Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tim Rejc

Dolgoročno sledenje objektov z uporabo predlaganja regij

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: izr. prof. dr. Matej Kristan

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi obdelajte problem sledenja objektov v sekvenci slik. Osredotočite se na problem dolgoročnega sledenja, kjer sledilniku podate edini učni primer objekta v prvi sliki, sledilnik pa mora poročati lokacijo izbranega objekta v vseh naslednjih slikah. Ker objekt lahko med sledenjem izgine iz vidnega polja kamere, mora sledilnik implementirati tako ugotavljanje prisotnosti izbranega objekta, kot tudi detekcijo ob ponovni pojavitvi. Hkrati pa naj se sledilnik prilagaja trenutnemu izgledu objekta. Za metodološki okvir izberite konvolucijske nevronske mreže in postopke predlaganja regij, ki so se v zadnjih letih dobro obnesli na problemih detekcije objektov. Delovanje sledilnika analizirajte na standardnih podatkovnih zbirkah.

Zahvalil bi se mentorju izr. prof. dr. Mateju Kristanu za pomoč in strokovno usmerjanje skozi izdelavo te diplomske naloge. Zahvaljujem se tudi družini in kolegom, ki so me spodbujali, podpirali in mi pomagali ostati na nogah.

Kazalo

Povzetek

Naslov: Dolgoročno sledenje objektov z uporabo predlaganja regij

Avtor: Tim Rejc

V diplomski nalogi naslavljamo problem sledenja poljubnega objekta na sekvenci slik. Predlagamo dolgoročni sledilnik, ki temelji na uporabi siamske konvolucijske nevronske mreže. Izbrani objekt sledenja na sliki detektiramo s predlogo, kjer ujemanje merimo s križno korelacijo na vsaki točki iskalne slike. Predlogo inicializiramo na prvi sliki sekvence, kjer na vhod konvolucijske nevronske mreže podamo izsek slike, ki vsebuje objekt sledenja. Po vsaki lokalizaciji sledenega objekta napovemo ali je prišlo do odpovedi sledenja. Predlagamo dve metodi prilagajanja na vizualno predstavitev tarče. Prva posodablja predlogo, druga pa prilagaja parametre konvolucijske nevronske mreže na način, da je korelacija s predlogo in trenutno lokalizacijo večja. Izvedemo dve analizi, kjer na zbirki LTB35 [21] izmerimo uspešnost dolgoročnega sledena na modifikacijah sledilnika. Pri prvi analizi ugotovimo dobro nastavitev števila in oblik predlaganih regij. Pri drugi analizi pa testiramo uspešnost predlaganih metod posodabljanja vizualnega modela, kjer brez posodabljanja dosežemo F-mero 0.34, s posodabljanjem predloge 0.22, s prilagajanjem parametrov mreže 0.38 in z obema 0.20. Implementaciji, kjer prva sproti uči mrežo in druga brez posodabljanja predloge ali učenja mreže primerjamo z metodami objavljenimi na tekmovanju VOT-LT2018 [16], kjer se slednja uvrsti na 12. mesto, prva pa na 11. mesto.

Ključne besede: sledilnik, konvolucijske nevronske mreže, računalniški vid.

Abstract

Title: Long-term tracking using region proposals

Author: Tim Rejc

In this thesis we address the problem of tracking an arbitrary object in a sequence of images. We propose a long-term tracker based on the use of Siamese convolutional neural networks. For detection, we use a template with which we compute cross correlation on every point of the search image to find the best matching region. The template is initialized on the first frame, where we crop the image so that it represents only the tracking object and input it to the convolutional neural network. After each localization the tracker detects if tracking has failed. We propose two online methods of updating the visual model. One updates the template and the other fine tunes the parameters of the network. We carried out two analysis, where we measure long-term tracking performance on dataset LTB35 [21] on modifications of our tracker. With the first analysis we find out what is a good setting for generating region proposals. The purpose of the second analysis is to test the proposed methods for updating the visual model. We find out that without updating the visual model, our tracker achieves F-measure of 0.34, when updating the template 0.22, when fine tuning 0.38 and with both methods we get 0.20. Finally we compared the performance of our tracker with the trackers submitted in the VOT-LT2018 [16] challange, and achieved 11^{th} place when fine tuning and 12^{th} without fine tuning or updating the template.

Keywords: tracker, convolutional neural networks, computer vision.

Poglavje 1

Uvod

Sledenje objektov je problem na področju računalniškega vida, ki se ukvarja z lokalizacijo enega ali več objektov na sekvenci slik. Lokalizacija objekta je proces, kjer na sliki najdemo pozicijo in velikost regije, ki vsebuje iskani objekt. Rešitve tega problema služijo na področju razvoja avtonomnih vozil [25], video nadzora [33, 11], obogatene resničnosti [22, 13], video kompresije [14] in medicine [32]. Lokalizacija in sledenje objektov sta aktualna problema računalniškega vida, zaradi česa je bilo razvitih že precej metod [10], ki ta problem rešujejo.

Ko razvijamo sledilnik, ki je namenjen za neko specifično nalogo, je ključno, da izberemo take metode, kjer bodo hitrost, natančnost in robustnost primerne za ta scenarij. Na primer, pri sledenju objekta z namenom stabilizacije videa, sta natančnost in robustnost sledenja pomembnejši kot pa hitrost. Vendar izbira pravih metod za neko specifično nalogo ni vedno trivialna, saj težko predvidimo, ne da bi testirali, kako se metode izkažejo v danem okolju. V tem primeru si lahko pomagamo z rezultati testiranja sledilnikov [3, 24], katerih naloga je detekcija in sledenje poljubnih objektov na katerikoli sekvenci slik. Tak sledilnik mora biti za uspešno sledenje prilagodljiv na deformacije objekta in druge težavne scenarije. Recimo, lahko se zgodi, da je video, na katerem sledimo objektu, slabe kakovosti ali pa vsebuje šum. Prav tako tresljaji kamere lahko popačijo sliko. Ali pa je posnetek v

vremenu, kjer je slaba vidljivost, ali se med sledenjem spremeni svetlost slike, ali pa pojavi delna, mogoče popolna, okluzija objekta.

