimizing a tradeoff between the visual and geometric agreement. For tractability, most of the recent trackers use star-based topology, e.g. [28, 31, 32, 33, 34, 29], or local connectivity, e.g. [21], instead of a fully-connected constellation [30], but at a cost of a reduced power of the geometric model.

1.1 Our approach and contributions

Our main contribution is a new class of fully-connected part-based trackers for robust tracking. In contrast to using a large number of small detailed parts, a small set of fully connected mid-level parts is used. Note that many approaches claim using a spring-like model on geometry only and apply ad-hoc combination with the visual models often leading to nonconvex optimization. In contrast we show that the geometric and visual constraints can be jointly modeled by a single equivalent *physical* spring system and the MAP inference is achieved by minimizing a convex cost function. To the best of our knowledge, this is the first work to efficiently jointly treat the visual and the geometric constraints within a single spring system. The proposed tracker combines a coarse object representation with a mid-level deformable parts model in top-down localization and bottom-up updates (Figure 1.2). The coarse model initializes the mid-level representation at approximate object location. An equivalent spring system is formed and optimized, yielding a MAP constellation estimate. The parts are updated and the estimated constellation is used to update the coarse representation. Due to the two-layer

architecture, the proposed tracker is called Layered Deformable Parts tracker (LDP). Our tracker is rigorously compared against a set of state-of-the-art

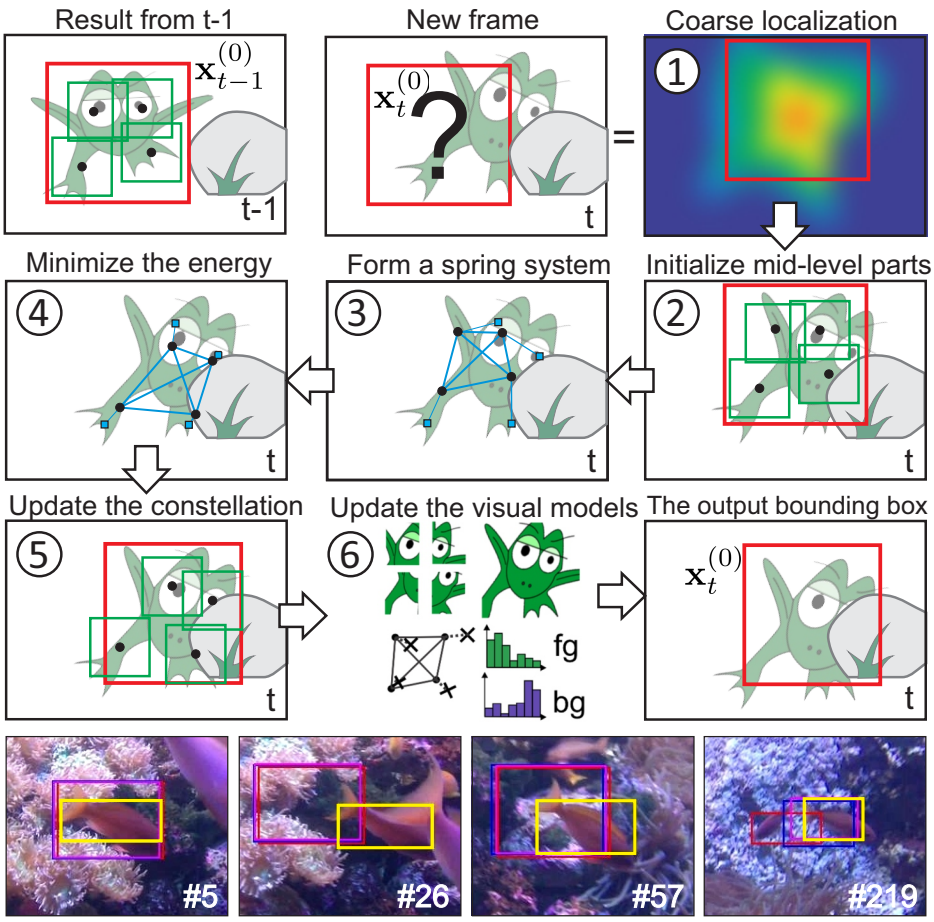

Figure 1.2: Illustration of coarse-to-fine tracking by spring system energy minimization in a deformable part model (top). Tracking examples with our tracker LDP (yellow), KCF (red), IVT (blue) and Struck (magenta) are shown in the bottom.

trackers on a highly challenging recent benchmark VOT2014 [3] and outperforms all trackers, including the winner of the VOT2014 challenge. Additional tests show that improvements come from the fully-connected constellation and the top-down/bottom-up combination of the coarse representation with the proposed deformable parts model.

1.2 Thesis outline

The remainder of the thesis is organized as follows. Chapter 2 describes related work and presents different tracking approaches. Chapter 3 details an approach for fast learning of so-called correlation filters that we use for part localization and learning in the proposed tracker. Chapter 4 presents the proposed part-based tracker and provides important implementation details. In Chapter 5 experimental results are presented. Our tracker is evaluated on VOT2014 [3] benchmark which is described in Section 5.2. Analysis of the tracker is presented in Section 5.3 and in Section 5.4 comparison with state-of-the-art trackers is given. In Chapter 6 we draw the conclusion and discuss the future work.

Chapter 2

Related work

This section overviews the most related works and points out the differences with our approach. Depending on the approach for target localization, the trackers are divided into two classes: (i) generative trackers and (ii) discriminative trackers. If the target is modeled as a whole, the visual model is treated as holistic (see Figure 2.1 left), while if it is modeled with several parts, the approach is called part-based visual model (see Figure 2.1 right). Both approaches with the state-of-the-art methods are described in the following.

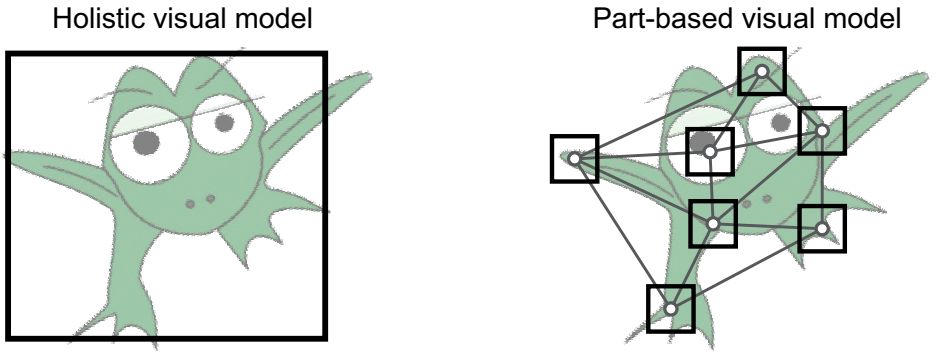

Figure 2.1: Difference between holistic visual model (left) where target is modeled globally and part-based visual model (right) with many small patches modeling local parts of the target.

2.1 Holistic trackers

Holistic models are known by the robust tracking, because of the small number of free parameters which need to be estimated and they cover the whole target by the global visual model. Holistic visual models also have some issues: they do not explicitly address partial occlusion and object changes e.g., deformation. Occlusion is especially problematic since even partial occlusion with another object corrupts the entire visual model, which leads to reduced performance and potential failure. Another issue of holistic trackers is scale change adaptation which is also not addressed naturally. In particular, holistic trackers track with the fixed bounding box or they adapt size with exhaustive search over nearby scales [4, 5, 6]. Trackers that apply holistic models can be divided into two classes, generative and discriminative, depending on the formulation of the target localization by their visual model. Most related holistic trackers from these two classes are reviewed in the following.

2.1.1 Generative models

Generative models typically learn a target appearance model and localize it by finding an image position that maximizes the similarity between the model with the features extracted at that position. Early work in holistic models explored generative models like color histograms [7] and subspace-based templates [8, 9] or sparse reconstruction templates [10]. Black et al. [35] presented generative eigen-tracker which learns an eigenspace to represent the object. Mei et al. [36] proposed a tracker using ℓ_1 minimization within a particle filter framework [37]. To reduce computational complexity of a ℓ_1 tracker, dimensionality reduction and orthogonal matching pursuit algorithm was used in [38]. Several authors applied feature combination [7, 11] and recently Gaussian process regression was proposed for efficient updating [39]. Generative trackers have the following common issues. They need a lot of training examples extracted from an image to learn appearance model which

can lead to slow performance. They tend to drift from target at significant appearance changes of an object. They also do not use visual information of background surrounding an object, which can often help to improve tracking performance. The latter issue is better addressed by the discriminative trackers.

2.1.2 Discriminative models

Trackers with discriminative models formulate tracking as a classification problem. In these models, an online classifier is trained during tracking and the target is localized by trying to find a boundary between the object and the background as accurate as possible. Early work includes support vector machines (SVM) [12], Adaboost [13] and multiple-instance learning (MIL) [14] which require to set the class label to each training sample. Kalal et al. [40] presented positive-negative learning process where samples are labeled via classifier using structural constraints. Saffari et al. [41] proposed random forest algorithm combined with on-line bagging, random feature selection and an on-line tree building approach. Wu et al. [42] tracked using the image alignment method. Their approach seeks for an optimal image transformation such that the transformed image can be decomposed as the sum of a sparse error and a linear decomposition of well-aligned basis set by solving a sequence of convex optimization problems. Recently excellent performance was demonstrated by structured SVMs [15]. The crucial difference to standard classifiers like SVM and Adaboost is that the training labels are no longer hard-assigned to positive and negative classes, but are rather continuous values between zero and one. Bolme et al. [16] proposed regression-based discriminative model resulting in discriminativelly trained correlation filters that minimize the output sum of squared errors. These filters have been recently extended to multivariate features e.g., color [43], HoG [44], outperforming the related approaches on various benchmarks. Henriques et al. [4] combined correlation filters with kernels to achieve more accurate object localization. This work was extended by Li et al. [17] to address scale

adaptation. Danelljan et al. presented a correlation filter-based tracker with the ability to accurate address scale change problem [5]. Zhang et al. [18] adapted the correlation filters to spatio-temporal context learning.

2.2 Part-based trackers

Part-based models are constellations of (generative or discriminative) holistic models and vary in modeling the constellation geometric properties. An example of part-based model is shown on Figure 2.1 (right). Since the target is modeled with more than one part, they usually successfully address the partial occlusion problem. When this happens, there are usually only a few parts under occlusion, but the majority of them is not. Part-based models are typically successful in handling the object deformations. But because of the larger number of free parameters, they usually do not achieve as accurate performance as holistic trackers.

Hoey [19] proposed a flock-of-features tracking in which parts are independently tracked by optical flow. The flock is kept on target by identifying parts that deviate too far from the flock and replacing them with new ones. But because of weak geometric constraints, tracking is prone to drifting. Martinez et al. [20] proposed connecting triplets of parts and tracked them by kernels while enforcing locally-affine deformations. The local connectivity resulted in inefficient optimization and parts required careful manual initialization. Artner et al. [30] proposed a key-point-based tracker with a fully-connected constellation. They use a geometric model that enforces preservation of inter-keypoint distance ratios. Because the ratios are not updated during tracking and due to the ad-hoc combination of geometric and visual models, the resulting optimization is quite brittle, requiring manual initialization of parts and the resulting tracker handles only moderate locally-affine deformations. Pernici et al. [45] address nonrigid deformations by oversampling key-points, apply multiple instances of target models and use a similarity transform for matching. But, the tracker still fails at significant nonrigid deformations.

Several works simplify a geometric model to a star-based topology in interest of simplified optimization. A number of these works apply part detectors and a generalized Hough transform for localization. Examples of part detectors are key-points [29], random forest classifiers [33], ferns [46] and pixels [47]. Cai et al. [31] apply superpixels as parts combined with segmentation for efficient tracking, but the high reliability on color results in significant failures during illumination changes. Kwon et al. [28] apply generative models in a star-based topology with adding and removing parts and Cehovin et al. [21] increase the power of the geometric model by local connectivity. Both approaches require efficient stochastic optimizers for inference. Yao et al. [32] address the visual and geometric model within a single discriminative framework. They extend the structured SVM [15] to multiple part tracking, but cannot handle scale changes. This model was extended by Zhu et al. [34] to account for context as well, but uses a star-based topology for making the inference tractable.

In contrast to these approaches, we treat the visual and geometric deformations within a generative framework of a single spring system with fully-connected mid-level parts. The resulting model naturally addresses the aspect and scale changes as well as occlusion, affords efficient optimization and runs in real-time.

Chapter 3

Correlation filters

The visual model of our tracker is based on the holistic correlation filter trackers [4]. In the recent benchmarks [3, 1], these trackers have achieved a state-of-the-art performance, while running faster than real-time. In this chapter we give full exposition of this important class of trackers.

Let $\mathbf{I}$ be an image in which the target is to be localized and let $\mathbf{h}$ be an appropriate filter. The localization is performed via convolution, i.e.,

$$\mathbf{g} = \mathbf{h} * \mathbf{I}, \tag{3.1}$$

where $\mathbf{g}$ is the output of the same size as the input image, preferably zero everywhere, except at the location of the target (see Figure 3.1). The main challenge in the correlation filter tracking is then finding a filter $\mathbf{h}$, that produces a desired response $\mathbf{g}$ when convolved over an image $\mathbf{I}$. To address this challenge, the particular property of convolution under Fourier transform is used. The convolution of the two signals in spatial domain corresponds to the dot product of two signals in Fourier domain. Let $\hat{\mathbf{h}}$ be a Fourier transform of the signal $\mathbf{h}$, i.e.,

$$\hat{\mathbf{h}} = \mathcal{F}(\mathbf{h}). \tag{3.2}$$

In the following the symbol $\hat{(\cdot)}$ will be used to denote Fourier transform of a signal. The Fourier transform of the optimal filter is obtained by applying

Fourier transform to (3.1) and solving for $\hat{\mathbf{h}}$, i.e.,

$$\hat{\mathbf{h}}^* = \frac{\hat{\mathbf{g}}}{\hat{\mathbf{I}}}, \tag{3.3}$$

where division is performed element-wise and symbol $(\cdot)^*$ represents the complex-conjugate element. The desired (ideal) output $\mathbf{g}$ (also called labels) is typically defined as a Gaussian function (see Figure 3.1).

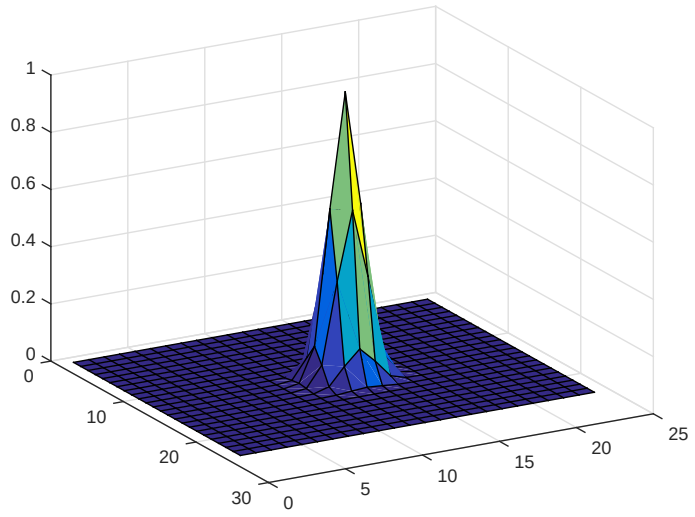

Figure 3.1: Visualization of Gaussian-shaped "labels" $\mathbf{g}$, of each output position. Correlation filter-based trackers typically model a filter, so that the response of the convolution between filter and current patch is as similar as possible to the Gaussian function on the image.

One of the first correlation filter-based trackers was presented by Bolme et al., called MOSSE tracker [16]. When creating a filter $\hat{\mathbf{h}}$, this tracker minimizes the output sum of squared errors defined as

$$\min_{\hat{\mathbf{h}}^*} \sum_i |\hat{\mathbf{I}}_i \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}_i|^2, \tag{3.4}$$

where i goes over the image elements and $\odot$ represents the element-wise multiplication.

Henriques et al., built on MOSSE [16] and presented kernelized correlation filter-based tracker (KCF) [4], which uses kernels for better target localization and multivariate features. The KCF tracker is an essential part of our proposed tracker and we detail it in Section 3.1. The KCF applies a powerful and fast descriptor called the histogram of oriented gradients (HoG), which is described in Section 3.2.

3.1 Tracking with kernelized correlation filters

This section summarizes the kernelized correlation filters (KCF) and draws heavily from the original paper [4]. The KCF takes advantage of the connection between linear regression and correlation filters using circulant matrices. This property allows tracker to train the model on all translated positions within the search region in real-time. A linear ridge regression is an extended linear regression with an additional term which reduces the overfitting in classification. Training is defined as finding a function $f(\mathbf{z}) = \mathbf{w}^T\mathbf{z}$ that minimizes the squared error over the samples $\mathbf{x}_i$ and regression targets y_i

$$\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i) - y_i)^2 + \lambda ||\mathbf{w}||^2. \tag{3.5}$$

Parameter λ is a regularization parameter that controls overfitting, similar as regularization in the support vector machines (SVM). Note that i-th sample $\mathbf{x}_i$ is a n-dimensional feature vector extracted from image. Minimization can be represented in a closed-form solution

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}, \tag{3.6}$$

where $\mathbf{X}$ is the data matrix with one sample per row $\mathbf{x}_i$, and $\mathbf{y}$ is the vector with regression target elements y_i. The identity matrix is represented by $\mathbf{I}$. Since the fourier domain operates with the complex numbers, Equation (3.6) can be written as

$$\mathbf{w}^* = (\mathbf{X}^H\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^H\mathbf{y}, \tag{3.7}$$

where $\mathbf{X}^H$ is the Hermitian transpose, $\mathbf{X}^H = (\mathbf{X}^*)^T$.

For understanding KCF fast implementation, terms of cyclic shifts and circulant matrices need to be clarified. A 1D signal of length d, $\mathbf{x} \in \mathcal{R}^{d \times 1}$, can be shifted by a single element via multiplication by a permutation matrix $\mathbf{P}$ defined as

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}. \tag{3.8}$$

Translation of the object in the image patch can be modeled with the product $\mathbf{Px}$. Larger translations can be modeled using the power matrix $\mathbf{P}^u\mathbf{x}$. The full set of shifted signals can be obtained with the equation

$$\{\mathbf{P}^u\mathbf{x} | u = 0, \dots, n-1\}. \tag{3.9}$$

An example of cycle-shifting the image is shown in Figure 3.2. Because of the larger shifts, the warp-around effect happens, which is shown in Figure 3.3.

Figure 3.2: Vertically shifted base sample [4].

Using cycling shifts from (3.9) for rows, the following circulant matrix is obtained

$$\mathbf{X} = C(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n \\ x_n & x_1 & x_2 & \dots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \dots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \dots & x_1 \end{bmatrix}. \tag{3.10}$$

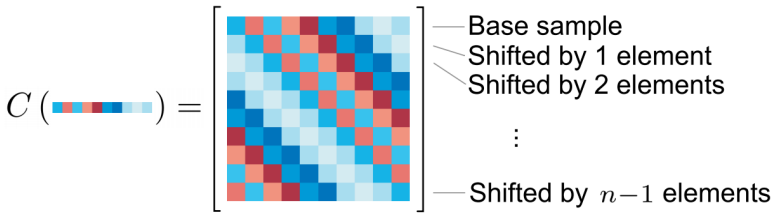

Figure 3.3: Each row of a circulant matrix represents a base sample (1D signal in the first row) translated for 1-more element than in the signal a row above it. Properties of circulant matrices can be used also on 2D images [4].