Obstajajo tekmovanja, na primer VOT [15] in OxUvA [29], kjer se te sledilniki primerjajo na določeni zbirki sekvenc. Te množice vsebujejo sekvence z različnimi scenariji, tako da je ocena uspešnosti sledilnika čim bolj točna, saj je prilagodljivost različnim razmeram pomembno merilo kvalitete sledilnika.

V diplomski nalogi naslavljamo problem robustnega dolgoročnega sledenja z uporabo konvolucijskih nevronskih mrež. Implementirali smo svoj sledilnik poljubnih objektov, kjer smo uporabili bolj sodobne metode sledenja in detekcije.

1.1 Pregled področja

V zadnjih letih se je pojavilo veliko metod za detekcijo in sledenje, ki na nek način uporabljajo konvolucijske nevronske mreže. Eden glavnih razlogov je pojavitev velikih učnih množic, kot na primer ILSVRC [27], na katerih se konvolucijske nevronske mreže lahko učijo.

Eden od načinov za reševanje problema sledenja objektov je z iskanjem podobnosti na sliki s predlogo. Tak pristop predlagajo v članku [3], kjer predstavijo uporabo siamske arhitekture za iskanje objekta, s sledilnikom SiamFC. Uporabijo metodo detekcije, kjer na vhod siamske konvolucijske nevronske mreže podajo predlogo tarče in iskalno sliko. Nato pa izračunajo mapo podobnosti z drsenjem predloge po iskalni sliki in s križno korelacijo izračunajo ujemanje. Tam kjer je odziv križne korelacije največji, je ujemanje predloge in iskalne slike največje, kar se tretira za pozicijo največje verjetnosti pristonosti tarče. Ta pozicija se zato vzame kot položaj tarče v trenutni sliki.

Sledilnika MDNet [24] in SO-DLT [31] uporabita naučeno konvolucijsko mrežo, ki zna oceniti, kateri deli slike vsebujejo objekte. Med sledenjem pa naučita detektor s prilagajanjem uteži mreže na klasifikacijo specifičnegai objekta, tako da vzorce objekta skozi sekvenco uporabita za učne primere.

Ta pristop imenujemo sledenje-učenje-detekcija [12]. Vendar zaradi računske zahtevnosti prilagajanja uteži konvolucijske nevronske mreže je sledenje s tem pristopom dokaj počasno.

Med sledenjem lahko sledilnik tudi sam zaznava odpoved sledenja. Takim sledilnikom pravimo dolgoročni sledilniki. V članku [20] predlagajo metodo ocenjevanja moči detekcije. Predstavijo nam sledilnik FCLT, ki je polno korelacijski dolgoročni sledilnik. Ta med sledenjem izračuna moč detekcije glede na razmerje med odzivom križne korelacije in njegovega maksimuma. Nato primerja izračunano moč detekcije s povprečno močjo detekcije zadnjih nekaj slik in oceni zanesljivost detekcije. Dolgoročnost pri sledenju pomaga v primerih, ko objekt sledenja izgine iz slike in ga ob pojavitvi moramo ponovno detektirati.

Robustni sledilniki med sledenjem prilagajajo dimenzije okvirja okoli objekta sledenja. Tako ima sledilnik boljšo vizualno predstavo objekta in ga posledično lažje detektira. V članku [26] predstavijo metodo mreže predlaganih regij, ki z enim prehodom skozi konvolucijsko nevronsko mrežo ocenimo lokacijo in okvir objekta. Sledilnik SiamRPN [18] uporablja naučeno mrežo predlaganih regij v sodelovanju s siamsko mrežo. Deluje na principu sidriščnih regij, ki so porazdeljene po iskalni sliki. Mreža predlaganih regij vsebuje dve veji, klasifikacijsko in regresijsko. Klasifikacijska veja oceni verjetnost, ali se v neki sidriščni regiji nahaja iskani objekt. Regresijska veja pa oceni skalo in zamik za vsako sidriščno regijo. Nato pa uporabijo metodo tlačenja ne-maksimumov (*angl. non-maximum suppression*), ki izbere tisto sidriščno regijo, ki objekt najbolje pokriva.

1.2 Prispevki

Glavni prispevek te diplomske naloge je razvoj dolgoročnega sledilnika in študija uporabljenih metod. Uporabili smo bolj aktualne metode detekcije in sledenja, ki smo jih implementirali v naš sledilnik. Dolgoročni sledilnik, ki ga predlagamo, uporabi siamsko konvolucijsko nevronsko mrežo, kateri na

vhod podamo izgled sledenega objekta in iskalno sliko, da dobimo predlogo in iskalni tenzor. Ujemanje izračunamo s križno korelacijo in tam kjer je korelacija največja predpostavimo, da se nahaja tarča. Ko tarčo detektiramo, predlagamo metodo prilagajanja dimenzij okvirja, ki uporabi dimenzije tiste predlagane regije, ki se najbolje ujema s predlogo. Za dolgoročno funkcionalnost uporabimo metodo, ki meri zanesljivost lokalizacije na trenutni sliki. Če je zanesljivost slaba, sledilnik zazna odpoved. Če pa je zanesljivost dobra pa predlagamo in analiziramo dve metodi za posodabljanje vizualnega modela na barvno predstavitev tarče in ozadja.

1.3 Struktura naloge

Diplomska naloga je strukturirana na naslednji način. Poglavje 2 opisuje osnovne metode, ki smo jih uporabili pri implementaciji sledilnika. V Poglavju 3 podrobno opišemo delovanje posameznih delov sledilnika in delovanje celotnega sledilnika. V Poglavju 4 opišemo postopek testiranja našega sledilnika, kakšne parametre smo uporabili pri testiranju,opišemo rezultate dveh analiz uspešnosti dolgoročnega sledenja različnih modifikacij sledilnika in primerjamo rezultate našega sledilnika z metodami iz tekmovanja VOT-LT2018 [16].

Poglavje 2

Uporabljene metode

V tem poglavju bomo predstavili ključne metode, ki smo jih uporabili pri izdelavi našega sledilnika. V Poglavju 2.1 opišemo glavne sestavne dele konvolucijske nevronske mreže in kako jih učimo. V Poglavju 2.2 pa na kratko opišemo merjenje podobnosti dveh signalov.

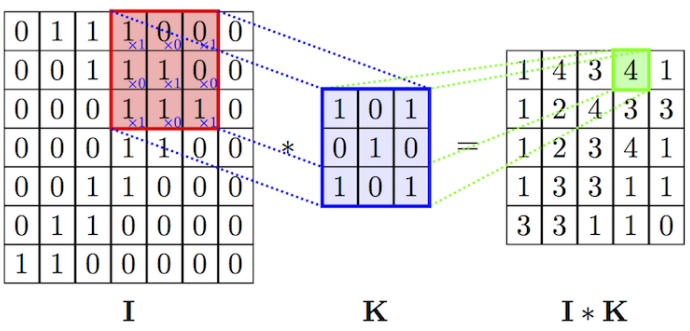
2.1 Konvolucijske nevronske mreže

Konvolucijske nevronske mreže (*angl. convolutional neural networks*) so v času pisanja tega dela zelo popularne na področju računalniškega vida. Zelo dobro se izkažejo pri klasifikaciji objektov na sliki. To je leta 2012 na tekmovanju ILSVRC [27] (ImageNet Large-Scale Visual Recognition Challenge) dokazala ekipa, ki je zmagala s prvo globoko arhitekturo konvolucijske nevronske mreže imenovano AlexNet [17]. Da konvolucijske nevronske mreže sploh lahko klasificirajo objekte na sliki, jih moramo najprej naučiti na neki anotirani slikovni množici. Učenje je zelo časovno zahtevno. Vendar večja kot je učna množica, bolj splošno se bo mreža naučila klasifikacije.

V tem poglavju bomo opisali osnove sestavnih delov konvolucijskih nevronskih mrež, principe učenja in nato si pogledali še prilagajanje parametrov zadnjih nekaj slojev (*angl. fine-tuning*).

2.1.1 Konvolucijski nivo

Konvolucijski nivo je primarni sestavni del konvolucijske nevronske mreže. Kot vhod prejme matriko z dimenzijami $W_1 \times H_1 \times D_1$ in pa štiri hiper parametre. Parametri označujejo število filtrov K, velikost filtrov F, velikost koraka drsečega okna (*angl. stride*) S in pa velikost dodane ničelne obrobe (*angl. zero padding*) P. Ničelna obroba se uporablja za ohranjanje višine in širine na izhodu konvolucijskega nivoja. Filtri so matrike velikosti $F \times F \times D_1$ in so sestavljeni iz uteži, katere nastavljamo pri učenju. Celoten sloj tako vsebuje $F^2 \cdot D_1 \cdot K$ uteži in še K vrednosti pristrankosti (*angl. bias*).



Slika 2.1: Prikazuje konvolucijo med vhodom **I** in filtrom **K**. Slika je vzeta iz [30]

Med izvajanjem vsak filter drsi po širini in višini vhodne matrike in s konvolucijo ali križno korelacijo, odvisno od implementacije, izračuna dvodimenzionalno matriko, ki predstavlja rezultat enega filtra. Rezultate vseh filtrov združimo v tridimenzionalno izhodno matriko velikosti $W_2 \times H_2 \times D_2$. Kjer so [19]:

$$W_2 = \frac{W_1 - F + 2P}{S + 1}, \tag{2.1}$$

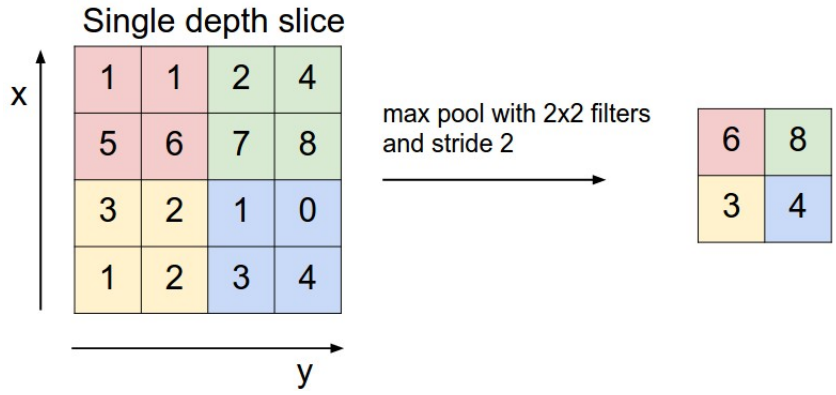
$$H_2 = \frac{H_1 - F + 2P}{S + 1}, \tag{2.2}$$

$$D_2 = K. \tag{2.3}$$

2.1.2 Združevalni nivo

Združevalni nivo se običajno nahaja za konvolucijskim nivojem in kot vhod prejme izhodne značilnice predhodnega sloja. Funkcija tega sloja je, da zmanjša višino in širino vhodnega tenzorja X, velikosti $W_1 \times H_1 \times D_1$, na podlagi določenega pravila združevanja. To zmanjša število parametrov, kar posledično zmanjša časovno zahtevnost in pa preprečuje verjetnost pojavitve prenasitosti med učenjem.

Sloj sprejme dva parametra. Prvi predstavlja velikost drsečega okna F, drugi pa velikost koraka drsečega okna S. Poznamo več pravil združevanja na podlagi katerih izbiramo vrednosti izhodnega bloka. Eni izmed bolj uporabljenih so na primer združevanje po maksimumu, povprečju in pa L2 normi.



Slika 2.2: Prikazuje primer združevanja po maksimumu. Slika je vzeta iz [19]

Primer, kjer uporabljamo združevanje po maksimumu na vhodni matriki X. Drseče okno potuje po višini in širini vhodne matrike in s formulo

$$Y_{i,j} = \max_{0 \leq x,y \leq S} (X_{i+y,j+x}) \tag{2.4}$$

določi elemente izhodne matrike Y, kot je prikazano na Sliki 2.2. Velikost

matrike Y se preslika po enačbah [19]:

$$W_2 = \frac{W_1 - F}{S} + 1, \tag{2.5}$$

$$H_2 = \frac{H_1 - F}{S} + 1, \tag{2.6}$$

$$D_2 = D_1. \tag{2.7}$$

Pogosto se uporablja združevalni nivo s parametri $F = 2$ in $S = 2$, tako da dobimo $W_2 = \frac{W_1}{2}$ in $H_2 = \frac{H_1}{2}$. Kar nam zmanjša število parametrov na naslednjih nivojih za 75%.