Matrix $\mathbf{X}$ is used to compute regression with shifted samples and since it is a circulant matrix, the diagonalization can be done by the Discrete Fourier Transform (DFT)

$$\mathbf{X} = \mathbf{F}^H \mathrm{diag}(\hat{\mathbf{x}})\mathbf{F}, \tag{3.11}$$

where $\mathbf{F}$ is the constant DFT matrix that does not depend on $\mathbf{x}$. Using properties of cycling shifts and circulant matrices, it was shown in [4] that regression from (3.7) can be written as

$$\hat{\mathbf{w}}^* = \mathrm{diag}\left(\frac{\hat{\mathbf{x}}^*}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda}\right)\hat{\mathbf{y}} = \frac{\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}}}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda}. \tag{3.12}$$

For improved performance, the KCF tracker applies a non-linear regression. The linear ridge regression is transformed into a kernelized non-linear regression using "kernel-trick". The kernel trick is a well-known approach from machine learning that transforms a low dimensional feature into a high-dimensional feature space $\varphi(\mathbf{x})$. In fact, since all operations among the features are in form of a dot product, it suffices to transform the outputs of the dot products by kernels, thus avoiding explicit transformation of features [48]. In particular, the weight vector $\mathbf{w}$ is rewritten (in the dual feature space) as a linear combination

$$\hat{\mathbf{w}}^* = \sum_i \alpha_i \varphi(\mathbf{x}_i), \tag{3.13}$$

where $\boldsymbol{\alpha}$ is the term that is being optimized. The dot-product among the transformed features can be defined using kernel function as

$$\varphi^T(\mathbf{x})\varphi(\mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}'), \tag{3.14}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are original feature vectors and $\kappa(\cdot, \cdot)$ is the kernel function (e.g., Gaussian or polynomial). The dot-product between the pairs of samples is defined as an element in matrix $\mathbf{K}$,

$$K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j). \tag{3.15}$$

The regression function from (3.5) in dual feature space can be written as

$$f(\mathbf{z}) = \mathbf{w}^T\mathbf{z} = \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{z}, \mathbf{x}_i), \tag{3.16}$$

and its closed-form solution is given as

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}, \tag{3.17}$$

where $\mathbf{K}$ is the kernel matrix and $\boldsymbol{\alpha}$ is the vector of coefficients α_i that represents the solution in the dual space. Since $\mathbf{K}$ is circulant, it is possible to diagonalize (3.17) and represent it in the form

$$\hat{\boldsymbol{\alpha}}^* = \frac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda}, \tag{3.18}$$

where $\hat{\mathbf{k}}^{\mathbf{xx}}$ is the first row of the kernel matrix $\mathbf{K} = C(\mathbf{k}^{\mathbf{xx}})$. In general, vector $\mathbf{k}^{\mathbf{xx}'}$ can be obtained with a *kernel correlation*. For two vectors $\mathbf{x}$ and $\mathbf{x}'$, the kernel correlation is defined as

$$k_i^{\mathbf{xx}'} = \kappa(\mathbf{x}, \mathbf{P}^{i-1}\mathbf{x}'). \tag{3.19}$$

The training in the KCF tracker is defined with the so-called *kernel auto-correlation* and is given by

$$\hat{\boldsymbol{\alpha}}^* = \frac{\hat{\mathbf{y}}}{\mathcal{F}(exp(-\frac{1}{\sigma^2}(||\mathbf{x}||^2 + ||\mathbf{x}||^2 - 2\mathcal{F}^{-1}(\sum_c \hat{\mathbf{x}}_c \odot \hat{\mathbf{x}}_c^*)))) + \lambda}, \tag{3.20}$$

where $\mathbf{x}_c$ is the c-th channel of the training image (or feature) patch.

Detection of the object in a new image is defined by

$$\hat{f}(\mathbf{z}) = (\hat{\mathbf{k}}^{\mathbf{xz}})^* \odot \hat{\boldsymbol{\alpha}}, \tag{3.21}$$

where $\mathbf{z}$ is the image patch extracted from current frame at the position of the object in the previous frame. Equation (3.21) can be rewritten into the full form, getting

$$\hat{f}(\mathbf{z}) = (\mathcal{F}(exp(-\frac{1}{\sigma^2}(||\mathbf{x}||^2 + ||\mathbf{z}||^2 - 2\mathcal{F}^{-1}(\sum_c \hat{\mathbf{x}}_\mathbf{c} \odot \hat{\mathbf{z}}_\mathbf{c}^*)))))^* \odot \hat{\boldsymbol{\alpha}}. \tag{3.22}$$

Algorithm 1 : Main Matlab functions of the KCF tracker used for training and tracking, appropriate for multi-channel features.

Input:

x: training image patch ($m \times n \times c$),

y: regression target, Gaussian-shaped ($m \times n$),

z: test image patch ($m \times n \times c$)

Output:

`responses`: detection scores for each location ($m \times n$)

```
function alphaf = train(x, y, sigma, lambda)
        k = kernel_correlation(x, x, sigma);
        alphaf = fft2(y) ./ (fft2(k) + lambda);
end
function responses = detect(alphaf, x, z, sigma)
        k = kernel_correlation(x, z, sigma);
        responses = real(ifft2(alphaf .* conj(fft2(k))));
end
function k = kernel_correlation(x1, x2, sigma)
        c = ifft2(sum(fft2(x1) .* conj(fft2(x2)), 3));
        d = x1(:)'*x1(:) + x2(:)'*x2(:) - 2*c;
        k = exp(-1/sigma^2 * abs(d) / numel(d));
end
```

For tracking in time-step t, the position $\mathbf{p}_{t-1}$ of an object at time-step $t-1$ has to be given. The target model is represented as $\hat{\mathbf{x}}_{t-1}$ and $\hat{\boldsymbol{\alpha}}_{t-1}$. The result of tracking is a new position of the object $\mathbf{p}_t$, as well as updated $\hat{\mathbf{x}}_t$ and $\hat{\boldsymbol{\alpha}}_t$. The Algorithm 2 summarizes the tracking process at the time-step t on the image $\mathbf{I}_t$, using functions from Algorithm 1.

Algorithm 2 : Localization of the target with the KCF tracker.

Require:

Image $\mathbf{I}_t$, position of the object $\mathbf{p}_{t-1}$ at the time-step $t-1$ and target representations $\hat{\mathbf{x}}_{t-1}$ and $\hat{\boldsymbol{\alpha}}_{t-1}$.

Ensure:

New position of the object $\mathbf{p}_t$ at the current time-step t and updated $\hat{\mathbf{x}}_t$ and $\hat{\boldsymbol{\alpha}}_t$.

Procedure:

1: Extract image patch $\mathbf{z}_t$ from position $\mathbf{p}_{t-1}$ in the image $\mathbf{I}_t$.

2: Extract features (e.g., HoG, color, grayscale, etc.) from the image patch $\mathbf{z}_t$, multiply it with the cosine window and convert it to fourier domain, obtaining $\hat{\mathbf{x}}$.

3: Calculate cross correlation using function `detect`. Finding the position of cross correlation maximum response results in a position of the object $\mathbf{p}_t$ in the current frame.

4: Repeat steps 1 and 2 on the position $\mathbf{p}_t$, obtaining new $\hat{\mathbf{x}}$.

5: Calculate the new alpha $\hat{\boldsymbol{\alpha}}$ according to (3.20), using function `train`. Note that $\hat{\mathbf{y}}$ in (3.20) represents the target variable in Fourier domain (e.g., Gaussian-shaped labels which are shown in Figure 3.1).

6: Update $\hat{\mathbf{x}}_t$ and $\hat{\boldsymbol{\alpha}}_t$ using $\hat{\mathbf{x}}$ and $\hat{\boldsymbol{\alpha}}$ with a running average

$$\begin{aligned} \hat{\mathbf{x}}_t &= (1-\eta)\hat{\mathbf{x}}_{t-1} + \eta\hat{\mathbf{x}}, \\ \hat{\boldsymbol{\alpha}}_t &= (1-\eta)\hat{\boldsymbol{\alpha}}_{t-1} + \eta\hat{\boldsymbol{\alpha}}, \end{aligned} \tag{3.23}$$

where η is a learning rate parameter.

3.1.1 Improved KCF tracker

The KCF described so far follows the details from [4], but the authors made three improvements for the VOT2014 Challenge [3] and they are described in the following sections.

Scale adaptation

The scale adaptation problem is addressed by calculating the response on multiple scales. Steps 1-3 in Algorithm 2 are repeated on a current scale and also on one smaller and one larger scale. The scale with the highest maximum response obtained with the function `detect` from Algorithm 1 is chosen as the optimal scale.

Tracker updating

The update phase is modified so that numerator and denominator of the $\hat{\boldsymbol{\alpha}}^*$ in (3.18) or (3.20) are learned separately. Equation for $\hat{\boldsymbol{\alpha}}^*$ can be rewritten into the form

$$\begin{aligned} \hat{\boldsymbol{\alpha}}^* &= \frac{\mathbf{A}}{\mathbf{B}}, \\ \mathbf{A} &= \hat{\mathbf{y}}, \\ \mathbf{B} &= \mathcal{F}(exp(-\frac{1}{\sigma^2}(||\mathbf{x}||^2 + ||\mathbf{x}||^2 - 2\mathcal{F}^{-1}(\sum_c \hat{\mathbf{x}}_\mathbf{c} \odot \hat{\mathbf{x}}_\mathbf{c}^*)))) + \lambda. \end{aligned} \tag{3.24}$$

Taking into account also Gaussian kernel function, (3.24) is rewritten into

$$\begin{aligned} k &= \mathcal{F}(exp(-\frac{1}{\sigma^2}(||\mathbf{x}||^2 + ||\mathbf{x}||^2 - 2\mathcal{F}^{-1}(\sum_c \hat{\mathbf{x}}_\mathbf{c} \odot \hat{\mathbf{x}}_\mathbf{c}^*)))), \\ \mathbf{A} &= \hat{\mathbf{y}} \odot k, \\ \mathbf{B} &= k \odot (k + \lambda). \end{aligned} \tag{3.25}$$

Update step is done in the same way as in (3.23) – both factors are updated with a running average.

Sub-cell peak estimation

The third improvement is sub-cell peak estimation [49] of the response map. In each direction (horizontal and vertical) three pixels values of the response around the maximum are extracted. Given the maximum pixel value p_0, two neighbouring pixel values are denoted p_- and p_+ (see Figure 3.4). Thus given three pixel values, the interpolated peak location Δp is obtained at the maximum of the fitted parabola

$$\Delta p = \frac{p_+ - p_-}{4p_0 - 2(p_+ + p_-)}. \tag{3.26}$$

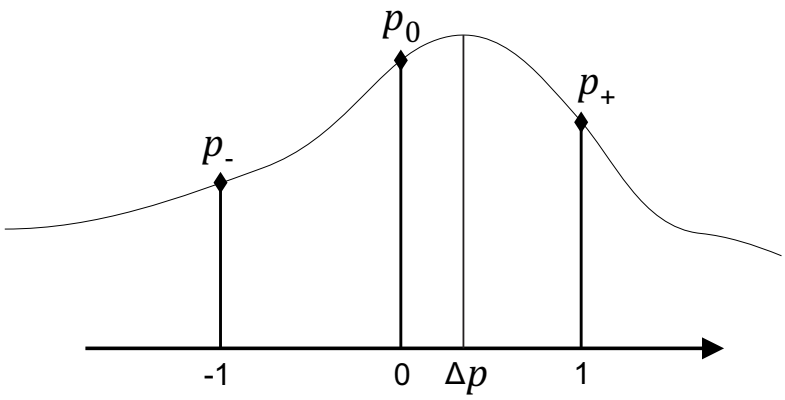

Figure 3.4: Estimation of the Δp, position of a peak of the curve fitted on the three points.

3.2 Histogram of oriented gradients

Histogram of oriented gradients (HoG) is a descriptor developed by Dalal and Triggs [44], which won the PASCAL object detection challenge in 2006. Felzenswalb et al. [50] increased performance of HoG descriptor by increasing its speed and descripting power. They proposed a deformable parts model for object detection with the ability of handling object deformation and multi-scale changes of the object. Even though HoG was originally developed for object detection problems, it can be used as a powerful descriptor for object tracking. Calculation of the HoG descriptor is summarized by [50, 44] and it can be divided into three steps, described in the following sections.

3.2.1 Pixel-level feature map

HoG descriptor is based on the image gradients which are efficiently computed using finite difference filters $[-1, 0, 1]$ and $[-1, 0, 1]^T$. For color images, the color channel with the largest magnitude is used to compute orientation $\theta(x, y)$ and magnitude $r(x, y)$ at a pixel (x, y). The gradient orientation is at each pixel discretized into one of the q values using either a contrast sensitive (B_1) or contrast insensitive (B_2) definition,

$$B_1(x, y) = \text{round}(\frac{q\theta(x, y)}{2\pi}) \mod q, \tag{3.27}$$

$$B_2(x, y) = \text{round}(\frac{q\theta(x, y)}{\pi}) \mod q. \tag{3.28}$$

Given $b \in \{0, \ldots, q-1\}$ representing q orientation bins, a pixel-level feature map that specifies a sparse histogram of gradient magnitudes at (x, y) is defined as

$$F(x, y) = \begin{cases} r(x, y) & \text{if } b = B(x, y) \\ 0 & \text{otherwise.} \end{cases} \tag{3.29}$$

Orientation edge map $\mathbf{F}$ can be represented as a $w \times h \times q$ matrix, where w and h are dimensions of an original image.

3.2.2 Spatial aggregation

Given a pixel-level feature map $\mathbf{F}$ for a $w \times h$ image and parameter $k > 0$, specifying the side length of a square image region, a dense grid of rectangular "cells" is defined. Pixel-level features are aggregated to obtain a cell-based feature map $\mathbf{C}$, with feature vectors $C(i,j)$ for $0 \leq i \leq \lfloor (w-1)/k \rfloor$ and $0 \leq j \leq \lfloor (h-1)/k \rfloor$. This aggregation provides some invariance to small deformations and reduces the size of feature map.

Mapping each pixel to a cell is achieved by "soft binning" presented in [44], where each pixel contributes to the feature vectors in the four cells around it using bilinear interpolation.

3.2.3 Normalization and truncation

Gradients are invariant to changes in bias and this invariance can be achieved via normalization. As in [44], four different normalization factors for the feature vector $C(i,j)$ are used. These normalization factors $N_{\delta,\gamma}(i,j)$ where $\delta, \gamma \in \{-1, 1\}$ are

$$N_{\delta,\gamma}(i,j) = (||C(i,j)||^2 + ||C(i+\delta,j)||^2 + ||C(i,j+\gamma)||^2 + ||C(i+\delta,j+\gamma)||^2)^{\frac{1}{2}}. \tag{3.30}$$

Each factor measures the "gradient energy" in a square block of four cells containing (i,j).

The HoG feature map is obtained by concatenating the result of normalizing the cell-based feature map $\mathbf{C}$ with respect to each normalization factor followed by truncation,

$$H(i,j) = \begin{pmatrix} T_\alpha(C(i,j)/N_{-1,-1}(i,j)) \\ T_\alpha(C(i,j)/N_{+1,-1}(i,j)) \\ T_\alpha(C(i,j)/N_{+1,+1}(i,j)) \\ T_\alpha(C(i,j)/N_{-1,+1}(i,j)) \end{pmatrix}, \tag{3.31}$$

where $T_\alpha(v)$ denotes the component-wise truncation of a vector v by α (the i-th entry in $T_\alpha(v)$ is the minimum of the i-th entry of v and α). HoG features are commonly using the following parameters:

- contrast insensitive gradient orientations, $q = 9$, discretized with B_2, Equation (3.28),
- cell size, $k = 8$,
- truncation, $\alpha = 0.2$.

This setting leads to a 36-dimensional feature vector.

Chapter 4

Deformable parts tracker

In this chapter our tracking method, layered deformable part tracker (LDP) is presented. The tracker output at time-step t is an axis-aligned bounding box $\mathbf{x}_t^{(0)}$, which is estimated by integrating the coarse and mid-level object representations. The coarse representation is explained in Section 4.1, the mid-level representation in Section 4.2 and the resulting tracking algorithm is presented in Section 4.3. Step-by-step tracking and parts of the LDP tracker are visualized in Figure 4.1. The tracker localizes the target with the coarse representation (step 1) and initializes parts on mid-level representation (step 2). Then the spring system on mid-level representation is formed and optimized to fine localization of the target (steps 3-4). All parts on both layers are updated next (steps 5-6) and position of the object is reported.

4.1 Coarse representation

The coarse representation consists of two high-level object models: the object global template $I_t = \{\mathbf{z}_t^{(0)}\}$ and a global color model $C_t = \{p(\mathbf{x}_t|f), p(\mathbf{x}_t|b)\}$, specified by the foreground and background color histograms, $p(\mathbf{x}_t|f)$ and $p(\mathbf{x}_t|b)$. These models are used in each tracking iteration to coarsely estimate the center of the target bounding box $\mathbf{x}_t^{(0)}$ within a specified search region (Figure 1.2, step 1), which is subsequently refined by the mid-level rep-

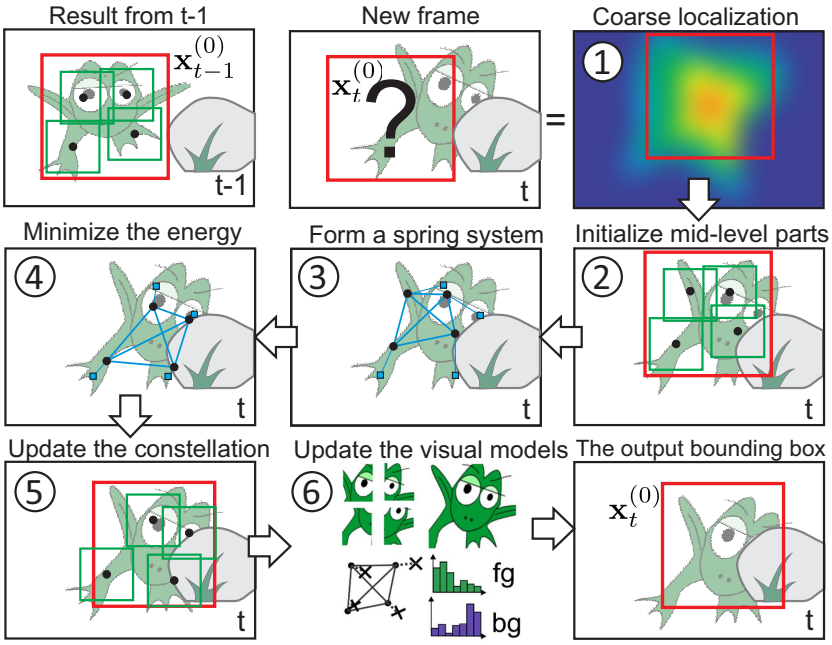

Figure 4.1: Visualization of tracking with LDP tracker.

resentation. Given a search region (Figure 4.2a), the center is estimated by maximizing the density $p(\mathbf{x}_t^{(0)}|I_t, C_t) \propto p(I_t|\mathbf{x}_t^{(0)})p(C_t|\mathbf{x}_t^{(0)})$. The first term, $p(I_t|\mathbf{x}_t^{(0)})$, is the template probability reflecting the similarity between the patch centered at $\mathbf{x}_t^{(0)}$ and the target template $\mathbf{z}_t^{(0)}$ (Figure 4.2b). The second term is the color probability defined as $p(C_t|\mathbf{x}_t^{(0)}) = p(f|\mathbf{x}_t)(1 - \alpha_{\text{col}}) + \alpha_{\text{col}}$, where $p(f|\mathbf{x}_t)$ is the probability of a pixel belonging to a foreground and α_{col} is a weak uniform distribution to address sudden temporal changes of the object color, since the $p(f|\mathbf{x}_t)$ might be uninformative in these situations and would deteriorate localization. The probability of a pixel belonging to a foreground, $p(f|\mathbf{x}_t)$, is calculated by histogram backprojection, i.e., by applying the Bayes rule with $p(\mathbf{x}_t|f)$ and $p(\mathbf{x}_t|b)$, and regularized by a Markov random field [51] to arrive at a smoothed foreground posterior (Figure 4.2c). Multiplying the template and color probabilities yields the density $p(\mathbf{x}_t^{(0)}|I_t, C_t)$ (Figure 4.2d). Note that on their own, the template and color result in ambiguous densities but their combination drastically reduces the ambiguity. In

the following we briefly overview the color segmentation model.