2.1.3 Polno povezan nivo

Polno povezan nivo vsebuje N nevronov, ki so povezani z vsemi nevroni prejšnjega nivoja. Če prejšnji nivo vsebuje N_{prej} nevronov, potem povezan nivo vsebuje $N_{prej} \cdot N$ parametrov. Izhod tega nivoja je vektor dolžine N.

V konvolucijskih nevronskih mrežah se te nivoje uporablja na koncu mreže za klasifikacijo. Izhod zadnjega povezanega nivoja je vektor, kjer vsak element predstavlja klasifikacijski razred, vrednost elementa pa ocenjeno verjetnost klasifikacije razreda.

2.1.4 Aktivacijska funkcija

Aktivacijske funkcije so nelinearne in se aplicirajo izhodnim elementom konvolucijskega nivoja. Na vhod sprejme po en element izhodnega bloka in ga preslika z neko nelinearno matematično funkcijo. Nelinearnost je pomemben faktor, sicer bi se nevronska mreža obnašala kot linearen klasifikator. Druga pomembna lastnost funkcije pa je odvedljivost, da lahko izračunamo gradient.

Trenutno aktualne nelinearne funkcije so na primer hiperbolični tangens, sigmoid in rektificirana linearna enota oz. ReLU [23] (*angl. Rectified Linear Unit*). Slednja je definirana kot

$$f(x) = \max(0, x), \tag{2.8}$$

kar predstavlja upragovanje elementa x nad nič. ReLU se v praksi izkaže za precej uporabno aktivacijsko funkcijo in je zato popularna izbira pri mnogih arhitekturah nevronskih mrež.

2.1.5 Arhitektura

Konvolucijsko nevronsko mrežo sestavljajo nivoji različnih funkcionalnosti. Zaporedje, število in struktura nivojev določa arhitektura mreže. Najbolj pogost način strukturiranja konvolucijske nevronske mreže je, da po nekaj konvolucijskih slojih umestimo združevalni sloj, zadnji nivoji mreže pa so povezani nivoji.

Arhitektura nevronske mreže vpliva na hitrost in uspeh učenja. Ponavadi se nam ni potrebno izmisliti svoje arhitekture, temveč uporabimo tiste, ki so že testirane na množici slik ImageNet [8]. Poznamo veliko različnih arhitektur konvolucijskih nevronskih mrež, kot na primer AlexNet [17], VGGNet [28] in ResNet [9].

2.1.6 Kriterijska funkcija

Kriterijska funkcija (*angl. loss function*) oceni uspešnost delovanja nevronske mreže. Uporablja se pri učenju in nam pove odstopanje ocene nevronske mreže od resničnega podatka (*angl. ground truth*).

Izbira prave kriterijske funkcije temelji na problemu, ki ga nevronska mreža rešuje. Če je problem klasifikacijski, lahko uporabimo funkcijo Softmax s križno entropijo, ki je definirana kot

$$E_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_{j=1}^{C} e^{\mathbf{f}_j}}\right), \tag{2.9}$$

kjer je f_{y_i} pripravljen resnični podatek, $\mathbf{f}$ pa izhodni vektor $\mathbf{f} \in \mathcal{R}^C$ zadnjega sloja nevronske mreže, $\mathbf{f}_j$ pa je j-ti element vektorja $\mathbf{f}$.

S kriterijsko funkcijo preslikamo vsak izhod nevronske mreže med učenjem.

Po prehodu vseh podatkov iz učne množice izračunamo povprečje z enačbo

$$E = \frac{1}{N} \sum_i E_i, \tag{2.10}$$

kjer je N število podatkov v učni množici. E predstavlja izgubo učenja in večji kot je, večje je odstopanje med oceno mreže in resničnim podatkom.

2.1.7 Učenje nevronskih mrež

Za učenje moramo imeti pripravljeno anotirano množico podatkov, ki ji rečemo učna množica. Med učenjem iz učne množice vzamemo tenzor ali vektor $\mathbf{x}$ in ga podamo nevronski mreži na vhod. Kot izhod dobimo $\mathbf{y}$, ki je preslikava vhoda glede na parametre mreže. Cilj učenja je, da parametre mreže prilagodimo, da bo izhod mreže $\mathbf{y}$ čim boljši približek resničnemu podatku. Nenaučeni mreži prilagajamo parametre na vseh nivojih, že naučeni mreži pa ponavadi samo na določenih nivojih.

Učenje nevronske mreže opredelimo kot optimizacijski problem, zato za reševanje uporabimo optimizacijske algoritme [5]. Najbolj popularen tak algoritem za učenje globokih mrež je gradientni spust. To je iterativen algoritem, ki najde minimum neke funkcije. Nevronsko mrežo lahko predstavimo kot funkcijo $g(\mathbf{x}, \mathbf{w})$, kjer je $\mathbf{x}$ vhod, $\mathbf{w}$ pa so uteži oz. parametri mreže. Gradientni spust ob vsaki iteraciji izračuna gradient $\nabla E(g(\mathbf{x}, \mathbf{w}))$ in na podlagi tega prilagodi parametre $\mathbf{w}$, na način, da je za vhod $\mathbf{x}$ napaka zmanjšana.

Gradient izračunamo z uporabo metode vzvratnega razširjanja (*angl. backpropagation*). Metoda temelji na pravilu veriženja (*angl. chain rule*) [2]. To pravilo se upošteva, ker so nevronske mreže strukturirane po nivojih in lahko funkcijo $g(\mathbf{x}, \mathbf{w})$ definiramo kot kompozicijo

$$g = f_n \circ f_{n-1} \circ \cdots \circ f_1, \tag{2.11}$$

kjer je f_i preslikava enega nivoja. Vsak nivo f_i na vhod prejme izhod prejšnjega nivoja y_{i-1} in ga preslika na podlagi vsebovanih parametrov $\mathbf{w}_i$. Izhod i-tega nivoja je tako definiran kot

$$y_i = f_i(\mathbf{y}_{i-1}, \mathbf{w}_i). \tag{2.12}$$

Tako vzvratno razširjanje od izhodnega do vhodnega nivoja, za vsak vhod $\mathbf{x}$, rekurzivno izračuna gradiente za vse parametre z uporabo pravila veriženja.