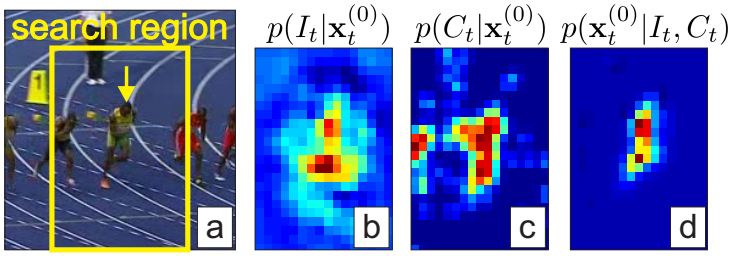

Figure 4.2: Example of a search region and the tracked object indicated by a rectangle and an arrow, respectively (a). The coarse template probability, the color probability and the full coarse model density are shown in (b), (c) and (d), respectively.

4.1.1 The color segmentation model

One of the components on the coarse representation is the color probability map (Figure 4.2c). It is obtained via an object segmentation using foreground and background object probabilities (3-dimensional color histograms), $\mathbf{h}_t^{(f)}$ and $\mathbf{h}_t^{(b)}$, which are learned during the tracking. Pixels in probability map are modeled using Markov random field (MRF) method, presented by Kristan et al. [52] shown in Figure 4.3.

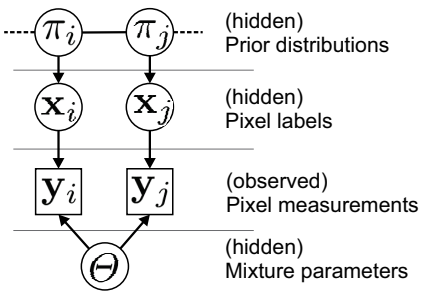

Figure 4.3: Graphical model of Markov random field with hidden and observed layers.

Mixture parameters in the MRF in Figure 4.3 consist of two histograms – foreground and background, i.e., $\Theta = \{\mathbf{h}_t^{(f)}, \mathbf{h}_t^{(b)}\}$. Pixel measurement in an image with M pixels is denoted as $\mathbf{y}_i$ and it represents a d-dimensional feature vector of i-th pixel. The i-th pixel label x_i is an unobserved random variable governed by the class prior distribution $\pi_i = [\pi_{i1}, \pi_{i2}]$, where $\pi_{il} = p(x_i = l)$. Since our model consists of two components – foreground and background, the class prior distribution π_i has two elements. The MRF from Figure 4.3 is solved, i.e., the per-pixel posteriors are estimated, by minimizing cost function defined as

$$C = \sum_{i=1}^{M} \log p(\mathbf{y}_i, \Theta) - \frac{1}{2}(E(\pi_i, \pi_{N_i}) + E(\mathbf{p}_i, \mathbf{p}_{N_i})). \tag{4.1}$$

The first term on the right-hand side of (4.1) represents the joint probability that is obtained via histogram back-projection defined as

$$p(\mathbf{y}_i, \Theta) = \sum_{k=1}^{2} p(\mathbf{y}_i | x_i = k)\pi_{ik}. \tag{4.2}$$

Note that the k runs over two components – foreground and background, therefore the term $x_i = 1$ represents the foreground label and $x_i = 2$ represents the background label. Using Bayes rule, foreground probability can be rewritten into

$$p(x_i = 1|\mathbf{y}_i) = \frac{p(\mathbf{y}_i|x_i = 1)p(x_i = 1)}{p(\mathbf{y}_i|x_i = 1)p(x_i = 1) + p(\mathbf{y}_i|x_i = 2)p(x_i = 2)}. \tag{4.3}$$

The probability distributions $p(\mathbf{y}_i|x_i = 1)$ and $p(\mathbf{y}_i|x_i = 2)$ are modeled by the foreground and background histograms, $\mathbf{h}_t^{(f)}$ and $\mathbf{h}_t^{(b)}$, respectively. The background prior probability $p(x_i = 2)$ is calculated as

$$p(x_i = 2) = \frac{S_b}{S_f}, \tag{4.4}$$

where S_f represents the size of the target (foreground) and S_b is the size of the search region (background) within which the segmentation is performed. The foreground prior is therefore defined as $p(x_i = 1) = 1 - p(x_i = 2)$.

The second term on the right-hand side of (4.1) is a local-consistency term consisting of two energy functions $E(\cdot,\cdot)$. The mixture distribution over the priors of the i-th pixel neighbors N_i is defined as

$$\pi_{N_i} = \sum_{j \in N_i, j \neq i} \lambda_{ij} \pi_j, \tag{4.5}$$

where λ_{ij} are fixed positive weights such as the one for i-th pixel $\sum_j \lambda_{ij} = 1$. The segmentation smoothness is further enforced by placing another MRF over the posteriors. In this respect, the posteriors, like priors, are treated as random variables $\mathbf{P} = \{\mathbf{p}_i\}_{i=1:M}$, where the components of $\mathbf{p}_i$ are defined as

$$p_{ik} = p(x_i = k|\Theta, \mathbf{y}_i), \tag{4.6}$$

and can be computed by the Bayes rule from $p(y_i|x_i = k, \Theta)$ and $p(x_i = k)$. The term $\mathbf{p}_{N_i}$ is a mixture defined in the same spirit as π_{N_i}. The smoothness term in the energy function (4.1) is defined as

$$E(\pi_i, \pi_{N_i}) = D(\pi_i||\pi_{N_i}) + H(\pi_i). \tag{4.7}$$

The term $D(\pi_i||\pi_{N_i})$ is Kullback-Leibler divergence which penalizes the difference between prior distributions over neighboring pixels (π_i and π_{N_i}), defined as

$$D(\pi_i||\pi_{N_i}) = \sum_{k=1}^{2} \pi_{ik} \ln \frac{\pi_{ik}}{\pi_{N_{ik}}}. \tag{4.8}$$

The term $H(\pi_i)$ is entropy defined as

$$H(\pi_i) = -\sum_{k=1}^{2} \pi_{ik} \log \pi_{ik}, \tag{4.9}$$

which penalizes uninformative priors π_i. The term enforcing the smoothness of posteriors in (4.1), i.e., $E(\mathbf{p}_i, \mathbf{p}_{N_i})$, is defined in the same manner as $E(\pi_i, \pi_{N_i})$.

In our approach, the segmentation is performed by optimizing the MRF, i.e., minimizing the cost (4.1), over the posteriors using the fast convolution-based approach as in Kristan et al. [52].

4.2 Mid-level representation

The mid-level representation is a geometrically constrained constellation of N_p parts $\mathbf{X}_t = \{\mathbf{x}_t^{(i)}\}_{i=1:N_p}$. Each part $\mathbf{x}_t^{(i)}$ is a local mid-level representation of target and is specified by the part visual model (template) $\mathbf{z}_t^{(i)}$. The probability of a constellation being at state $\mathbf{X}_t$ conditioned on the measurements $\mathbf{Y}_t$ and parameters of the deformation model Θ is decomposed into

$$p(\mathbf{X}_t|\mathbf{Y}_t, \Theta) \propto p(\mathbf{Y}_t|\mathbf{X}_t, \Theta)p(\mathbf{X}_t|\Theta). \tag{4.10}$$

The density $p(\mathbf{Y}_t|\mathbf{X}_t, \Theta)$ is the *measurement constraint* term, reflecting the agreement of measurements with the current state $\mathbf{X}_t$ of constellation, whereas the second term, $p(\mathbf{X}_t|\Theta)$, reflects the agreement of the constellation with the *geometric constraints*.

4.2.1 Geometric constraints

The constellation is specified by a set of links $(i, j) \in \mathcal{L}$ indexing the connected pairs of parts (Figure 4.4). The parts and links form an undirected graph and the joint pdf over the part states can be factored over the links as

$$p(\mathbf{X}_t|\Theta) = \prod_{(i,j)\in\mathcal{L}} \phi(||d_t^{(i,j)}||; \mu^{(i,j)}, k^{(i,j)}), \tag{4.11}$$

where $d_t^{(i,j)} = \mathbf{x}_t^{(i)} - \mathbf{x}_t^{(j)}$ is a difference in positions of the linked parts, $\mu^{(i,j)}$ is the preferred distance between the pair of parts and $k^{(i,j)}$ is the intensity of this constraint. The factors in (4.11) are defined as Gaussians $\phi(\cdot; \mu, k)$ with mean μ and variance k to reflect the property that deviations from the preferred distances should decrease the probability (4.11).

4.2.2 Measurement constraints

Given a fixed part state, $\mathbf{x}_t^{(i)}$, the measurement at that part is independent from the states of other parts and the measurement probability decomposes into a product of per-part visual likelihoods

$$p(\mathbf{Y}_t|\mathbf{X}_t,\Theta) = \prod\nolimits_{i=1:N_p} p(\mathbf{y}_t^{(i)}|\mathbf{x}_t^{(i)},\Theta). \tag{4.12}$$

The visual likelihoods are chosen from the same family of pdfs as factors in (4.11) for tractability. Let $\mathbf{x}_{tA}^{(i)}$ be the position in vicinity of $\mathbf{x}_t^{(i)}$ that maximizes the similarity of the visual model $\mathbf{z}_t^{(i)}$ and the measurement $\mathbf{y}_t^{(i)}$ (see Figure 4.4, left). The visual likelihood can then be defined as a Gaussian $p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)},\Theta) = \phi(||d_t^{(i)}||;0,k^{(i)})$ where $d_t^{(i)} = \mathbf{x}_t^{(i)} - \mathbf{x}_{tA}^{(i)}$ is the difference of the part current state and its visually-ideal position, and $k^{(i)}$ is the intensity of this constraint. The visual likelihood thus reflects the agreement of the visual model $\mathbf{z}^{(i)}$ with the measurement $\mathbf{y}^{(i)}$ at position $\mathbf{x}_t^{(i)}$. Since the parts are of mid-size and the part search region is relatively small compared to the size of the target, the local Gaussian approximation of the likelihood is a valid choice. As pointed out in Section 5.1, the visual models $\mathbf{z}_t^{(i)}$ are in fact filters discriminatively regressed to a Gaussian response function.

4.2.3 Collecting constraints

Equations (4.11,4.12) lead to an exponential posterior $p(\mathbf{X}_t|\mathbf{Y}_t,\Theta) \propto \exp(-E)$, with

$$E = \frac{1}{2}\sum_{i=1:N_p} k_t^{(i)} \left\| d_t^{(i)} \right\|^2 + \sum_{i,j\in\mathcal{L}} k_t^{(i,j)} (\mu_t^{(i,j)} - \left\| d_t^{(i,j)} \right\|)^2. \tag{4.13}$$

Note that E corresponds to an energy of a spring system in which pairs of parts are connected by springs and each part is connected by another spring to an image position most similar to the part visual model (Figure 4.4, right). The terms $\mu^{(i,j)}$ and $k^{(i,j)}$ are nominal lengths and stiffness of springs interconnecting parts (dynamic springs), while $k^{(i)}$ is stiffness of the spring connecting part to the image location (static spring). This spring system is a dual representation of the deformable parts model and minimization of its

(convex) energy corresponds to the maximum a posteriori state estimation of (4.10). Several solvers for spring systems exist from physics. We tested two optimization methods e.g., conjugated gradient descent and direct method. Experimental comparison of both methods is given in Section 5.3.1.

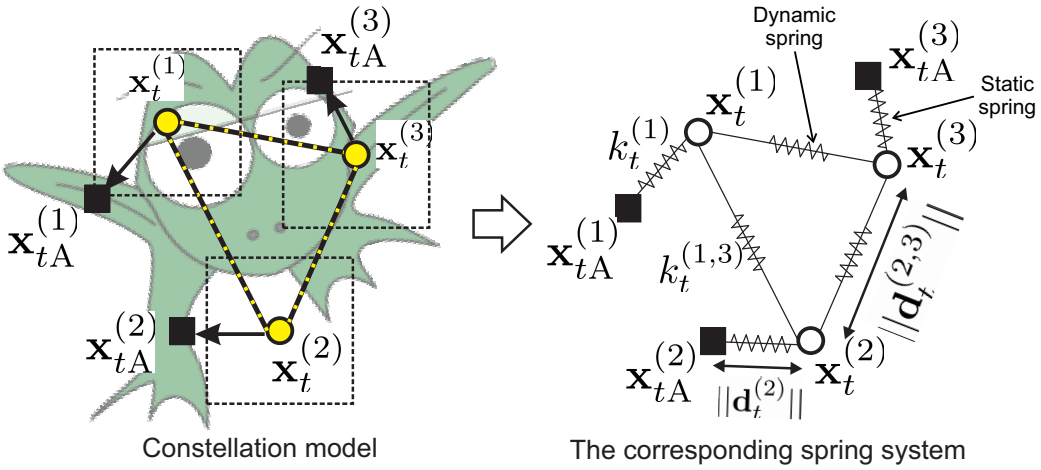

Figure 4.4: Example of a constellation model with rectangular parts and arrows pointing to the most visually similar positions (left) and the dual form corresponding to a spring system (right).

4.2.4 Conversion of DPM to a spring system

The deformable parts model from previous section can be converted into a spring system as follows. The main challenge is to define the stiffness of each spring in a system.

The stiffness $k_t^{(i)}$ of a spring connecting a part to the image (in Figure 4.4 denoted as *static spring*[1]) should reflect the uncertainty of the visually most similar location estimate $\mathbf{x}_{tA}^{(i)}$ in the search region of the i-th part. The visual model $\mathbf{z}_t^{(i)}$ is evaluated at all displacements in the search region, producing

[1]Note that *static spring* does not mean actual static spring with fixed length. Name static comes from term *static node* which represents the location $\mathbf{x}_{tA}^{(i)}$ and it does not change its position during the optimization.

the similarity map. The maximum on the map, $w_t^{(i)}$, is obtained at the location $\mathbf{x}_{tA}^{(i)}$. The spatial uncertainty in the search region is estimated as the weighted variance $\sigma_t^{2(i)}$, i.e., the expected squared distance from $\mathbf{x}_{tA}^{(i)}$. The spring stiffness is a product of the response $w_t^{(i)}$ and spatial uncertainty, i.e.,

$$k_t^{(i)} = w_t^{(i)} / \sigma_t^{2(i)}. \tag{4.14}$$

The stiffness of springs interconnecting the parts (in Figure 4.4 denoted as *dynamic spring*) depends on the allowed constellation size change between consecutive time-steps and part location uncertainty. The stiffness of a spring connecting parts indexed by (i, j) is defined as

$$k_t^{(i,j)} = w_t^{(i,j)} / (\mu_{t-1}^{(i,j)} \alpha_{\text{scl}})^2, \tag{4.15}$$

where $\mu_t^{(i,j)}$ is the preferred part distance estimated from the previous time-step, α_{scl} is the allowed scale change factor and $w_t^{(i,j)}$ is the average weight of the two parts.

4.2.5 Spring system optimization

The spring system is defined as a two-dimensional problem with two different types of nodes. Geometric constraints from (4.11) are represented with the dynamic nodes. They change their positions during the optimization to minimize the energy of the spring system. Measurement constraints from (4.12) are represented with the static nodes. They are called also anchors, because they do not change the position during the optimization. There is no closed-form solution for a multi-dimensional spring system. But the energy of the system is convex and standard iterative optimization approaches can be applied. Two optimization methods are considered. The first is the conjugated gradient descend [53, 54], which is a common approach for low-complexity efficient nonlinear optimization (Section 4.2.5). However, our cost function has a particular form and we propose a specialized optimization tailored for the spring-system cost functions in Section 4.2.5.

Conjugated gradient descent

Conjuated gradient descend (CGD) optimization requires gradient of the energy function from Equation (4.13) and it is defined w.r.t. $\mathbf{x}_t^{(i)}$ as

$$\frac{\partial E}{\partial \mathbf{x}_t^{(i)}} = -k_t^{(i)} d_t^{(i)} - \sum_{i,j \in \mathcal{L}(i)} (\mu_t^{(i,j)} - \left\| d_t^{(i,j)} \right\|) \frac{k_t^{(i,j)} d_t^{(i,j)}}{\left\| d_t^{(i,j)} \right\|}, \tag{4.16}$$

where $\mathcal{L}(i)$ denotes the links connecting part i with the neighboring parts via dynamic springs, while the first term refers to a static spring of i-th part. The terms of Equation (4.16) are described in Section 4.2. Conjugated gradient descend is an iterative optimization method which is, in contrast to gradient descend, able to automatically adjust the length of a step in the direction of gradient in each iteration of the optimization. This ensures much faster convergence and more robust performance, since the length of the step does not need to be set in advance. The CGD method is described in Algorithm 3, where function $f(\mathbf{x})$ represents spring system energy function (4.13) and $\nabla f(\mathbf{x})$ is its derivative, defined in (4.16). Note that the term $\mathbf{Q}(\mathbf{x})$ represents the Hessian matrix of a function $f(\mathbf{x})$.

Direct optimization

In this section a novel optimization method called iterative direct approach (IDA) is presented. The optimization iteratively minimizes the energy of a spring system by decomposing the 2D spring system into two 1D independent systems (Figure 4.5), minimizing the separate energies in a closed form and re-composing the 2D spring system. This procedure is iterated until convergence. In the following, an efficient algebraic closed-form solution is derived.

Algorithm 3 : Conjugate gradient descend optimization.

Require:

Approximation of the solution $\mathbf{x}_0$, function $f(\mathbf{x})$ and its gradient $\nabla f(\mathbf{x})$.

Ensure:

Solution $\mathbf{x}$ that minimizes the function $f(\mathbf{x})$.

Procedure:

Set $i = 0$, $\mathbf{p}_{-1} = 0$, $\beta_0 = 0$, $\mathbf{d}_{-1} = 0$, and set stopping criterion ϵ.
while $||\nabla f(\mathbf{x}_i)|| > \epsilon$ **do**
  if $i > 0$ **then**
    $\beta_i = \frac{\nabla f(\mathbf{x}_i)^T \nabla f(\mathbf{x}_i)}{\nabla f(\mathbf{x}_{i-1})^T \nabla f(\mathbf{x}_{i-1})}$.
  end if
  $\mathbf{p}_i = -\nabla f(\mathbf{x}_i) + \beta_i \mathbf{p}_{i-1}$
  $\mathbf{d}_i = -\nabla f(\mathbf{x}_i) + \beta_{i-1} \mathbf{d}_{i-1}$
  $\alpha_i = -\frac{\nabla f(\mathbf{x}_i)^T \mathbf{d}_i}{\mathbf{d}_i^T \mathbf{Q}(\mathbf{x}_i) \mathbf{d}_i}$
  $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$
  $i = i + 1$
end while

Using standard Newtonian mechanics results, the relation between the forces at springs of a 1D spring system, the stiffness coefficients and spring expansions, can be written in a matrix form as

$$\mathbf{F}_{\text{springs}} = -\mathbf{K} \cdot \Delta\mathbf{L}, \tag{4.17}$$

where $\mathbf{K} = diag([k_1, \cdots, k_N])$ is a diagonal matrix of spring stiffness coefficients. The spring displacements vector $\Delta\mathbf{L}$ is defined as

$$\Delta\mathbf{L} = \mathbf{A}\mathbf{x} - \mathbf{L}, \tag{4.18}$$

where $\mathbf{x}$ is a vector of the 1D positions of the nodes and the current-lenght vector $\mathbf{L} = [l_1, \cdots, l_N]$, where element l_i represents the length of the i-th spring. Elements in the vector of positions $\mathbf{x}$ are arranged in the following form

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{\text{dyn}} \\ \mathbf{x}_{\text{stat}} \end{bmatrix}, \tag{4.19}$$

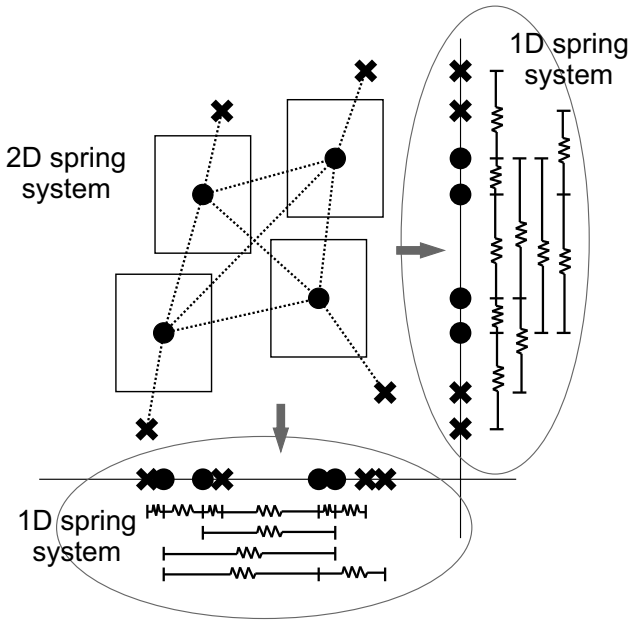

Figure 4.5: Example of decomposition of a 2D spring system with 4 dynamic (circles) and 4 static nodes (crosses) on two 1D spring systems. Each 1D spring system can be solved in a closed-form.

where $\mathbf{x}_{\text{dyn}}$ and $\mathbf{x}_{\text{stat}}$ are 1D positions of the dynamic and static nodes, respectively. The connectivity matrix $\mathbf{A}$ represents the directed connections between the nodes with the dimensionality $N_{\text{springs}} \times N_{\text{nodes}}$. If i-th spring connects a pair of nodes $\{n_{i1}, n_{i2}\}$, than the element a_{ij} of the matrix $\mathbf{A}$ is defined as

$$a_{ij} = \begin{cases} 1 & ; j \equiv n_{i1} \\ -1 & ; j \equiv n_{i2} \\ 0 & ; \text{otherwise} \end{cases} \tag{4.20}$$

Equation (4.17) can be rewritten in to the form $\mathbf{F}_{\text{springs}} = -\mathbf{KAx} + \mathbf{KL}$. The forces in the nodes are given by left-multiplying by $\mathbf{A}^T$, obtaining an equation

$$\mathbf{F}_{\text{nodes}} = -\mathbf{A}^T\mathbf{KAx} + \mathbf{A}^T\mathbf{KL}. \tag{4.21}$$

Equilibrium of the system is reached when forces in nodes equal to zero,

resulting in the following linear system

$$\mathbf{A}^T\mathbf{K}\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{K}\mathbf{L}. \tag{4.22}$$

For efficient calculation of $\mathbf{x}$, two more matrices are defined, $\widehat{\mathbf{K}} = \mathbf{A}^T\mathbf{K}\mathbf{A}$ and $\mathbf{C} = \mathbf{A}^T\mathbf{K}$. Since the vector of positions $\mathbf{x}$ is in the same form as in (4.19), the matrix $\widehat{\mathbf{K}}$ can be written into the form

$$\widehat{\mathbf{K}} = \left[\begin{array}{cc} \widehat{\mathbf{K}}_{\text{dyn}} & \widehat{\mathbf{K}}_{\text{stat}} \\ \hline & \widehat{\mathbf{K}}_{\text{rem}} \end{array}\right]. \tag{4.23}$$

Note that the remaining matrix $\widehat{\mathbf{K}}_{\text{rem}}$ is not used in the following. The matrix $\widehat{\mathbf{K}}$ has the dimensionality $N_{\text{nodes}} \times N_{\text{nodes}}$ and matrix $\widehat{\mathbf{K}}_{\text{dyn}}$ is defined as

$$\widehat{\mathbf{K}}_{\text{dyn}} = \begin{bmatrix} \widehat{k}_{1,1} & \cdots & \widehat{k}_{1,N_{\text{dyn}}} \\ \vdots & \cdots & \vdots \\ \widehat{k}_{N_{\text{dyn}},1} & \cdots & \widehat{k}_{N_{\text{dyn}},N_{\text{dyn}}} \end{bmatrix}, \tag{4.24}$$

where element $\widehat{k}_{i,j}$ represents an element from matrix $\widehat{\mathbf{K}}$. The element N_{dyn} represents the number of dynamic nodes in the spring system. The matrix $\widehat{\mathbf{K}}_{\text{stat}}$ is defined in the same manner as $\widehat{\mathbf{K}}_{\text{dyn}}$,

$$\widehat{\mathbf{K}}_{\text{stat}} = \begin{bmatrix} \widehat{k}_{1,N_{\text{dyn}+1}} & \cdots & \widehat{k}_{1,N_{\text{nodes}}} \\ \vdots & \cdots & \vdots \\ \widehat{k}_{N_{\text{dyn}},N_{\text{dyn}+1}} & \cdots & \widehat{k}_{N_{\text{dyn}},N_{\text{nodes}}} \end{bmatrix}. \tag{4.25}$$

Note that the dimensions of the matrix $\widehat{\mathbf{K}}_{\text{stat}}$ are $N_{\text{dyn}} \times N_{\text{stat}}$, where N_{stat} represents the number of static nodes in the spring system. The matrix $\mathbf{C}$ has the dimensionality $N_{\text{nodes}} \times N_{\text{springs}}$ and it can be, similarly as $\widehat{\mathbf{K}}$, written into the form

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{\text{dyn}} \\ \mathbf{C}_{\text{stat}} \end{bmatrix}. \tag{4.26}$$

Following the notation in (4.24) and (4.25), the elements of the matrix $\mathbf{C}$ are denoted as $c_{i,j}$ and $\mathbf{C}_{\text{dyn}}$ is defined as

$$\mathbf{C}_{\text{dyn}} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,N_{\text{springs}}} \\ \vdots & \cdots & \vdots \\ c_{N_{\text{dyn}},1} & \cdots & c_{N_{\text{dyn}},N_{\text{springs}}} \end{bmatrix}. \tag{4.27}$$

Since the matrix $\mathbf{C}_{\text{stat}}$ is not used in the following, we do not give the explanation how to derive it from $\mathbf{C}$. Using the matrices $\widehat{\mathbf{K}}$ and $\mathbf{C}$, the spring system from Equation (4.22) can be written as

$$\widehat{\mathbf{K}}\mathbf{x} = \mathbf{C}\mathbf{L}. \tag{4.28}$$

If we take into account that static nodes do not change their positions during the optimization, the system can be decomposed into the form

$$\widehat{\mathbf{K}}_{\text{dyn}}\mathbf{x}_{\text{dyn}} + \widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}} = \mathbf{C}_{\text{dyn}}\mathbf{L}. \tag{4.29}$$

Optimal positions of the dynamic nodes can be written in to the following closed-form solution

$$\mathbf{x}_{\text{dyn}} = \widehat{\mathbf{K}}_{\text{dyn}}^{-1}(\mathbf{C}_{\text{dyn}}\mathbf{L} - \widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}}). \tag{4.30}$$

Note that the expression $\widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}}$ can be calculated only once and it remains unchanged during the optimization. The spring system optimization is described in Algorithm 4.

The iterative direct approach (IDA) is summarized in the following. At each iteration, 2D system is decomposed into two 1D systems, each system is solved by (4.30) and then the 2D system is re-assembled. This process is iterated until convergence (Algoritm 4). Note that the term $\widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}}$ is independent from the dynamic nodes and can be precomputed before entering the iterations.

Algorithm 4 : Optimization of a 2D spring system.

Require:

Positions of dynamic $\mathbf{x}_{\text{dyn}}$ and static nodes $\mathbf{x}_{\text{stat}}$, stiffness vector $\mathbf{k}$ and adjacency matrix $\mathbf{A}$, defined as in (4.20).

Ensure:

New positions of dynamic nodes $\mathbf{x}_{\text{dyn}}$.

Procedure:

1: Construct stiffness matrix $\mathbf{K} = diag(\mathbf{k})$.
2: Calculate matrices $\widehat{\mathbf{K}} = \mathbf{A}^T\mathbf{K}\mathbf{A}$ and $\mathbf{C} = \mathbf{A}^T\mathbf{K}$.
3: Obtain matrices $\widehat{\mathbf{K}}_{\text{dyn}}$, $\widehat{\mathbf{K}}_{\text{stat}}$, and $\mathbf{C}_{\text{dyn}}$ according to (4.24), (4.25) and (4.27).
4: Calculate the product $\widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}}$ for each dimension.
5: **while** stop condition **do**
6: For each dimension do:
7: Extract 1D positions of dynamic nodes from $\mathbf{x}_{\text{dyn}}$.
8: Calculate vector of current lengths $\mathbf{L}$ of the springs (the distances between connected nodes).
9: Solve Equation (4.30) using product from Step 4 and update the vector $\mathbf{x}_{\text{dyn}}$.
10: **end while**

4.3 Layered deformable parts tracker (LDP)

The coarse and mid-level representations from Section 4.1 and Section 4.2 are integrated into a tracker that localizes the target at each time-step within a search region by a top-down localization and bottom-up updates. Our layered deformable parts tracker (LDP) is described in the following, summarized in Algorithm 5 and visualized in Figure 4.6.

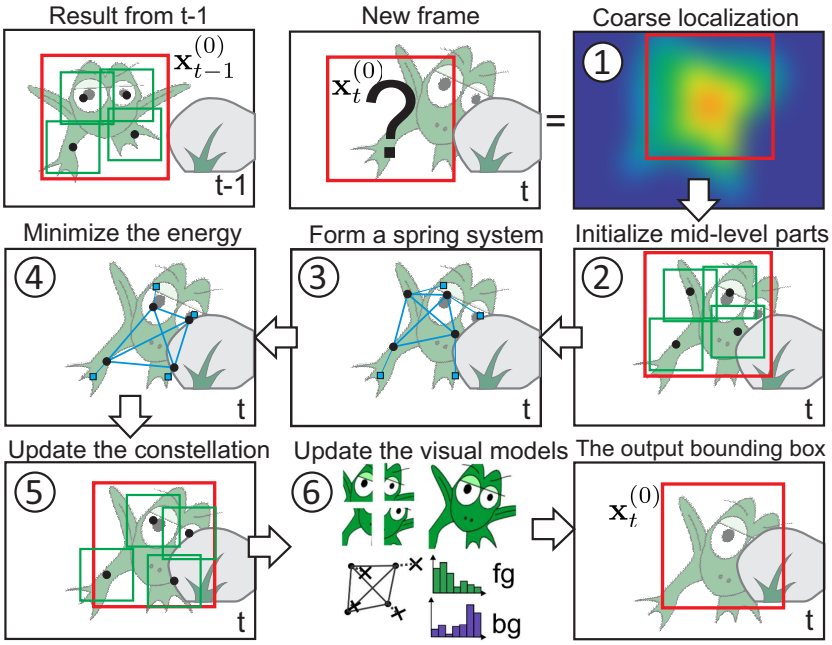

Figure 4.6: Visualization of tracking with LDP tracker. Top down localization is shown in steps 1-4, while bottom-up update is shown in steps 5-6. Result of the tracker at time-step t is bounding box $\mathbf{x}_t^{(0)}$.

4.3.1 Top-down localization

The object is coarsely localized within a search region corresponding to the bounding box estimated at the previous time-step with size increased by α_{reg}. The object center is approximated by position that maximizes the

conditional probability $p(\mathbf{x}_t^{(0)}|I_t, C_t)$ from Section 4.1 and a coarse center translation from $t-1$ to t is estimated (Figure 4.6, step 1). The mid-level representation, i.e, constellation of parts, is initialized by this translation. For each translated part $\mathbf{x}_t^{(i)}$, a local search region centered at the part with the size equal to the part size scaled by α_{reg} is set and similarity to the corresponding visual model $\mathbf{z}_t^{(i)}$ is calculated for all displacements within the region. The position of maximum similarity response, $\mathbf{x}_{tA}^{(i)}$, is determined for each part along with the stiffness coefficients $k_t^{(i)}$ and $k_t^{(i,j)}$ as detailed in Section 4.2.4. A maximum aposteriori probability (MAP) constellation estimate $\hat{\mathbf{X}}_t$ is calculated by minimizing the energy (4.13) of the equivalent spring system (Figure 4.6, steps 2-4).

4.3.2 Bottom-up update

The mid-level and coarse representations are updated as follows (Figure 1.2, steps 5,6). The visual models of parts, $\mathbf{z}_t^{(i)}$, are updated by the visual models taken at the MAP estimates of part positions $\hat{\mathbf{x}}_t^{(i)}$. The details of the update depend on the particular implementation of the visual model. The proposed tracker can apply any visual model, but see Section 5.1 for the details of the models used in the experiments. Updating all part visual models at constant rate might lead to drifting and failure whenever the object is partially occluded or self-occluded. An effective mechanism is applied to address this issue. A part is updated only if its response (similarity) at the MAP position $\hat{\mathbf{x}}_t^{(i)}$ is at least half of the strongest response among all parts. Experimental presentation of this mechanism is shown in Section 5.3.5. The nominal spring lengths (the preferred distances between parts) are updated by an autoregressive scheme $\mu_t^{(i,j)} = \mu_{t-1}^{(i,j)}(1-\alpha_{\text{spr}}) + ||\hat{d}_t^{(i,j)}||\alpha_{\text{spr}}$, where $||\hat{d}_t^{(i,j)}||$ is the distance between the parts (i,j) in the MAP estimate $\hat{\mathbf{X}}_t$ and α_{spr} is the update factor.

The coarse representation is updated next. The MAP object bounding box is estimated by $\hat{\mathbf{x}}_t^{(0)} = \mathbf{T}_t\hat{\mathbf{x}}_{t-1}^{(0)}$, where $\mathbf{T}_t$ is a similarity transform estimated by least squares from the constellation MAP estimates $\hat{\mathbf{X}}_{t-1}$ and $\hat{\mathbf{X}}_t$.

The object template $\mathbf{z}_t^{(0)}$ is updated from $\hat{\mathbf{x}}_t^{(0)}$ in the same manner as parts visual models. The histograms in the global color model C_t are updated from $\hat{\mathbf{x}}_t^{(0)}$ as well. A histogram $\mathbf{h}_t^{(f)}$ is extracted from $\hat{\mathbf{x}}_t^{(0)}$ and another histogram $\mathbf{h}_t^{(b)}$ is extracted from a region surrounding $\hat{\mathbf{x}}_t^{(0)}$ by a factor α_{sur}. The foreground histogram is updated by an autoregressive model, i.e.,

$$p(\mathbf{x}_t|f) = p(\mathbf{x}_{t-1}|f)(1 - \alpha_{\text{hist}}) + \mathbf{h}_t^{(f)}\alpha_{\text{hist}}, \tag{4.31}$$

where α_{hist} is the forgetting factor. The background histogram is updated with $\mathbf{h}_t^{(b)}$ in the same way. The top-down localization and bottom-up update steps are summarized in Algorithm 5.

Algorithm 5 : A tracking iteration of a two-layer deformable parts tracker.

Require:

Coarsely representation $\{\mathbf{I}_{t-1}, \mathbf{C}_{t-1}\}$, mid-level representation $\{\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}\}$, position of an object $\mathbf{x}_{t-1}$ at time-step $t-1$.

Ensure:

New position of an object $\mathbf{x}_t$ at the current time-step t and updated coarse $\{\mathbf{I}_t, \mathbf{C}_t\}$ and mid-level representation $\{\mathbf{X}_t, \mathbf{Z}_t\}$.

Procedure:

1: Coarse estimate the position of the object, denoted as $\mathbf{x}_t^c$. Compute displacement vector Δ_t between positions $\mathbf{x}_t^c$ and $\mathbf{x}_{t-1}$.

2: Displace the mid-level parts by displacement vector Δ_t.

3: For each part calculate the part responses and form an extended spring system – set position of the nodes and stiffness of the springs using (4.15) and (4.14). (Section 4.2.4)

4: Solve spring system by computing the MAP estimate $\mathbf{X}_t^{\text{MAP}}$ of the dynamic nodes by maximizing (4.11) and (4.12). (Section 4.2.5)

5: Calculate the similarity transform $\mathbf{T}_t$ between the positions of mid-level parts from previous and current time-step.

6: Update the spring system parameters and the visual models of the parts ($\mathbf{z}_t^{(i)} \in \mathbf{Z}_t, i = 1...N_P$) on the mid-level representation.

7: Update the position and size of the coarse representation by the transformation $\mathbf{T}_t$ and update the global color $\mathbf{C}_t$ (histograms $\mathbf{h}_f$ and $\mathbf{h}_b$) and root visual model $\mathbf{I}_t$ (template $\mathbf{z}^{(0)}$) by an autoregressive scheme i.e., (4.31).

8: Position of the object $\mathbf{x}_t$ is defined as the center of the coarse representation.

Chapter 5

Experimental analysis

In this chapter the implementation details of the proposed layered deformable parts model (LDP) are given in Section 5.1. Experimental setup is described in Section 5.2, analysis of the LDP tracker is given in Section 5.3 and experimental results of the comparison with state-of-the-art methods are reported in Section 5.4.