Vpliv w_{ij}, ki je j-ti parameter na i-tem nivoju, na izhod $\mathbf{y}$ izračunamo kot odvod kriterijske funkcije E v odvisnosti od w_{ij}, po enačbi

$$o_{ij} = \frac{dE}{dw_{ij}}. \tag{2.13}$$

Če velja $o_{ij} > 0$, se pri povečanju vrednosti w_{ij} poveča tudi E, če pa $o_{ij} < 0$, se pri večanju w_{ij} vrednost E zmanjša. Zato se pri gradientnem spustu pomikamo proti negativu gradienta in prilagoditev parametra w_{ij} izračunamo po enačbi

$$\Delta w_{ij} = -\lambda o_{ij}, \tag{2.14}$$

kjer $\lambda > 0$ in predstavlja hitrost učenja (*angl. learning rate*), ki jo nastavimo kot hiper parameter. Ta hiper parameter določa hitrost spusta po gradientu in je lahko statična ali pa se med učenjem po nekem pravilu spreminja.

Na hitrost in uspeh učenja precej vpliva pravilna nastavitev hiper parametrov in arhitektura nevronske mreže. Čeprav gradientni spust ne zagotavlja, da najde globalni minimum, se mreža v praksi lahko dobro izkaže tudi, če smo dosegli lokalni minimum. Problem se pojavi, če učenje pustimo predolgo in se naš model začne učiti specifične lastnosti iz učne množice in ne generalizira. Takrat se pojavi problem prekomernega prileganja (*angl. overfitting*) in bodo rezultati na poljubnih vhodih slabi. Zato je dobra praksa, da imamo pripravljeno še množico za testiranje, ki je manjša od učne množice in na kateri po učni iteraciji preverimo napako. Učna in testna množica morata vsebovati med seboj različne elemente.

2.1.8 Fino prilagajanje nivojev

To je metoda, ki prilagodi zadnjih nekaj nivojev naučene konvolucijske nevronske mreže za neko bolj specifično nalogo. V poštev pride, ko si lahko za reševanje našega problema pomagamo z že naučeno mrežo in nimamo velike učne množice. Takrat bi učenje celotne konvolucijske nevronske mreže povzročilo prekomerno prileganje parametrov.

Prvih nekaj slojev običajno ne prilagajamo, ker zaznavajo bolj splošne lastnosti, kot recimo robove in barve. Prilagajanje poteka kot navadno učenje, le da prilagajamo parametre na specifičnih nivojih. Uporabimo gradientni spust ali kateri drug optimizacijski algoritem in kriterijsko funkcijo, ki oceni napovedane vrednosti mreže. Vendar tudi ta metoda lahko povzroči prekomerno prileganje, če imamo premajhno množico ali pa preveliko hitrost učenja.

2.2 Merjenje podobnosti med signali

Za merjenje razdalje med dvema signaloma $\mathbf{f}$ in $\mathbf{g}$ obstaja več metod. Te metode lahko uporabimo za prepoznavanje vzorcev ali podobnosti. Primer metode je merjenje Evklidske razdalje, ki je definirana s formulo

$$d(\mathbf{f}, \mathbf{g}) = \sqrt{\sum_{i=1}^{n} (\mathbf{f}_i - \mathbf{g}_i)^2}. \tag{2.15}$$

Manjši kot je $d(\mathbf{f}, \mathbf{g})$ bolj sta si signala $\mathbf{f}$ in $\mathbf{g}$ podobna. Pogosto uporabljena metoda za zaznavanje podobnosti na večdimenzionalnih matrikah, t.i. tenzorjih, je uporaba križne korelacije. Križna korelacija je definirana s formulo

$$(\mathbf{f} \star \mathbf{g})[n] \stackrel{def}{=} \sum_{m=-\infty}^{\infty} \mathbf{f}^*[m]\mathbf{g}[m+n], \tag{2.16}$$

kjer je n zamik signala $\mathbf{g}$. Križno korelacijo izračunamo na intervalu zamikov signala $\mathbf{g}$, da dobimo korelacijo na vsaki točki. Operacija je podobna konvoluciji ter večja kot je vrednost $(\mathbf{f} \star \mathbf{g})$, bolj sta signala podobna. Dve meri, ki sta pravtako aktualni pri metodah sledenja sta Hellingerjeva in Bhattacharyyajeva razdalja [6].

Poglavje 3

Opis sledilnika

V tem poglavju podrobno opišemo vse elemente predlaganega sledilnika SiamRP-LT. Vse vizualne lastnosti kodiramo z modificirano konvolucijsko nevronsko mrežo VGG-16, ki jo opišemo v Poglavju 3.1. V Poglavju 3.2 opišemo postopek lokalizacije objekta sledenja na sliki. Metodo ocenjevanja zanesljivosti detekcije, s čim zaznavamo odpoved sledenja, smo opisali v Poglavju 3.4. V Poglavju 3.3 opišemo predlagane metode za posodabljanje vizualnega modela. Delovanje celotnega sledilnika povzamemo v Poglavju 3.5.

3.1 Modificirana mreža VGG-16

Konvolucijsko nevronsko mrežo VGG-16 [28] so leta 2014 predstavili na tekmovanju ILSVRC [27]. Mreža je bila naučena na slikovni množici ImageNet [8], ki vsebuje 1000 klasifikacijskih razredov. Na tekmovanju se je ekipa uvrstila na drugo mesto v klasifikaciji in na prvo mesto v lokalizaciji.

Vsebuje 16 nivojev z naučljivimi parametri, od tega 13 konvolucijskih in 3 polno povezane nivoje. Konvolucijski nivoji vsi uporabljajo konvolucijske filtre velikosti 3×3. Aktivacijska funkcija ReLU se aplicira na izhod vsakega konvolucijskega nivoja. Prav tako mreža vsebuje 5 združevalnih nivojev, ki imajo drseče okno velikosti 2×2 in se pomikjo s koraki po 2 mesti. Ozko grlo mreže so polno povezani nivoji, ki vsebujejo zelo veliko parametrov, zaradi

česa se časovna in prostorska zahtevnost prehoda skozi mrežo močno poveča. Celotna mreža vsebuje 140 milijonov parametrov, od tega jih je približno 120 milijonov samo v polno povezanih slojih [19].

Mi uporabimo za kodiranje vizualnih lastnosti izhod 13. nivoja. Tako da spustimo vse polno povezane sloje, 3 konvolucijske sloje in 2 združevalna sloja. Izhodni tenzor ima širino in višino osemkrat manjšo od vhoda in globino vedno 512. Arhitektura, ki jo uporabimo je prikazana v Tabeli 3.1.

Zap. št.	**Tip nivoja**	**Parametri nivoja**
1	Konvolucijski	K=64, F=3, S=1, P=1
2	Konvolucijski	
3	Združevalni	F=2, S=2
4	Konvolucijski	K=128, F=3, S=1, P=1
5	Konvolucijski	
6	Združevalni	F=2, S=2
7	Konvolucijski	K=256, F=3, S=1, P=1
8	Konvolucijski	
9	Konvolucijski	
10	Združevalni	F=2, S=2
11	Konvolucijski	K=512, F=3, S=1, P=1
12	Konvolucijski	
13	Konvolucijski	

Tabela 3.1: Arhitektura modificirane konvolucijske nevronske mreže VGG-16, katero uporabimo pri implementaciji sledilnika iz Poglavja 3.5.

3.2 Detekcija tarče s predlogo

Naš pristop k detekciji objekta sledenja temelji na iskanju podobnosti med značilkam iskalne slike in predlogo, ki sta kodirani z modificirano VGG-16, katere strukturo smo opisali v Poglavju 3.1 in jo prikažemo v Tabeli 3.1..

Postopek je prilagoditev metode detekcije, kot jo predstavijo v članku [3].

Na vhod v t-ti iteraciji dobimo sliko $\mathbf{I}_t$, na kateri se nahaja sledeni objekt in $\mathbf{T}_t$ je predloga v tej iteraciji. Na sliko in predlogo apliciramo transformacijo θ. Ta transformacija predstavlja kodiranje vizualnih lastnosti z modificirano konvolucijsko nevronsko mrežo VGG-16. Zaradi slojev združevanja, je izhod mreže osemkrat manjši po višini in širini. Tako $\theta(\mathbf{I}_t)$ predstavlja tridimenzionalni tenzor značilk, z dimezijami $W_t \times H_t \times 512$, ki ga dobimo pri prehodu slike $\mathbf{I}_t$ skozi konvolucijsko nevronsko mrežo. $\theta(\mathbf{T}_t)$ ima fiksno vnaprej določeno velikost $T_d \times T_d \times 512$, W_t in H_t pa sta lahko poljubna, vendar mora veljati $W_t \geq T_d$ in $H_t \geq T_d$. Ker je velikost predloge fiksna, sliki prilagodimo dimenzije, da bo oblika objekta po transformaciji θ enaka obliki predloge. Zato definiramo faktor skaliranja z enačbo

$$s(o_d) = \frac{8T_d}{o_d}, \tag{3.1}$$

kjer je o_d velikost dimenzije, ki jo prilagajamo. Skaliramo na podlagi okvirja objekta iz prejšnje slike, s širino o_w in višino o_h, tako da sliki širino skaliramo s faktorjem $s(o_w)$ in višino s $s(o_h)$.

Naj bo $\theta(\mathbf{I}_t)^{(i,j)}$ izsek iz $\theta(\mathbf{I}_t)$, ki ima zgornji levi kot v točki (i, j) in enake dimenzije kot predloga $\mathbf{T}_t$. Korelacijo med $\theta(\mathbf{I}_t)^{(i,j)}$ in $\theta(\mathbf{T}_t)$ izmerimo s funkcijo $f(\theta(\mathbf{T}_t), \theta(\mathbf{I}_t)^{(i,j)})$, ki izračuna križno korelacijo med obema vhodoma in je definirana kot

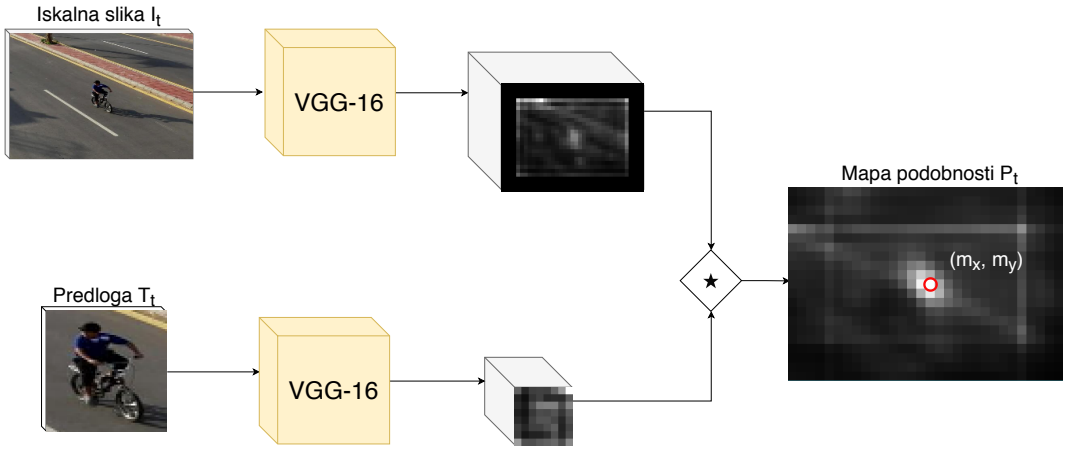
$$f(\mathbf{x}, \mathbf{z}) = \mathbf{x} \star \mathbf{z}. \tag{3.2}$$

Funkcija f vrne mero ujemanja kot skalar in večja kot je vrednost $f(\mathbf{x}, \mathbf{z})$ bolj sta si vhoda $\mathbf{x}$ in $\mathbf{z}$ podobna.