5.1 Implementation details and parameters

The LDP described in Section 4.2 requires specification of the coarse visual model $\mathbf{I}_t$ as well as part visual models $\{\mathbf{z}_t^{(i)}\}_{1:N_\mathrm{p}}$. In our implementation, the kernelized correlation filters (KCF) [4] with HOG [44] features are used for the part visual models. The KCF trains a filter to a Gaussian response function for robust patch-based localization, which makes it ideal for our part-based visual likelihood function which assumes approximately Gaussian form of the likelihood. The filter parameters and learning rate are the same as in [4]. The parts have to be large enough to capture locally visually-distinctive regions on the object and have to cover the object without significantly overlapping with each other. Taking into account the usual level of detail of the object appearance, we set the number of parts to $N_p = 4$. The LDP allows any type of connectivity among the parts and our implementation applies a

fully-connected constellation for maximally constrained geometry. The foreground/background models C_t are RGB color histograms with $64 \times 64 \times 64$ bins. The remaining parameters are as follows:

- the constellation scale change parameter is set to $\alpha_{\text{scl}} = 0.1$,
- the uniform component in the color similarity map is set to $\alpha_{\text{col}} = 0.01$,
- the search region scale parameter is set to $\alpha_{\text{reg}} = 1.5$,
- the background histogram extraction area parameter is set to $\alpha_{\text{sur}} = 1.6$,
- the rate of spring system update is $\alpha_{\text{spr}} = 0.95$,
- the histogram update rate is set to $\alpha_{\text{hist}} = 0.05$.

These parameters have a straight-forward interpretation and did not require special tuning. The parameters have been fixed throughout all experiments.

5.1.1 Initialization details

The coarse representation at time-step $t = 1$ is initialized by training a KCF template $\mathbf{z}_1^{(0)}$ from the bounding box $\mathbf{x}_1^{(0)}$ and sampling the foreground and background histograms from the initial bounding box and from the increased region outside the bounding box (as in the update stage). The mid-level representation is initialized as follows. The constellation model is initialized by splitting the initial object bounding box into four equal non-overlapping parts. A KCF visual model $\mathbf{z}_t^{(i)}$ is initialized for each part and the preferred distances between parts are calculated from the initialized positions. The tracker was implemented in Matlab with backprojection and HoG extraction implemented in C and performed at 19fps on a core i7 machine.

Since our tracker uses KCF as a visual model on parts and coarse representation, the complexity of our method is given w.r.t. the KCF complexity, which is $\mathcal{O}(n \log n)$, where n is the number of pixels in the search region.

The LDP has complexity five times the KCF, because of the four parts on mid-level representation and a template on the coarse representation. The localization and update of five KCF visual models take approximately 40ms. Our tracker consists also of the spring system and target segmentation. The spring system takes in average less than 3ms for the optimization. The color segmentation with the histogram extraction is performed in approximately 9ms.

5.2 Experimental setup

The LDP tracker was evaluated using the baseline experiments from the recent benchmark VOT2014 [3, 55]. The number of tested trackers makes VOT2014 the largest short-term tracking benchmark. The benchmark sequences have been collected from the recent Amsterdam library of ordinary videos [27], Online tracking benchmark [1] as well as from a set of previously unpublished sequences. A sequence selection protocol was applied to construct a dataset of 25 sequences that reflect real-life phenomena while keeping the number of sequences low. The targets are annotated by rotated bounding boxes and all sequences are per-frame annotated by visual attributes. The VOT2014 evaluation protocol initializes the tracker from a ground truth bounding box. Once the overlap between the ground truth and tracker output bounding box falls to zero, a failure is detected and tracker is re-initialized. The benchmark measures two aspects of tracking performance: accuracy and robustness. The accuracy is measured as the average overlap, while the robustness measures the number of failures during tracking. The tracking overlap at time-step t is defined as

$$\phi_t = \frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T}, \tag{5.1}$$

where A_t^G represents the ground truth bounding box and A_t^T is the bounding box predicted by a tracker. Apart from reporting raw accuracy/robustness values, the benchmark can rank trackers with respect to these measures sep-

arately and adjusts the ranks by identifying groups of trackers performing indistinguishably. The adjustment can be made by taking the mean, max or min rank from the group. In our setup the adjusted rank is the minimum rank in the group to prevent a potentially poorly-performing tracker pulling down the rank of a better-performing one. The performance equivalency is established by tests of statistical and practical differences. According to VOT2014 [3], the practical difference level was estimated for the ground truth on each sequence and specifies a level of noise in ground truth annotation. Any difference in accuracy of trackers below the level of practical difference is considered negligible.

5.3 The LDP design analysis

This section analyzes the design choices of the LDP. Section 5.3.1 reports the analysis of the spring system optimization, Section 5.3.2 analyzes the influence of the number of parts in the mid-level representation, Section 5.3.3 analyzes the influence of spring system topology, Section 5.3.4 analyzes the influence of individual LDP components and Section 5.3.5 provides qualitative demonstration of how parts react to occlusion. All experiments have been conducted on VOT2014 dataset, and the tracking performance is characterized by the average accuracy (average overlap with the ground truth) and the average robustness (number of failures).

5.3.1 Analysis of the spring system optimization

Since visual object tracking requires real-time performance, it is cruicial that all parts of the algorithm are as efficient as possible. A central part of the deformable parts model is the optimization method that minimizes the energy of the spring system. This section compares the standard baseline method, i.e., the conjugated gradient descend (CGD) from Section 4.2.5 with the proposed iterated direct approach (IDA) which we proposed in Section 4.2.5. To allow controlled experimental analysis of the optimization approach, a

simulated spring system with four dynamic nodes and four anchor nodes was used. We have analyzed the performance of the optimization methods by averaging over a large number of randomly generated spring systems and their displacements. In particular, experiment was repeated 100,000 times and the positions of all nodes were each time randomly perturbed around the initial positions. Four dynamic nodes were initialized on the positions (0,0), (0,1), (1,0) and (1,1). Each node was displaced by the vector $\mathbf{d} = [d_x, d_y]$, where d_x and d_y were sampled from uniform distribution $\mathcal{U}([-0.5; 0.5])$. Each anchor node was set by displacing the corresponding dynamic node by the vector $\mathbf{b} = [b_x, b_y]$, where b_x and b_y were sampled from uniform distribution $\mathcal{U}([-0.25; 0.25])$. The stiffness of i-th dynamic spring was set as

$$k_i = (\sigma d_i)^{-2}, \tag{5.2}$$

where d_i represents the length of the spring and σ is the size change, set on 0.1. The stiffness of j-th static spring was set as

$$k_j = \frac{1}{2} + u_j \overline{k}_{\text{dyn}}, \tag{5.3}$$

where $\overline{k}_{\text{dyn}}$ represents the average stiffness of the dynamic springs and u_j is the random number sampled from the uniform distribution $\mathcal{U}([0; 1])$. The performance of the optimization method was measured by the number of iterations and time needed to reach the stable state and energy of the spring system after the optimization. Time is important to achieve as fast performance as possible and energy should also be as small as possible to reach the state that is close to the global minimum.

Table 5.1 summarizes the performance analysis of CGD and IDA. The first column represents average, standard deviation and median of the number of iterations needed to reach the stable state. The second column represents average, standard deviation and median of the amount of energy that remains in the spring system after the optimization. The last column in the table shows the average time (in milliseconds) needed for the convergence of a method (both methods were implemented in Matlab). The results show that

the proposed IDA needs approximately two times less iterations than the CGD. The remaining energy in a spring system after optimization converges does not change much between the methods, which means that both methods reach global minima. This is not surprising since the optimization function is convex. The Figure 5.1 shows how energy of the spring system drops in each iteration for both methods. It is clear that energy at IDA drops much faster in the beginning of the optimization, which is the key reason for fast performance. Since both methods are equally accurate (both reach the global minima), the convergence speed is of crucial importance. The proposed IDA is in average approximately more than three-times faster than the CGD. With less than 3ms needed for optimization of the spring system, this method can be used in tracking tasks without any larger time losses.

	Number of iterations			**Energy**			**Time**
Method	Avg	Std	Median	Avg	Std	Median	Avg [ms]
IDA	12.75	9.32	10	1.22	3.34	0.83	2.92
CGD	28.01	9.90	28	1.28	3.34	0.83	9.72

Table 5.1: Comparison of the proposed iterative direct approach (IDA) and conjugated gradient descent (CGD) optimization methods.

We have analyzed the average time needed for the initialization of both methods and the time for iterating phase. The first column in Table 5.2 shows average time in milliseconds for each phase. The results show that initialization of the optimization method is faster at IDA. The last column shows the time needed for a single iteration, which is also faster at IDA. Note that Table 5.2 was obtained with the same experiment settings as Table 5.1.

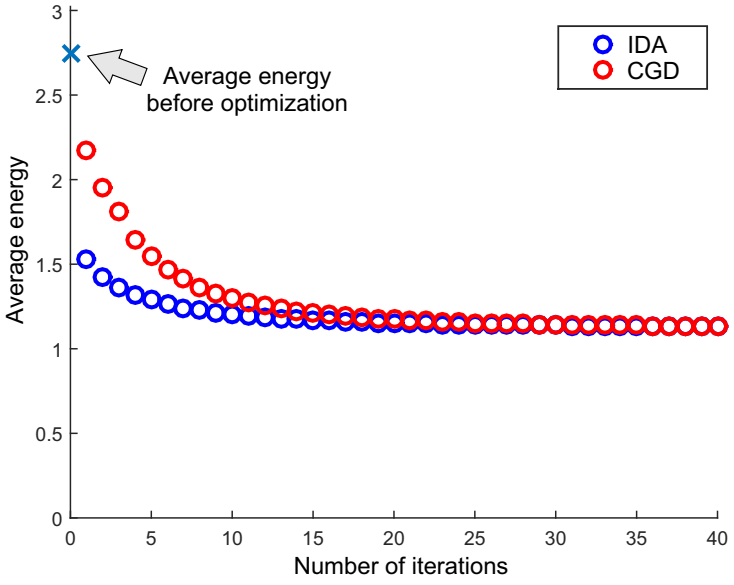

Figure 5.1: Figure illustrates how energy of the spring system drops in every iteration. Experiment is averaged over 10,000 spring systems.

	Average time [ms]				
Method	Init	Iterations	Total	Avg. Iterations	ms per-iteration
IDA	0.18	2.74	2.92	12.75	0.23 (0.215)
CGD	0.35	9.37	9.72	28.01	0.35 (0.335)

Table 5.2: Average time for method initialization, iterations and sum of both is given in the table. The last column represents the average time for one iteration. The number in parenthesis is the average time for one iteration without considering the initialization phase.

We have also observed that the proposed IDA is numerically more stable than the CGD. Figure 5.2 shows an example of numerically unstable spring system, where CGD does not reach the optimal state. The spring system consists of four static and four dynamic nodes. The blue lines represent dynamic springs of an initial spring system and the red lines represent dynamic springs of spring system after optimization. The green small circles denote

the dynamic nodes. The centers of the red circles represent anchor nodes and the radius of circles represent the variance of each node. The dotted lines are static springs before and after optimization. The energy of the final optimized spring system is also given in the figures. Note that the direct method converged to a stable state with much lower energy, while CGD method did not reach that state. The difference is noticeable also in the amount of energy remained in the spring system. Numerical unstability can be explained with the initial positions of the nodes. Since there are two nodes very close to each other, the spring system becomes numerically unstable. This effects on CGD method, while direct method converged to a stable state without problems. There is also a huge difference in number of steps needed for terminating the optimization e.g., the proposed IDA converged in only 5 iterations, whereas the CGD required 471 iterations to converge. Note that the IDA still attains a significantly lower energy than CGD. The unstable spring systems were automatically detected during the experiment and they were not considered in the calculation.

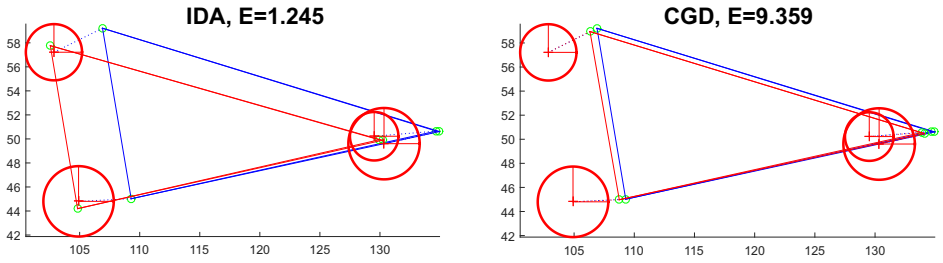

Figure 5.2: The figure shows CGD and IDA optimization methods on a numerically unstable spring system. Blue color represents the initial and red color the optimized spring system.

5.3.2 Number of parts

We used four parts initialized on a regular grid in our implementation. This section reports the analysis of performance variation with the number of parts in the mid-level representation. Figure 5.3 illustrates the constellation of parts for every tested configuration. The proposed tracker (with the 4 parts) is denoted by LDP, configuration with three parts is denoted by LDP-3 where horizontal and vertical orientations are possible, depending on whether the target height is larger or smaller than the width, respectively. The LDP-9 represents our tracker with 9 parts on mid-level. In all cases parts are of the same size – width and height are half of the object width and height, respectively. Note that topology on mid-level representation is always fully connected.

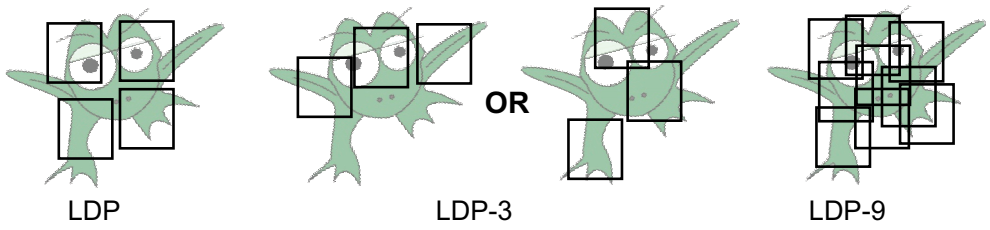

Figure 5.3: Different configurations of LDP parts on local layer.

The results of comparative performance analysis of the LDP, LDP-3 and LDP-9 are shown in Figure 5.4. Average number of failures (Robustness raw) and average overlap (Accuracy raw) are given on the graph. The results show that the accuracy is not significantly affected by the number of parts, on the other hand the robustness drops for the LDP-3 and LDP-9. We believe that tracker with three parts (LDP-3) is performing poorly because the target is covered unnaturally. The tracker with nine parts (LDP-9) has a lot of springs (because of the fully-connected topology), which results in very strong geometric constraints. This can be a reason for larger number of failures – the mid-level representation performs similarly as a coarse representation. Increasing number of parts or different positions of them do not result in

better performance. From this experiment it is clear that the optimal number of parts is four.

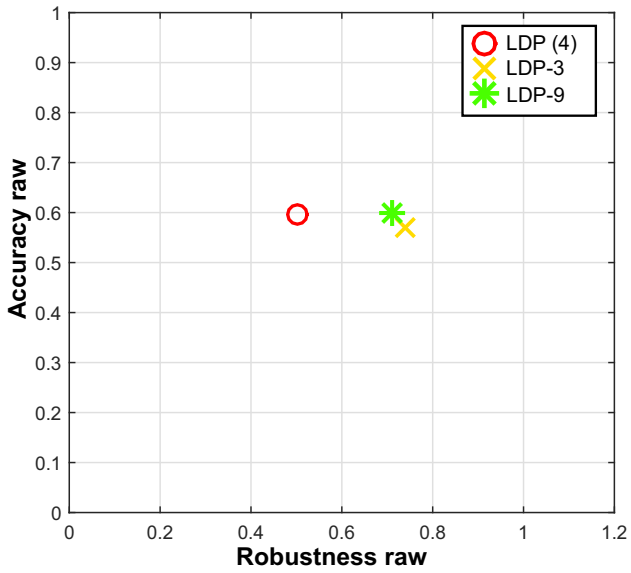

Figure 5.4: AR-plot with the comparison of the LDP tracker (four parts on mid-level representation) with other versions of LDP – each with different number of the parts on mid-level representation. See Figure 5.3 for the different part configurations.

5.3.3 Topology of the spring system

This section reports the analysis of how the constellation topology affects the tracker performance. The different topologies are illustrated in Figure 5.5. The LDP represents the proposed tracker with four fully-connected parts, the LDP-pairwise stands for a similar tracker as LDP, just without diagonal springs. The LDP-star tracker consists of four parts which are connected with a central node in the so-called star-based topology.

The results of performance analysis with respect to different topologies are given in Figure 5.6. The LDP achieves best results in terms of robustness, while the accuracy is comparable with the other two variants. The reason

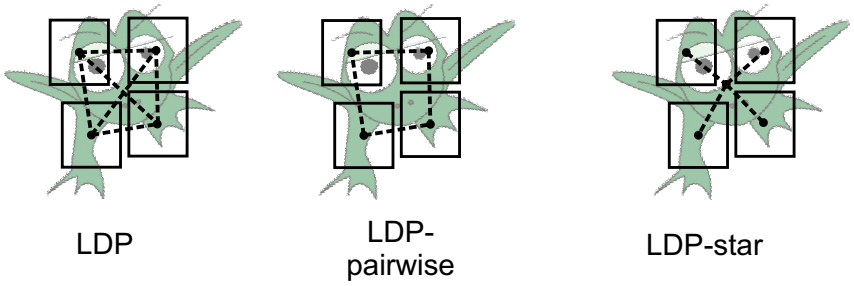

Figure 5.5: The figure shows different variants of the LDP spring system.

for this is likely in weaker geometric constraints of the spring system, which decrease performance. If one would like to achieve a better performance with the non-fully-connected spring system, the parameters should be set more restrictively. However, in our experience, the performance of a fully-connected spring system will be typically higher than of a non-fully-connected spring system.

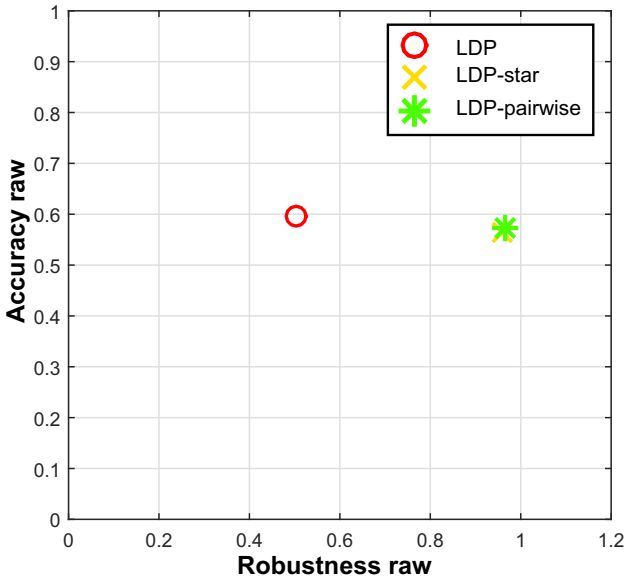

Figure 5.6: AR-plot with the comparison of the LDP tracker (fully connected spring system) with other versions of LDP, each with different spring system topology. See Figure 5.5 for the different topologies.