Korelacijo izmerimo na vsaki točki $\theta(\mathbf{I}_t)$ s predlogo $\theta(\mathbf{T}_t)$. Tako dobimo mapo podobnosti $\mathbf{P}_t$, ki je dvodimenzionalna matrika z enako širino in višino kot $\theta(\mathbf{I}_t)$. Dimenzije ohranjamo tako, da tenzor $\theta(\mathbf{I}_t)$ obdamo z ničelno oblogo velikosti $\lfloor \frac{T_d}{2} \rfloor$. Če v prejšnji iteraciji ni veljalo (3.16) iz Poglavja 3.4, kar pomeni, da nismo zaznali odpovedi, potem na matriko $\mathbf{P}_t$ apliciramo še Gaussov filter s standardno deviacijo σ, ki ima center v točki, kjer se je nahajal objekt na prejšnji sliki. Proces izračuna mape podobnosti $\mathbf{P}_t$ grafično prikažemo na Sliki 3.1.

V točki, kjer je vrednost $\mathbf{P}_t$ največja, smatramo, da se nahaja iskani objekt. Ko to točko imamo, jo moramo preslikati na prvotno sliko $\mathbf{I}_t$, zaradi pomanjšanja dimenzij pri transformaciji $\theta(\mathbf{I}_t)$. To storimo tako, da pomnožimo koordinati točke maksimuma m_x in m_y s faktorjem pomanjšanja dimenzij,

$$(x, y) = (8m_x, 8m_y). \tag{3.3}$$



Slika 3.1: Slika prikazuje proces izračuna mape podobnosti $\mathbf{P}_t$ s predlogo $\mathbf{T}_t$ na sliki $\mathbf{I}_t$

3.3 Prilagajanje vizualnega modela

V tem poglavju podrobno predstavimo predlagane metode prilagajanja vizualnega modela. Ker se med sledenjem oblika sledenega objekta spreminja in ker se lahko oddalji ali približa, smo implementirali način za prilagajanje okvirja in ga opišemo v Poglavju 3.3.1. Poleg tega pa se med sledenjem spreminja tudi ozadje in pa izgled sledenega objekta. Iz tega razloga predlagamo dve metodi za prilagajanje na te spremembe. Prvo, ki deluje na principu posodabljanja predloge, opišemo v Poglavju 3.3.2, drugo, ki pa fino prilagaja parametre mreže na način, da poveča korelacijo med predlogo in trenutno lokalizacijo, pa opišemo v Poglavju 3.3.3.

3.3.1 Prilagajanje dimenzij okvirja

Objekt, ki mu sledimo, lahko spremeni velikost ali obliko. Zato, da je sledilnik prilagodljiv na te spremembe, med sledenjem prilagajamo dimenzije okvirja. Pristop deluje tako, da generiramo n_a predlaganih regij $\mathbf{a}_i^\phi = [w_i^\phi, h_i^\phi]$ različnih velikosti in razmerij stranic, na način kot je prikazan na Sliki 3.2. Za vsako to regijo izrežemo izsek iz enkodirane iskalne slike preko VGG-16 iz Poglavja 3.1, in za dimenzije novega okvirja izberemo tisto, katere izsek doseže največjo korelacijo s predlogo $\theta(\mathbf{T}_t)$. Metoda generiranja teh regij izhaja iz članka [26].

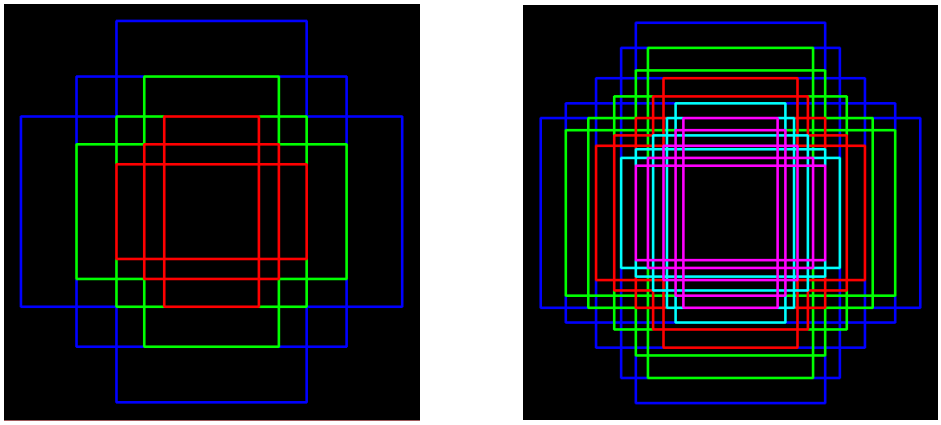
Pri generiranju imamo vnaprej pripravljen seznam $\mathbf{R}_s = \{r_s^{(i)}\}_{i=1:N_s}$, ki hrani razmerja stranic regij in pa $\mathbf{R}_v = \{r_v^{(i)}\}_{i=1:N_v}$, ki hrani razmerja velikosti. Predlagane regije delimo v sklope glede na velikost. Število teh sklopov je enako številu elementov seznama $\mathbf{R}_v$. Da dobimo ploščino predlaganih regij nekega sklopa, pomnožimo ploščino okvirja iz prejšnje iteracije z elementi iz $\mathbf{R}_v$. Vsak sklop vsebuje N_s predlaganih regij, ki imajo taka razmerja širine in višine, kot so definirane v $\mathbf{R}_s$. Tako širino w_i^ϕ in višino h_i^ϕ predlagane regije $\mathbf{a}_i^\phi$, ki je i-ta predlagana regija v ϕ-tem sklopu, izračunamo z enačbama

$$w_i^\phi = \mathbf{R}_v^{(\phi)} p_{t-1} \mathbf{R}_s^{(i)}, \tag{3.4}$$

$$h_i^\phi = \mathbf{R}_v^{(\phi)} p_{t-1} \mathbf{R}_s^{(N_s - i)}, \tag{3.5}$$

kjer sta seznama $\mathbf{R}_v$ in $\mathbf{R}_s$ urejena naraščajoče, p_{t-1} pa je ploščina okvirja iz prejšnje slike sledenja po transformaciji $\theta(\mathbf{x})$.