5.3.4 Analysis of the LDP components

This section reports the analysis of how much each component of the LDP affects the tracking performance. A variant of LDP without target segmentation on coarse representation is denoted with LDP-nosegm, LDP-nosprings represents LDP tracker without springs between the parts and LDP-nomid denotes a tracker without mid-level representation. Note that LDP-nomid has only coarse representation with visual model (correlation filter) and color segmentation, so the tracker does not adapt the size change of the target. The LDP-root-only tracks only with a root node (without segmentation and mid-level representation). Note that the root is actually a modified KCF tracker (without scale adaptation). Table 5.3 summarizes the components of the tested LDP variants, and results of the analysis are reported in Figure 5.7.

Tracker	Root	Segmentation	Springs	Mid-level (nodes)
LDP	✓	✓	✓	✓
LDP-nosegm.	✓	✗	✓	✓
LDP-nosprings	✓	✓	✗	✓
LDP-nomid	✓	✓	✗	✗
LDP-root-only	✓	✗	✗	✗

Table 5.3: Different variants of the LDP tracker. The components used are checked for each variant.

The results show that the LDP-nosegm has much lower robustness with the same accuracy as LDP. Since LDP-nosegm is proposed LDP tracker without segmentation, it is clear that this component has large impact on robustness. The LDP-nospring has lower both, accuracy and robustness, than original LDP. This shows that the spring system is important for both, accuracy and robustness, because it ensures geometric constraints and does not allow parts to be in an arbitrary position. The spring system plays particularly

important role during partial occlusion in which the non-occluded parts *pull* the occluded parts in a direction an object is moving. The spring system also prevents drift of the parts from the target, which improves performance. The LDP tracker without mid-level representation is close to the proposed LDP tracker in terms of robustness, which can explain the importance of the segmentation for the robust tracking. The accuracy at LDP-nomid is, as expected, lower because the mid-level representation handles size changes of the target. In this variant, the target is being tracked with the fixed size of the bounding box. As expected, the LDP-root-only achieves the worst results, since it consists of only a visual template on coarse representation. The accuracy is also lower than the proposed LDP, because the visual template is a KCF tracker without scale adaptation.

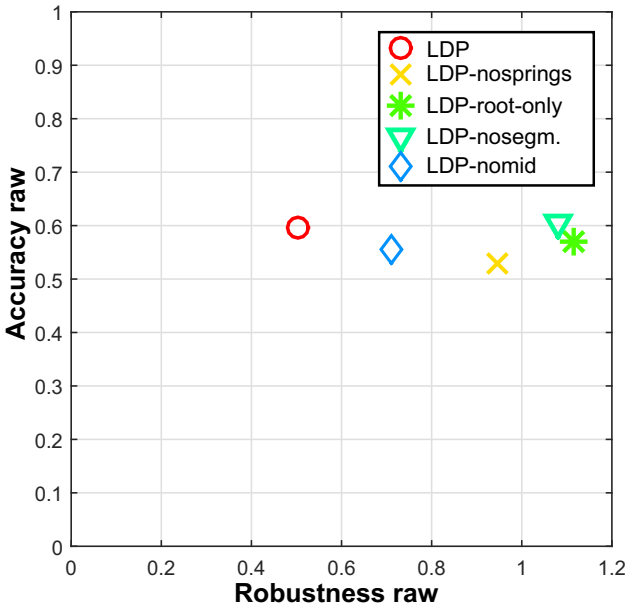

Figure 5.7: AR-plot with the comparison of the LDP tracker with other variants of LDP, each without a particular component. See Table 5.3 for details of the variants.

For completeness of the LDP design analysis, we have collected the analysis results from Section 5.3.2 and Section 5.3.3 with the LDP component analysis from this section. The results are shown in Figure 5.8. The results clearly show the superiority of the proposed LDP over all its variants.

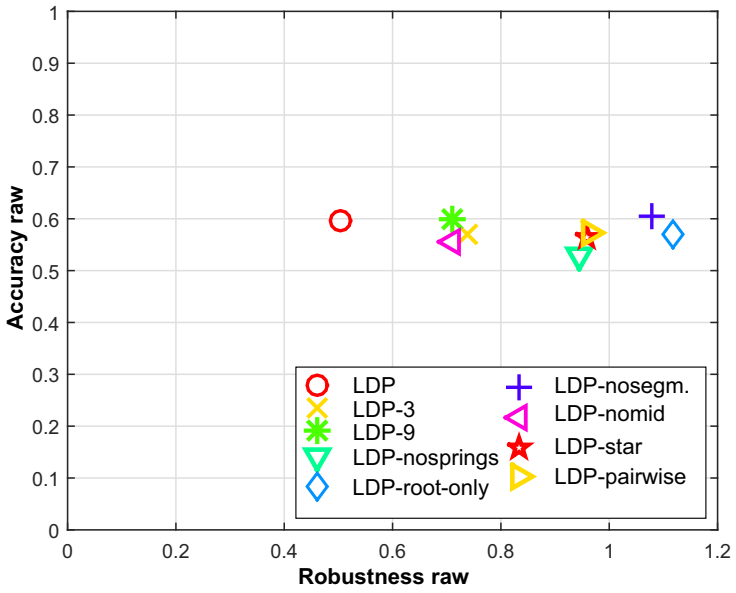

Figure 5.8: Combined visualization of comparison of different LDP variants. AR-plot is obtained combining three AR-plots from Figures 5.4, 5.6 and 5.7.

5.3.5 Partial occlusion handling

Another experiment was performed to qualitatively demonstrate the effectiveness of part adaptations during significant partial occlusions. The LDP was applied to a well-known sequence, in which target (face) undergoes repetitive partial occlusions by a book (see Figure 5.9). The LDP tracked the face without failures. Figure 5.9 shows images of the face taken from the sequence along with the graph of color-coded part weights $w_t^{(i)}$. The automatically computed adaptation threshold is shown in gray. Recall that part is updated if the weight exceeds this threshold (Section 4.3). Observe that partial occlusions are clearly identified by the weight graphs, resulting in drift prevention and successful tracking through partial occlusions.

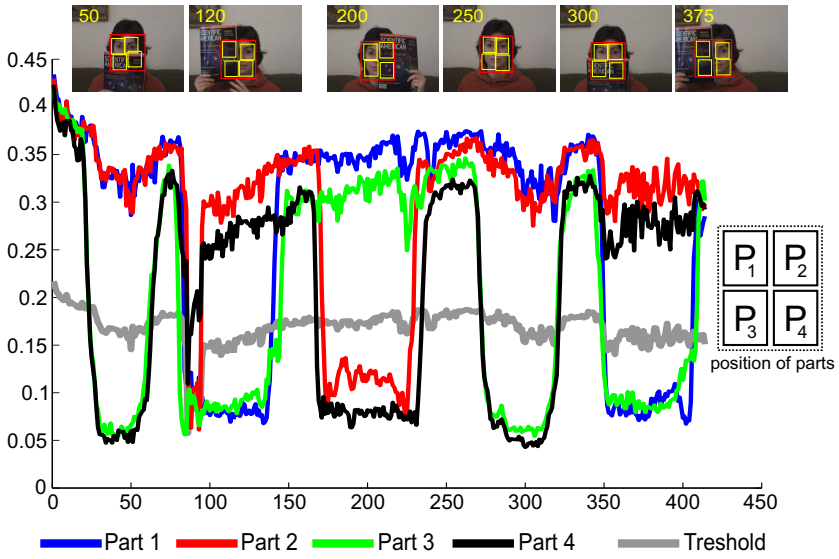

Figure 5.9: Qualitative tracking results of partially occluded target. A sketch of parts is shown on the right-hand side. Part weights are color-coded, with the update threshold shown in gray.

5.4 Comparison to the state-of-the-art

The LDP tracker was compared to the following 12 trackers:

(i) **The standard baselines:** CT [9], IVT [8] and MIL [14].

(ii) **The recent state-of-the-art part-based trackers:** PT [32], DGT [31] and LGT [21].

(iii) **State-of-the-art discriminative holistic trackers:** TGPR [39], Struck [15], DSST [5], KCF [4], SAMF [17] and STC [18].

This is a highly challenging set of recent state-of-the-art containing all published top-performing trackers on VOT2014, including the winner of the challenge DSST [5] and recent trackers from major computer vision conferences and journals.

5.4.1 Per-sequence analysis

The raw per-sequence accuracy and robustness values are shown in Table 5.5. It can be seen that the LDP achieves a high accuracy at low number of failures across all sequences. The LDP is nearly always among the top-three performing trackers per sequence in accuracy and robustness. These values are summarized by weighted averaging over all sequences with weights proportional to the sequence length. In terms of average accuracy, the LDP performs nearly as accurately as the most accurate tracker KCF [4]. In fact, the difference in average overlaps between LDP and KCF is 0.05, which is lower than the weighted average of practical difference over the sequences (0.068). However, the LDP considerably outperforms all other trackers in robustness.

Trackers are represented as points in the average accuracy/number-of-failures plot in Figure 5.10. The point of optimal performance is located at the upper-left corner, corresponding to total overlap and zero failures. Note that the LDP is the closest to that point among all tested trackers, which

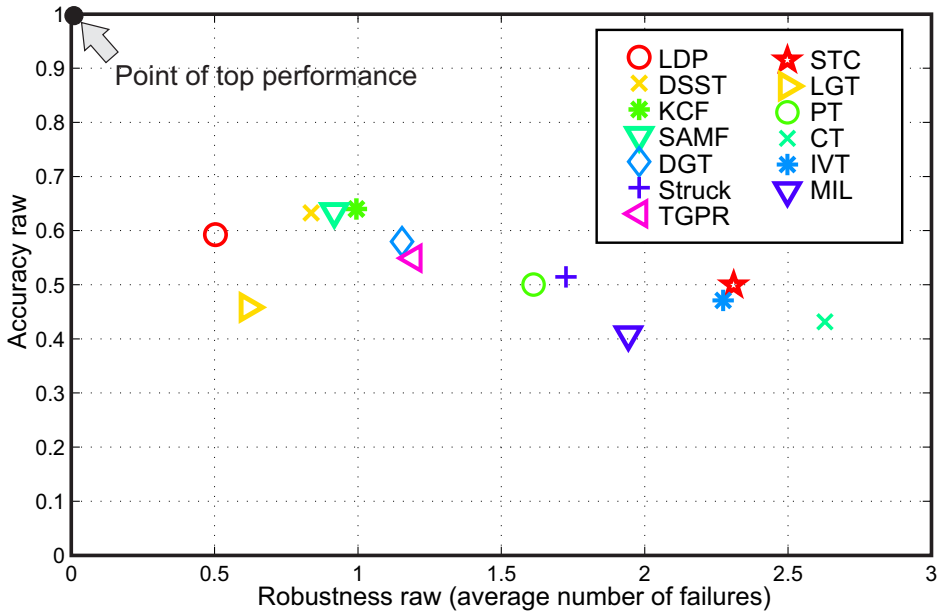

Figure 5.10: The plot shows each tracker as a point in the accuracy/number-of-failures space when averaging over the sequences from Table 5.5.

indicates that the tracker exhibits excellent performance in accuracy as well as robustness. The LDP outperforms the baseline trackers in all aspects and can be considered a state-of-the-art tracker. The LDP outperforms by a large margin the recently published state-of-the-art context tracker STC [18] as well as the recent Gaussian-process-based tracker TGPR [39] and the structured SVM tracker Struck [15] by producing more accurate tracks as well as exhibiting significant improvement in robustness. The LDP also achieves comparable accuracy to the state-of-the-art discriminative correlation trackers DSST [5], KCF [4] and SAMF [17] and significantly outperforms them in robustness. Note that the parts in LDP constellation are essentially KCF [4] trackers without scale adaptation. The significant increase of the robustness compared to KCF [4] thus speaks of effectiveness of the proposed constellation model that is used in LDP. The LDP also outperforms in accuracy and robustness the state-of-the-art part-based trackers, i.e., the structured

SVM parts PT [32], the segmentation-based DGT [31] and the stochastic part-based LGT [21].

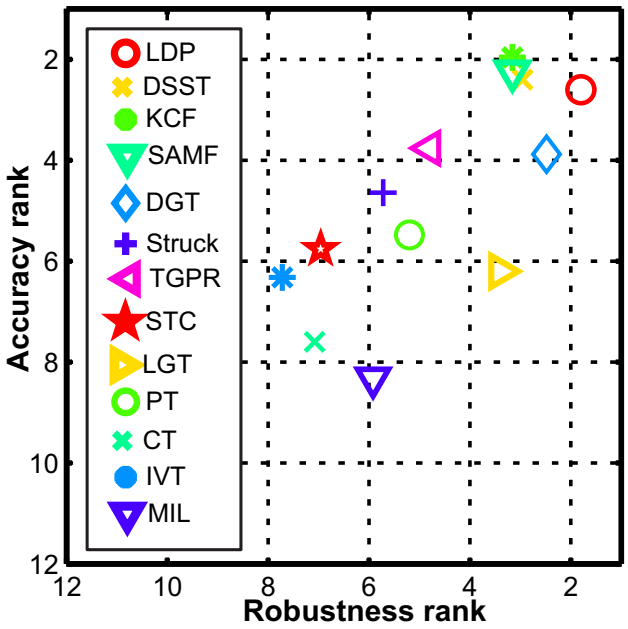

Figure 5.11: The VOT2014 sequence-normalized AR-rank plot.

The improvements of LDP over state-of-the-art are further supported by running the rigorous VOT2014 per-sequence ranking analysis with min-rank equivalency rule in which the final ranking is the average of per-sequence rankings. The results are shown in Table 5.4 under *sequence-normalized ranks* and visualized by the AR-rank plot in Figure 5.11. The point of top performance is in the upper-right corner, where a tracker would be ranked highest in terms of accuracy and robustness. Compared to top-performing trackers, the LDP achieves competitive rank in terms of accuracy and outperforms all trackers in robustness. The results in Table 5.4 show that the LDP achieves the best average rank among all tested trackers. Additional examples of tracking with LDP are shown in Figure 5.13 and in Figure 5.12, where LDP is compared with other trackers.

	seq. normalized			attrib. normalized			All
tracker	A	R	Av	A	R	Av	Av
LDP	2.60	**1.80**	**2.20**	2.00	*1.67*	*1.83*	**2.02**
KCF	**1.96**	3.16	*2.56*	**1.17**	2.50	*1.83*	*2.20*
DSST	2.40	2.96	2.68	**1.17**	2.33	**1.75**	2.21
SAMF	*2.24*	3.16	2.70	*1.33*	2.50	1.91	2.31
DGT	3.88	*2.48*	3.18	3.50	2.50	3.00	3.09
TGPR	3.76	4.8	4.28	5.00	4.33	4.66	4.47
LGT	6.20	3.40	4.80	7.33	**1.33**	4.33	4.56
Struck	4.64	5.72	5.18	4.67	4.83	4.75	4.96
PT	5.48	5.20	5.34	5.33	4.67	5.00	5.17
STC	5.76	6.96	6.36	6.17	9.33	7.75	7.05
IVT	6.32	7.72	7.02	6.83	9.17	8.00	7.51
MIL	8.32	5.92	7.12	10.17	6.50	8.33	7.73
CT	7.60	7.08	7.34	9.17	7.50	8.33	7.84

Table 5.4: Tracker ranks on accuracy (A), robustness (R) and averaged ranks (Av) for the sequence-normalized and attribute-normalized ranking (smaller value means better performance). The last column shows average ranks over both normalization types. Trackers ranked first, second and third are shown in red, blue and green, respectively.

	LDP		DSST		KCF		SAMF		DGT		Struck		TGPR		STC		LGT		PT		CT		IVT		MIL	
sequence	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F
ball (0.026)	0.74	**0.00**	0.56	1.00	0.75	1.00	*0.77*	1.00	**0.81**	**0.00**	0.57	0.87	0.68	**0.00**	0.42	1.00	0.31	1.13	0.46	1.00	0.39	1.00	0.33	4.00	0.45	*0.80*
basketball (0.07)	0.54	1.00	0.64	1.00	0.64	**0.00**	**0.75**	**0.00**	0.50	**0.00**	0.61	1.53	*0.66*	*0.60*	0.54	3.00	0.50	0.80	0.62	1.00	0.54	1.00	0.43	1.00	0.61	2.20
bicycle (0.073)	0.48	**0.00**	0.58	**0.00**	0.62	**0.00**	0.61	**0.00**	*0.63*	**0.00**	0.42	0.60	0.50	*0.13*	0.46	1.00	0.53	0.93	0.44	1.00	0.56	2.00	**0.71**	1.00	0.54	**0.00**
bolt (0.111)	**0.60**	**0.00**	0.56	1.00	0.49	3.00	*0.56*	2.00	0.49	**0.00**	0.52	4.60	0.47	2.53	0.53	4.00	0.38	*0.67*	0.52	7.00	0.42	9.00	0.40	6.00	0.50	5.00
car (0.036)	0.47	**0.00**	**0.73**	**0.00**	*0.70*	**0.00**	0.51	**0.00**	0.57	**0.00**	0.43	**0.00**	0.47	**0.00**	0.53	1.00	0.51	*0.80*	0.43	**0.00**	0.37	**0.00**	0.64	**0.00**	0.42	**0.00**
david (0.118)	0.78	**0.00**	0.80	**0.00**	**0.82**	**0.00**	*0.82*	**0.00**	0.53	1.00	0.59	*0.73*	0.61	1.20	0.74	**0.00**	0.56	**0.00**	0.55	1.00	0.40	1.00	0.69	**0.00**	0.50	**0.00**
diving (0.082)	*0.43*	*1.00*	**0.44**	*1.00*	0.25	4.00	0.24	4.00	0.34	**0.00**	0.28	2.47	0.27	4.00	0.25	4.00	0.33	1.27	0.33	2.00	0.24	2.00	0.23	3.00	0.24	*1.00*
drunk (0.046)	0.51	**0.00**	0.55	**0.00**	0.53	**0.00**	*0.57*	**0.00**	**0.67**	**0.00**	0.50	**0.00**	0.42	**0.00**	0.39	*1.00*	0.52	**0.00**	0.49	**0.00**	0.48	**0.00**	0.51	**0.00**	0.45	**0.00**
fernando (0.062)	0.33	1.00	0.34	1.00	0.41	1.00	0.39	1.00	**0.61**	**0.00**	0.38	2.07	0.40	1.47	0.34	1.00	*0.47*	*0.47*	0.38	1.00	0.39	3.00	0.39	3.00	0.46	2.00
fish1 (0.065)	*0.50*	2.00	0.32	1.00	0.42	3.00	0.49	3.00	**0.56**	**0.00**	0.35	7.07	0.34	2.47	0.22	10.00	0.36	*0.93*	0.39	4.00	0.36	10.00	0.26	4.00	0.40	2.80
fish2 (0.084)	0.34	**1.00**	*0.35*	4.00	0.26	6.00	0.30	5.00	**0.48**	2.00	0.21	6.27	0.26	4.40	0.22	8.00	0.28	*1.80*	0.14	5.00	0.21	4.00	0.20	6.00	0.22	5.87
gymnastics (0.099)	0.53	*1.00*	**0.63**	5.00	0.53	*1.00*	0.54	2.00	*0.58*	**0.00**	0.49	4.13	0.51	2.93	0.58	3.00	0.48	*1.00*	0.57	3.00	0.48	4.00	0.56	4.00	0.26	4.80
hand1 (0.068)	0.50	*1.00*	0.21	2.00	*0.56*	3.00	0.54	3.00	**0.63**	*1.00*	0.35	3.93	0.48	4.13	0.45	6.00	0.55	**0.00**	0.31	4.00	0.31	3.00	0.28	7.00	0.43	1.80
hand2 (0.068)	0.50	*3.00*	**0.52**	6.00	0.49	6.00	0.46	5.00	*0.52*	5.00	0.30	9.47	0.43	6.13	0.27	12.00	0.49	**1.20**	0.29	10.00	0.20	15.00	0.34	8.00	0.40	8.20
jogging (0.076)	*0.80*	*1.00*	0.79	*1.00*	0.79	*1.00*	**0.82**	*1.00*	0.66	**0.00**	0.77	*1.00*	0.78	*1.00*	0.78	*1.00*	0.35	*1.00*	0.72	*1.00*	0.77	*1.00*	0.72	2.00	0.20	*1.00*
motocross (0.046)	**0.50**	**1.00**	0.42	4.00	0.36	*2.00*	0.40	4.00	*0.49*	**1.00**	0.26	2.80	0.35	3.53	0.31	3.00	0.41	**1.00**	0.34	**1.00**	0.22	3.00	0.25	5.00	0.22	3.27
polarbear (0.051)	0.76	**0.00**	0.63	**0.00**	*0.78*	**0.00**	0.71	**0.00**	**0.81**	**0.00**	0.62	**0.00**	0.67	**0.00**	0.47	**0.00**	0.65	**0.00**	0.60	**0.00**	0.60	**0.00**	0.45	**0.00**	0.46	**0.00**
skating (0.099)	0.47	1.00	*0.59*	**0.00**	**0.68**	1.00	0.45	**0.00**	0.39	7.00	0.52	1.00	0.53	1.20	0.53	2.00	0.32	*0.40*	0.52	1.00	0.51	2.00	0.56	4.00	0.25	3.60
sphere (0.027)	0.81	**0.00**	**0.92**	**0.00**	*0.90*	**0.00**	0.88	**0.00**	0.84	**0.00**	0.70	**0.00**	0.72	**0.00**	0.71	**0.00**	0.64	**0.00**	0.65	**0.00**	0.61	**0.00**	0.38	**0.00**	0.57	**0.00**
sunshade (0.075)	**0.80**	**0.00**	*0.78*	**0.00**	0.76	**0.00**	0.76	**0.00**	0.51	**0.00**	0.78	**0.00**	0.71	*0.27*	0.75	**0.00**	0.55	0.40	0.63	**0.00**	0.40	4.00	0.75	3.00	0.43	2.87
surfing (0.081)	0.90	**0.00**	*0.90*	**0.00**	0.79	**0.00**	0.80	**0.00**	0.63	**0.00**	**0.90**	**0.00**	0.87	**0.00**	0.78	**0.00**	0.57	**0.00**	0.88	**0.00**	0.66	**0.00**	0.68	**0.00**	0.38	**0.00**
torus (0.029)	0.84	**0.00**	0.81	**0.00**	**0.85**	**0.00**	*0.84*	**0.00**	0.83	**0.00**	0.50	3.67	0.67	1.53	0.48	4.00	0.63	**0.00**	0.51	5.00	0.54	4.00	0.70	*1.00*	0.43	3.80
trellis (0.068)	0.62	**0.00**	*0.80*	**0.00**	0.79	**0.00**	**0.81**	**0.00**	0.48	**0.00**	0.53	1.60	0.61	*0.53*	0.66	2.00	0.48	**0.00**	0.48	1.00	0.33	5.00	0.54	3.00	0.42	4.73
tunnel (0.065)	0.35	**0.00**	**0.80**	**0.00**	*0.68*	**0.00**	0.54	**0.00**	0.44	8.00	0.32	*0.13*	0.44	0.27	0.31	**0.00**	0.36	1.47	0.31	**0.00**	0.19	**0.00**	0.30	**0.00**	0.32	2.47
woman (0.08)	*0.78*	1.00	**0.79**	1.00	0.74	1.00	0.76	1.00	0.54	**0.00**	0.75	**0.00**	0.77	1.00	0.75	1.00	0.36	1.13	0.76	1.00	0.57	4.00	0.47	4.00	0.26	*0.47*
Average (0.068)	0.59	**0.50**	*0.63*	0.84	**0.64**	0.99	**0.64**	0.92	0.58	1.15	0.51	1.73	0.55	1.19	0.50	2.31	0.46	0.62	0.50	1.61	0.43	2.63	0.47	2.27	0.41	1.94

Table 5.5: The per-sequence accuracy (A) and number of failures (F) over all 25 sequences and 13 tested trackers. The practical difference value of each sequence is given next to the name of each sequence in parentheses.

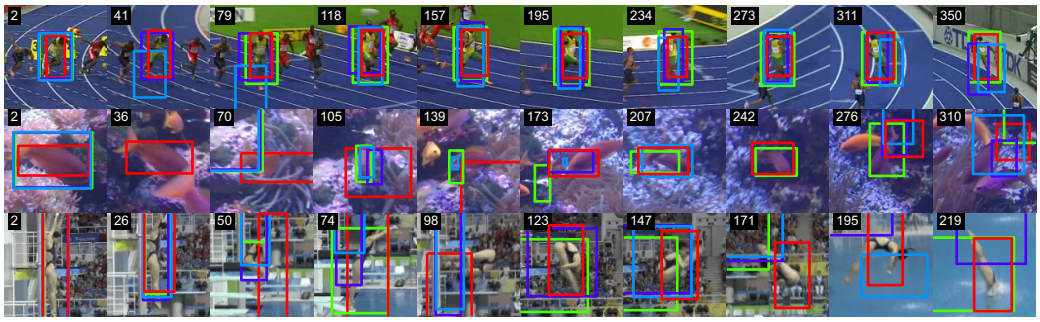

Figure 5.12: Qualitative comparative examples of tracking for LDP, KCF, Struck and IVT shown in red, green, violet and blue, respectively.

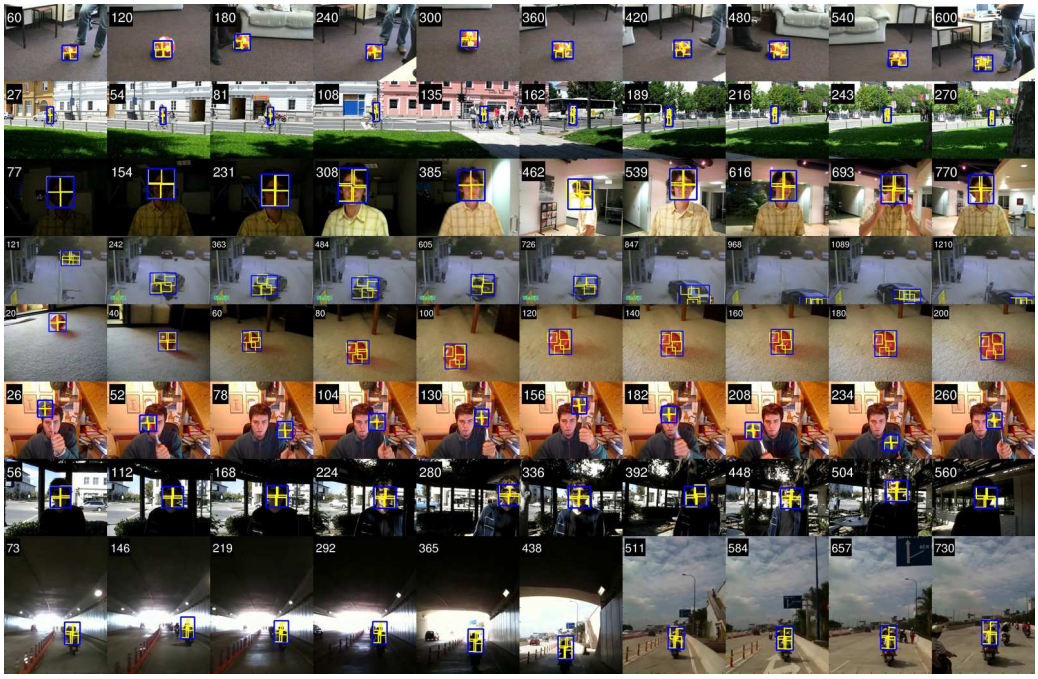

Figure 5.13: Qualitative examples of LDP tracker on six sequences. Tracking bounding box is visualized with blue color and four parts on mid-level representation are shown in yellow.

Per-difficulty analysis

According to VOT2014 [55], sequences are classified into four classes (hard, intermediate, intermediate/easy and easy). The twenty five VOT2014 sequences are classified as follows:

- **Hard:** motocross, hand2, diving, fish2, bolt, hand1.
- **Intermediate:** fish1, fernando, gymnastics, torus, skating.
- **Interm./easy:** trellis, basketball, tunnel, sunshade, jogging, woman.
- **Easy:** bicycle, david, ball, sphere, car, drunk, surfing, polarbear.

A sequence-normalized analysis was performed on each of this four sequence subsets. The AR-plots on the Figure 5.14 show rankings of the trackers for each subset. The LDP tracker is positioned on each plot among the top-performing trackers (close to the top-right corner). It is interesting that it performs the best on sequences which are labelled as hard.

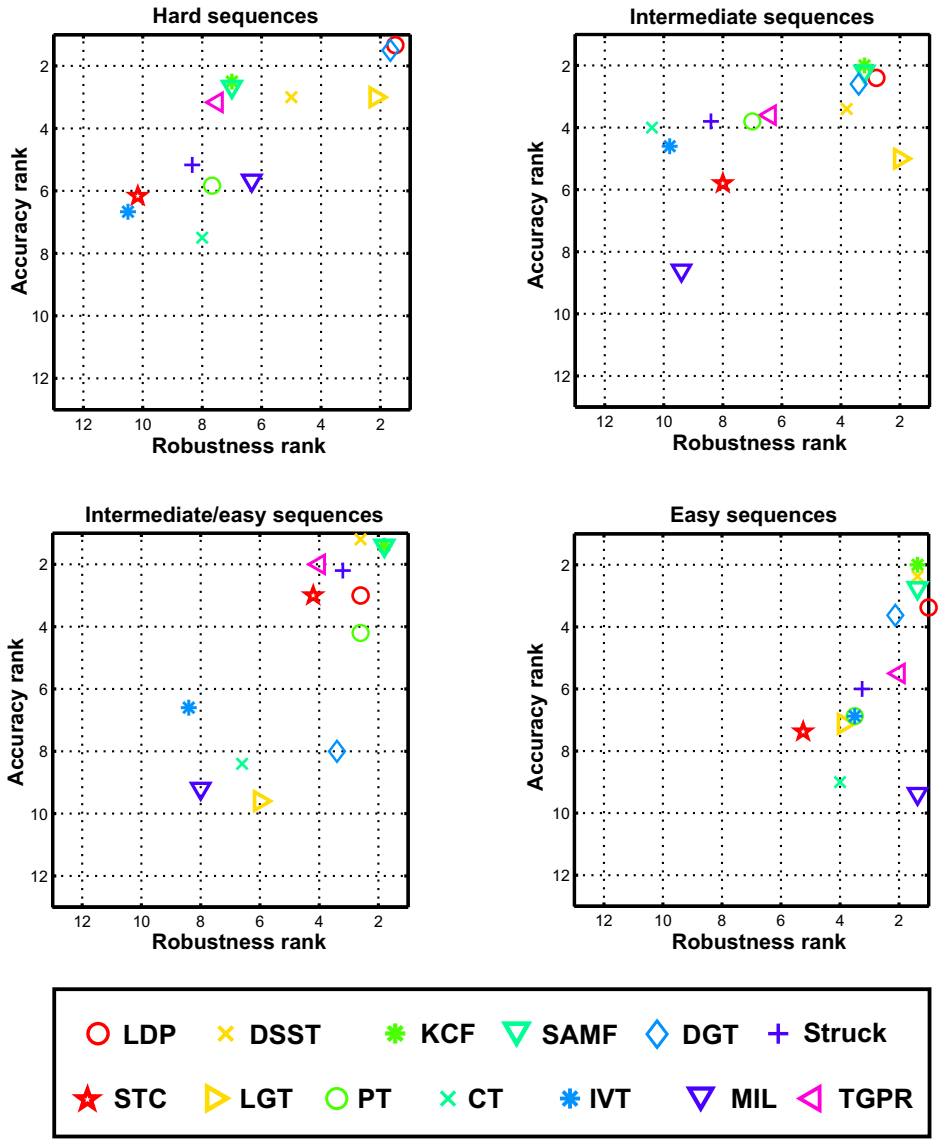

Figure 5.14: Four VOT2014 sequence-normalized AR-rank plots. Each plot represents tracker ranking on a subset of sequences. Each sequence is classified to a difficulty class according to the VOT2014.

5.4.2 Per-attribute analysis

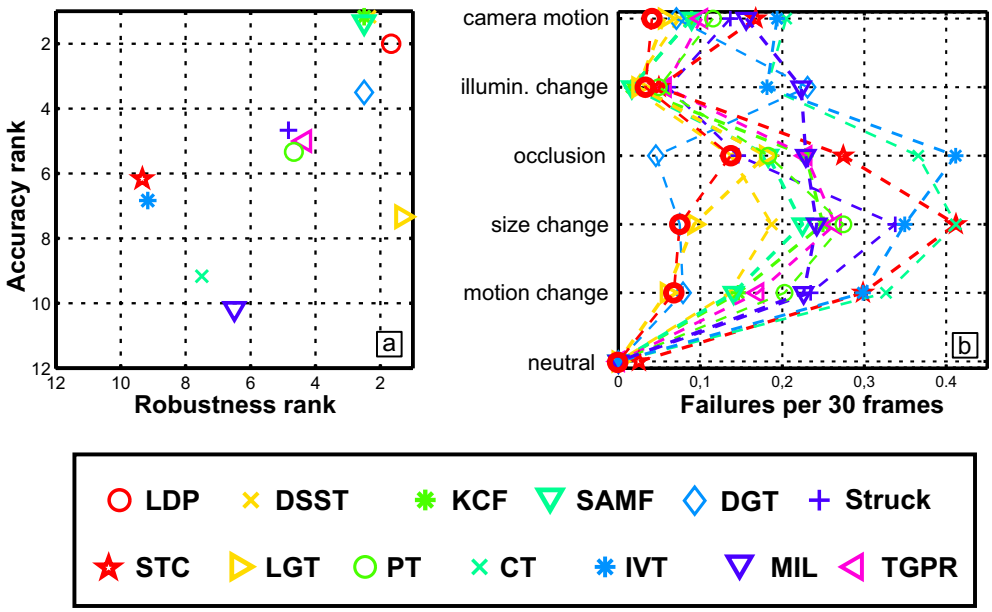

Figure 5.15: The VOT2014 attribute-normalized AR-rank plot (a) and per-attribute failures plot (b).

The VOT2014 benchmark contains per-frame visual attribute annotation. To analyze the robustness with respect to the attributes, the number of failures per attribute was obtained by summing across all 25 sequences. The number of failures per visual attribute for each tracker is shown in Figure 5.15b. Note that the LDP is again consistently the top or among the top-two trackers at each attribute, which implies that the tracker reaches top performance by addressing well the challenges posed by all attributes. Raw results of all tested trackers among the visual attributes are given in Table 5.6. The last column represents the average overlap and total number of failures across the attributes. The VOT2014 also provides attribute-normalized ranking that averages ranks across the attributes in contrast to the sequence-normalized ranking. The resulting attribute-normalized AR-rank plot is shown in Figure 5.15a, which again shows the LDP close to

the upper-right corner, exhibiting top performance. The values of ranks are shown in Table 5.4 under *attribute-normalized ranks*. Averaging ranks over both sequence-normalized and attribute-normalized ranks, the LDP is again the top-performing tracker. In Figure 5.16 AR-rank plot is shown for each visual attribute separately. Note that the LDP is consistently ranked among the top trackers with respect to each attribute.

	camera_motion (0.072)		illum_change (0.088)		occlusion (0.078)		size_change (0.065)		motion_change (0.069)		empty (0.050)		*Average*	
	Overlap	**Failures**	**Overlap**	**Failures**	**Overlap**	**Failures**	**Overlap**	**Failures**	**Overlap**	**Failures**	**Overlap**	**Failures**	**Overlap**	**Failures**
LDP	0.60	**11.00**	0.58	2.00	0.59	*3.00*	0.51	**6.00**	0.62	*12.00*	0.52	**0.00**	0.59	**8.87**
DSST	*0.66*	20.00	**0.75**	**1.00**	*0.63*	*3.00*	0.52	15.00	0.65	24.00	0.56	**0.00**	0.64	16.90
KCF	**0.67**	24.00	*0.74*	**1.00**	**0.64**	5.00	**0.58**	20.00	**0.67**	26.00	0.54	**0.00**	**0.66**	19.79
SAMF	0.66	24.00	0.67	**1.00**	0.61	4.00	0.56	18.00	*0.67*	25.00	*0.57*	**0.00**	*0.64*	19.23
DGT	0.56	19.00	0.47	14.00	0.48	**1.00**	*0.58*	**6.00**	0.58	14.00	**0.68**	**0.00**	0.56	13.78
Struck	0.53	36.67	0.51	3.67	0.58	3.20	0.40	27.07	0.51	41.67	0.50	*0.07*	0.51	30.39
TGPR	0.57	27.27	0.57	3.47	0.61	5.00	0.47	21.20	0.55	30.20	0.43	**0.00**	0.55	22.67
STC	0.54	45.00	0.57	3.00	0.60	6.00	0.42	33.00	0.53	53.00	0.41	1.00	0.52	37.76
LGT	0.44	*15.20*	0.45	*1.47*	0.33	3.93	0.43	*7.40*	0.46	**10.47**	0.52	**0.00**	0.45	*10.34*
PT	0.52	31.00	0.50	3.00	0.60	4.00	0.40	22.00	0.49	36.00	0.50	**0.00**	0.50	25.83
CT	0.43	55.00	0.38	11.00	0.42	8.00	0.36	33.00	0.42	58.00	0.49	**0.00**	0.42	44.03
IVT	0.47	52.00	0.56	11.00	0.40	9.00	0.41	28.00	0.49	53.00	0.52	**0.00**	0.48	40.83
MIL	0.41	41.87	0.38	13.53	0.28	5.00	0.37	19.40	0.41	40.13	0.42	**0.00**	0.40	32.15

Table 5.6: The per-attribute accuracy (overlap) and number of failures over all 5 visual attributes and 13 tested trackers. The practical difference value of each attribute is given next to the name of the attribute in parentheses.

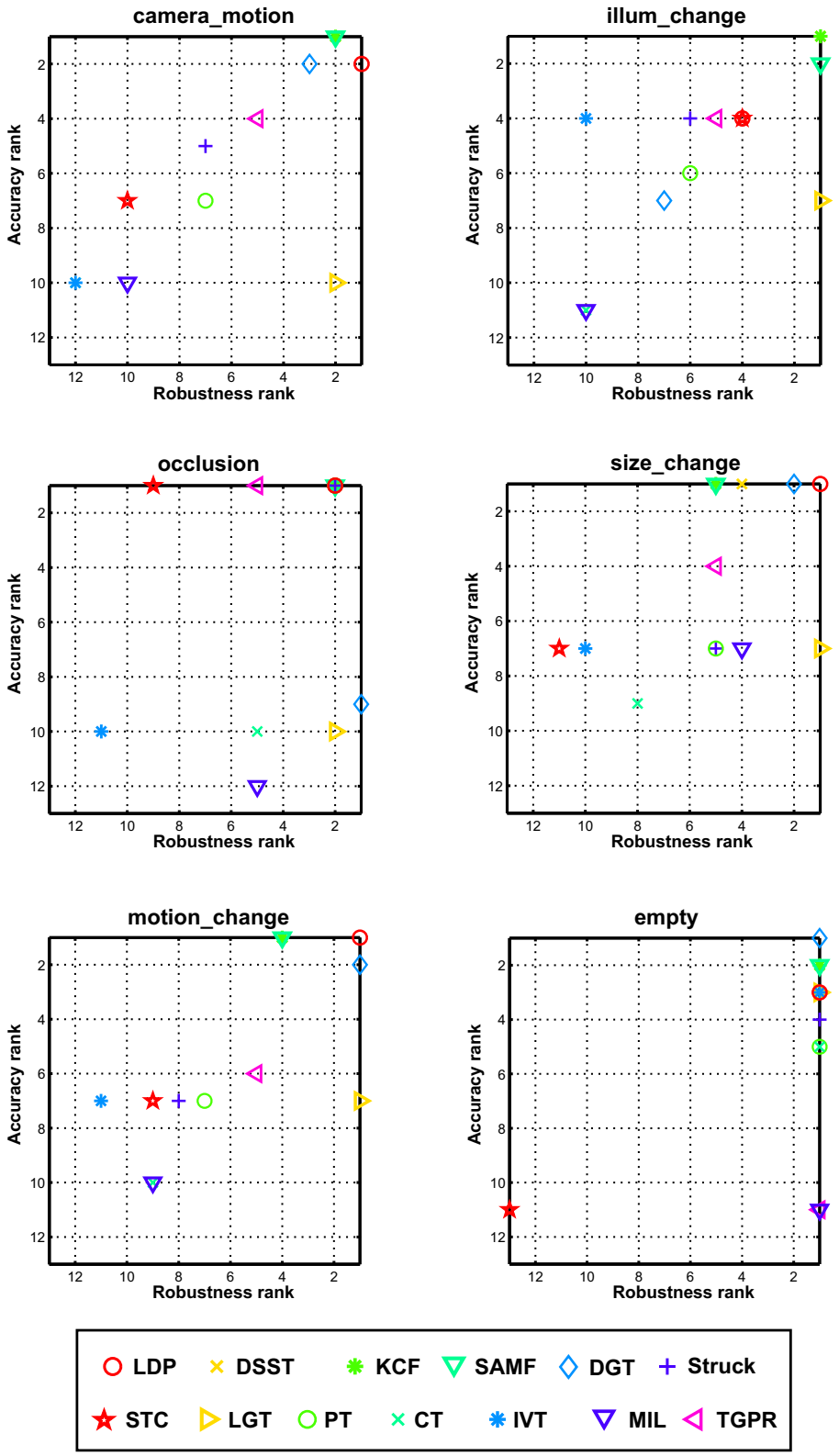

Figure 5.16: The VOT2014 AR-rank plots for each visual attribute. In each plot the top performing point is top-right corner.

Chapter 6

Conclusion

We presented a novel deformable-part based method for visual tracking, called LDP tracker. It consists of two layers, the coarse and mid-level representation. The coarse representation has a target template (holistic model) and a color model that uses Markov random field segmentation. The mid-level representation consists of a set of parts connected with the spring system. A part-based tracker (LDP) combines a coarse object representation with a mid-level deformable parts model in top-down localization and bottom-up updates. The main contributions of this thesis are as follows. The developed deformable parts model jointly treats the visual and geometric properties within a single formulation, resulting in a convex optimization problem. We have shown that the dual representation of such a deformable parts model is an extended spring system and that minimization of the corresponding energy function leads to a MAP inference on the deformable parts model. A novel optimization method – iterative direct approach (IDA) for efficient minimization of the spring system energy is proposed in this thesis.

The proposed optimization method, IDA, is analyzed and compared to the standard optimization approach, i.e., conjugated gradient descend (CGD). The results show that it outperforms the CGD in terms of lower number of iterations as well as in lower optimization time. Our tracker is rigorously compared against the state-of-the-art trackers on a recent highly challenging

VOT2014 [3] benchmark. The LDP tracker outperforms the related state-of-the-art part-based trackers as well as state-of-the-art trackers that use a single visual model, including the winner of the VOT2014 challenge and runs in real-time. Additional tests show that improvements come from the fully-connected constellation and the top-down, bottom-up combination of the coarse representation with the proposed deformable parts model.

6.1 Applications and dissemination

The work from this masters thesis was presented at several international conferences. Preliminary results were presented at the VOT Challenge 2014 [3] held at the European Conference on Computer Vision, ECCV2014, which is one of the top conferences in computer vision. The proposed visual tracker was presented at Computer Vision Winter Workshop conference in Austria as an invited talk [22]. We compared the performance of the tracker with the current state-of-the-art tracking methods on a VOT Challenge 2014 system [3]. Since the tracker was finalized after the challenge, it was not able to participate there, but all the challenge results are publicly available, which allowed us to re-run the challenge analysis with our tracker included. The analysis showed that our tracker outperformed the current state-of-the-art trackers i.e., it outperforms the VTO2014 winner by 40% better robustness. The new efficient spring system optimization was presented at ERK2015 conference [25].

Since the tracker achieved very good results on the standard benchmarks, we tested it on two challenging and diverse real-world projects. The goal of the first project, DroneTrack, was to enable a quadrotor following a person moving through space only with the use of a camera. This kind of application is highly challenging for visual tracking because it needs very robust tracking and real-time performance. Therefore we re-implemented the tracker in the programming language C++ and added speed-ups and simplifications to achieve real-time performance at 50 frames-per-second on a standard laptop.

In the Figure 6.1 left, the view from the drone during the tracking is shown and the schema of the system is shown on the right side of the image. A video presentation of the autonomous drone following a person was presented at the Night of researchers event in September 2015 [23] and also published at the web portal Frekvenca X on Val202 [24].

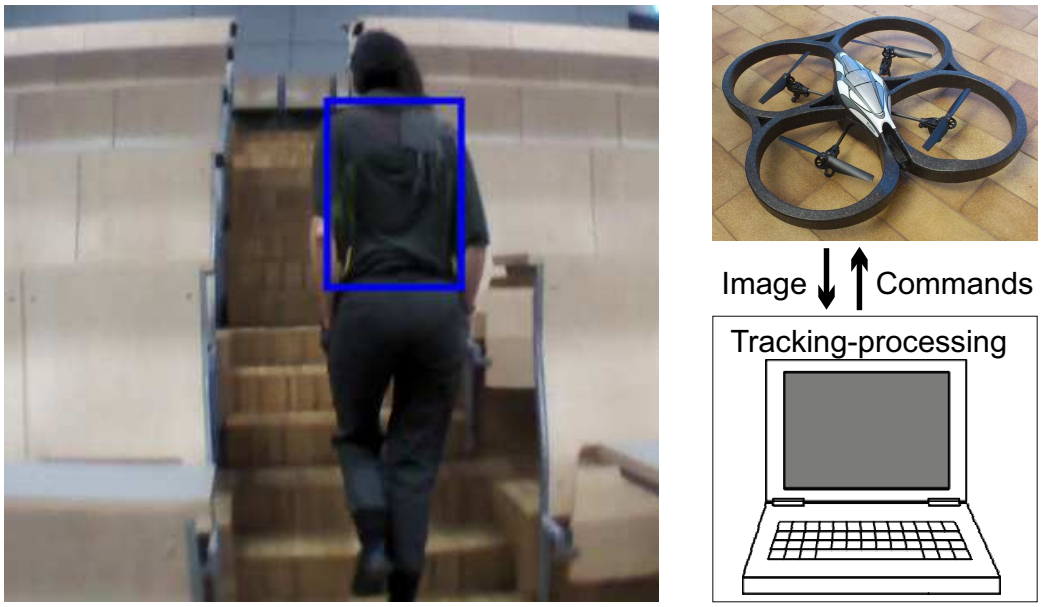

Figure 6.1: The system for visual object tracking with the drone. On the right side the communication between the drone and computer is shown. The drone sends sequential images to the computer which performs the tracking and calculates the response for the drone according to the object position in the image. Commands for the navigation are then sent to the drone. On the left side the image from drone is shown with the highlighted tracking region.

As the part of the ARRS L2-6765 [26] research project we applied the tracker on a traffic signs visual tracking from a mobile mapping systems. The challenge here arises from the significant size change of the traffic sign during the tracking (the size changes to as low as 5% of original size). The performance of the tracker was excellent. Figure 6.2 shows qualitative results from traffic signs tracking.

The results on the standard benchmarks show that our tracker is ranked among the current state-of-the-art trackers and it also outperforms the winner of the recent VOT2014 Challenge. The tracker therefore presents a scientific contribution on the field of visual tracking and the results from two real-world applications support its practical effectiveness. A paper describing this tracker is being submitted to a major journal in the field of computer vision.

Figure 6.2: Visual tracking of the traffic signs with the camera mounted on a car in a mobile mapping system. The tracker is able to addresses the size change of the object even though it significantly changes during the tracking.

6.2 Future work

The proposed deformable parts model is highly extendable, therefore the visual models on parts can be easily replaced with other discriminative or generative models. Since the model is fully probabilistic, it can be readily integrated with probabilistic dynamic models, i.e., Kalman filter, to improve the prediction of the target location in a new frame. The proposed tracker should also be modified for the long-term tracking, where tracker should be able to perform target re-detection. This could be done by an online-learned object detector to guide the tracker. Experiments showed that segmentation on the coarse representation contributes a lot to robust tracking, but it has problems on sudden illumination changes. Therefore we think that segmentation mask could be learned via correlation filter, which could result in a more robust tracking during changing illumination.

Bibliography

[1] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, in: Comp. Vis. Patt. Recognition, 2013, pp. 2411– 2418.

[2] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Čehovin, G. Nebehay, G. Fernandez, T. e. a. Vojir, The visual object tracking vot2013 challenge results, in: Vis. Obj. Track. Challenge VOT2013, In conjunction with ICCV2013, 2013, pp. 98–111.

[3] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojir, G. et al. Fernandez, The visual object tracking vot2014 challenge results, in: Proc. European Conf. Computer Vision, 2014, pp. 191–217.

[4] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Trans. Pattern Anal. Mach. Intell. 37 (3) (2014) 583–596.

[5] M. Danelljan, G. Häger, F. S. Khan, M. Felsberg, Accurate scale estimation for robust visual tracking, in: Proc. British Machine Vision Conference, 2014, pp. 1–11.

[6] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (5) (2003) 564–577.

[7] R. T. Collins, X. Liu, M. Lordeanu, Online selection of discriminative tracking features, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (2005) 1631–1643.

[8] D. A. Ross, J. Lim, R.-S. Lin, M.-H. Yang, Incremental learning for robust visual tracking, Int. J. Comput. Vision 77 (1-3) (2008) 125–141.

[9] K. Zhang, L. Zhang, M.-H. Yang, Real-time compressive tracking, in: Proc. European Conf. Computer Vision, 2012, pp. 864–877.

[10] X. Mei, H. Ling, Robust visual tracking and vehicle classification via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (11) (2011) 2259–2272.

[11] Z. Hong, X. Mei, D. Prokhorov, D. Tao, Tracking via robust multi-task multi-view joint sparse representation, in: Int. Conf. Computer Vision, 2013, pp. 649–656.

[12] S. Avidan, Support vector tracking, IEEE Trans. Pattern Anal. Mach. Intell. 26 (8) (2004) 1064–1072.

[13] H. Grabner, M. Grabner, H. Bischof, Real-time tracking via on-line boosting, in: Proc. British Machine Vision Conference, Vol. 1, 2006, pp. 47–56.

[14] B. Babenko, M.-H. Yang, S. Belongie, Robust object tracking with online multiple instance learning, IEEE Trans. Pattern Anal. Mach. Intell. 33 (8) (2011) 1619–1632.

[15] S. Hare, A. Saffari, P. H. S. Torr, Struck: Structured output tracking with kernels, in: Int. Conf. Computer Vision, IEEE Computer Society, Washington, DC, USA, 2011, pp. 263–270.

[16] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, in: Comp. Vis. Patt. Recognition, IEEE, 2010, pp. 2544–2550.

[17] Y. Li, J. Zhu, A scale adaptive kernel correlation filter tracker with feature integration, in: Proc. European Conf. Computer Vision, 2014, pp. 254–265.

[18] K. Zhang, L. Zhang, Q. Liu, D. Zhang, M.-H. Yang, Fast visual tracking via dense spatio-temporal context learning, in: Proc. European Conf. Computer Vision, Springer International Publishing, 2014, pp. 127–141.

[19] J. Hoey, Tracking using flocks of features, with application to assisted handwashing, in: Proc. British Machine Vision Conference, Vol. 1, 2006, pp. 367–376.

[20] B. Martinez, X. Binefa, Piecewise affine kernel tracking for non-planar targets, Patt. Recogn. 41 (12) (2008) 3682–3691.

[21] L. Čehovin, M. Kristan, A. Leonardis, Robust visual tracking using an adaptive coupled-layer visual model, IEEE Trans. Pattern Anal. Mach. Intell. 35 (4) (2013) 941–953.

[22] A. Lukežič, L. Čehovin, M. Kristan, A two-layer spring-system-based deformable parts model for visual tracking, Computer Vision Winter Workshop 2015, Invited talk (February 2015).
URL `http://cvww2015.icg.tugraz.at/program.php`

[23] M. Kristan, Računalniški vid v avtonomnih robotskih sistemih, Noč raziskovalcev (September 2015).

[24] Si lahko v vesolju zavežemo vezalke?, Val202 frekvenca X (September 2015).
URL `http://val202.rtvslo.si/2015/09/si-lahko-v-vesolju-zavezemo-vezalke/`

[25] A. Lukežič, L. Čehovin, M. Kristan, Efficient spring system optimization for part-based visual tracking, in: International Electrotechnical and Computer Science Conference, Vol. B, 2015, pp. 45–48.

[26] D. Skočaj, Vzdrževanje velikih podatkovnih baz na podlagi vizualne informacije z inkrementalnim učenjem, projekt ARRS: L2-6765 (B) (2014–2017).

[27] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, M. Shah, Visual tracking: An experimental survey, IEEE Trans. Pattern Anal. Mach. Intell. 36 (7) (2014) 1442–1468.

[28] J. Kwon, K. M. Lee, Tracking by sampling and integrating multiple trackers, IEEE Trans. Pattern Anal. Mach. Intell. 36 (7) (2014) 1428–1441.

[29] M. E. Maresca, A. Petrosino, Matrioska: A multi-level approach to fast tracking by learning, in: Proc. Int. Conf. Image Analysis and Processing, 2013, pp. 419–428.

[30] N. M. Artner, A. Ion, W. G. Kropatsch, Multi-scale 2d tracking of articulated objects using hierarchical spring systems, Patt. Recogn. 44 (4) (2011) 800–810.

[31] Z. Cai, L. Wen, Z. Lei, N. Vasconcelos, S. Li, Robust deformable and occluded object tracking with dynamic graph, IEEE Trans. Image Proc. 23 (12) (2014) 5497 – 5509.

[32] R. Yao, Q. Shi, C. Shen, Y. Zhang, A. van den Hengel, Part-based visual tracking with online latent structural learning, in: Comp. Vis. Patt. Recognition, 2013, pp. 2363–2370.

[33] M. Godec, P. M. Roth, H. Bischof, Hough-based tracking of non-rigid objects., Comp. Vis. Image Understanding 117 (10) (2013) 1245–1256.

[34] G. Zhu, J. Wang, C. Zhao, H. Lu, Part context learning for visual tracking, in: Proc. British Machine Vision Conference, 2014, pp. 1–12.

[35] M. L. Black, A. D. Jepson, Eigentracking: Robust matching and tracking of articulated objects using a view-based representation, Int. J. Comput. Vision 26 (1998) 63–84.

[36] X. Mei, H. Ling, Robust visual tracking using ℓ_1 minimization, in: Int. Conf. Computer Vision, 2009, pp. 1436–1443.

[37] P. Pérez, C. Hue, J. Vermaak, M. Gangnet, Color-based probabilistic tracking, in: Proc. European Conf. Computer Vision, 2002, pp. 661–675.

[38] H. Li, C. Shen, Q. Shi, Real-time visual tracking using compressive sensing, in: Comp. Vis. Patt. Recognition, 2011, pp. 1305–1312.

[39] J. Gao, H. Ling, W. Hu, J. Xing, Transfer learning based visual tracking with gaussian processes regression, in: Proc. European Conf. Computer Vision, Vol. 8691, 2014, pp. 188–203.

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

[41] A. Saffari, C. Leistner, J. Santner, M. Godec, H. Bischof, On-line random forests, in: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, 2009, pp. 1393–1400.

[42] Y. Wu, B. Shen, H. Ling, Online robust image alignment via iterative convex optimization, in: Comp. Vis. Patt. Recognition, 2012, pp. 1808–1814.

[43] H. K. Galoogahi, T. Sim, S. Lucey, Multi-channel correlation filters, in: Int. Conf. Computer Vision, 2013, pp. 3072–3079.

[44] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Comp. Vis. Patt. Recognition, Vol. 1, 2005, pp. 886–893.

[45] F. Pernici, A. Del Bimbo, Object tracking by oversampling local features, IEEE Trans. Pattern Anal. Mach. Intell. 36 (12) (2013) 2538–2551.

[46] X. Yang, Q. Xiao, S. Wang, P. Liu, Real-time tracking via deformable structure regression learning, in: Proc. Int. Conf. Pattern Recognition, 2014, pp. 2179–2184.

[47] S. Duffner, C. Garcia, PixelTrack: a fast adaptive algorithm for tracking non-rigid objects, in: Int. Conf. Computer Vision, 2013, pp. 2480–2487.

[48] B. Schölkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, 2002.

[49] D. Bailey, Sub-pixel estimation of local extrema, in: Image and Vision Computing New Zeland, 2003, pp. 414–419.

[50] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Trans. Pattern Anal. Mach. Intell. 32 (9) (2010) 1627–1645.

[51] A. Diplaros, N. Vlassis, T. Gevers, A spatially constrained generative model and an em algorithm for image segmentation, IEEE Trans. Neural Networks 18 (3) (2007) 798 – 808.

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

[53] I. Griva, S. G. Nash, A. Sofer, Linear and Nonlinear Optimization, Second Edition, Siam, 2009.

[54] E. K. P. Chong, S. H. Żak, An Introduction to Optimization, John Wiley and sons, 2013.

[55] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, L. Cehovin, A novel performance evaluation methodology for single-target trackers, arXiv:1503.01313.