Za vsako generirano predlagano regijo $\mathbf{a}_i^\phi$ vzamemo izsek $\mathbf{A}_i^\phi$ iz tenzorja značilk trenutne slike $\theta(\mathbf{I}_t)$, tako da je njegov center v točki, kjer smo namerili maksimum ujemanja s predlogo, dimenzije pa so enake kot pri predlagani regiji $\mathbf{a}_i^\phi$, ki jih izračunamo z enačbama (3.4) in (3.5). Ta postopek prikazuje Slika 3.3. Izsekom $\mathbf{A}_i^\phi$ z bikubično interpolacijo prilagodimo dimenzije, da sta širina in višina enaka kot pri predlogi. Predlagane regije medseboj primerjamo s stopnjo ujemanja s predlogo, ki jo izračunamo s skalarnim produktom



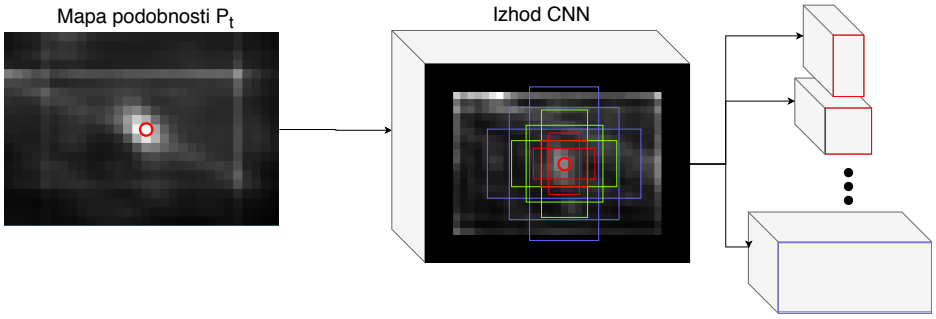
(a) (b)

Slika 3.2: Slika prikazuje dimenzije generiranih predlaganih regij, kjer pri Sliki 3.2a uporabimo seznama $\mathbf{R}_s = [0.5, 1, 2]$ in $\mathbf{R}_v = [0.5, 1, 2]$, pri Sliki 3.2b pa $\mathbf{R}_s = [0.5, 0.66, 1, 1.5, 2]$ in $\mathbf{R}_v = [0.5, 0.66, 1, 1.5, 2]$

izseka $\mathbf{A}_i^\phi$ in predloge $\theta(\mathbf{T}_t)$, po definiciji

$$u(\mathbf{a}_i^\phi) = \sum_{z=0}^{T_d-1} \sum_{y=0}^{T_d-1} \sum_{x=0}^{T_d-1} a_{z,y,x} t_{z,y,x}, \tag{3.6}$$

kjer sta $a_{z,y,x}$ in $t_{z,y,x}$ elementa iz vrstice y in stolpca x pri globini z tenzorjev $\mathbf{A}_i^\phi$ in $\theta(\mathbf{T}_t)$.



Slika 3.3: Slika prikazuje proces generiranja izsekov $\mathbf{A}_i^\phi$, ki jih uporabimo pri računanju ujemanja s predlogo.

Večja kot je vrednost $u(\mathbf{a}_i^\phi)$, bolj privlačna je prilagoditev na okvir z dimenzijami predlagane regije $\mathbf{a}_i^\phi$. Vendar se izkaže, da je mera (3.6) bolj

naklonjena k manjšim regijam in se lahko okvir hitro pomanjša in tako izgubimo objekt. Zato smo dodali faktor pristranskosti $b(\mathbf{a}_i^{\phi})$, ki je definiran kot

$$b(\mathbf{a}_i^{\phi}) = 1 + \phi/b_v, \tag{3.7}$$

kjer je b_v vnaprej določen parameter in predpostavimo, da so sklopi predlaganih regij naraščajoče urejeni po površini vsebujočih regij, kjer je ϕ indeks v vrstnem redu.

Naj bo $\mathbf{a}_{max}$ predlagana regija, od katere dobimo največje ujemanje, kar pomeni, da je najboljši predlagan okvir za objekt na trenutni sliki. Ker pa želimo okvir posodobiti le, če ujemanje odstopa od ostalih regij oziromo, ko velja neenačba

$$b(\mathbf{a}_{max})u(\mathbf{a}_{max}) \geq \mu_u + \sigma_u\psi, \tag{3.8}$$

kjer je $u(\mathbf{a}_{max})$ izračunano ujemanje z $\mathbf{a}_{max}$ po enačbi (3.6), $b(\mathbf{a}_{max})$ je faktor pristranskosti iz enačbe (3.7), μ_u in σ_u sta povprečje in standardni odklon izračunanih ujemanj vseh predlaganih regij, kjer upoštevamo faktor pristranskosti iz (3.7), ψ pa je vnaprej določen parameter. Vrednost ψ moramo previdno izbrati. Prevelik ψ povzroči, da se dimenzije okvirja ne bodo pravočasno prilagajale. Če pa je ψ nizek, pa lahko škoduje sledenju, saj ni nujno, da je napoved okvirja vedno optimalna.

3.3.2 Prilagajanje predloge

Predloga $\mathbf{T}_0$ predstavlja izgled objekta na prvi sliki videa, $\theta(\mathbf{T}_t)$ pa je transformacija predloge $\mathbf{T}_t$, ki je tenzor značilk, s katerimi iščemo objekt v t-ti iteraciji. Ker se med sledenjem objekt lahko deformira, ali pa se spremeni svetlost ali ozadje, smo implementirali posodabljanje predloge. Predloga se posodablja le na slikah, na katerih velja neenačba (3.16) iz Poglavja 3.4.

Naj $\mathbf{O}_t$ predstavlja izsek iz slike, kjer na trenutni sliki lokaliziramo objekt. Z bikubično interpolacijo prilagodimo dimenzije izseka $\mathbf{O}_t$, da po transformaciji $\theta(\mathbf{O}_t^*)$ dobimo tenzor velikosti $T_d \times T_d \times D$. Tako z enačbo

$$\theta(\mathbf{T}_{t+1}) = (1 - \alpha) \cdot \theta(\mathbf{T}_t) + \alpha \cdot \theta(\mathbf{O}_t^*) \tag{3.9}$$

izračunamo novo vrednost predloge, kjer je α vnaprej določen parameter. To predlogo uporabimo za detekcijo tarče na naslednji iteraciji sledenja.

3.3.3 Fino prilagajanje parametrov

Med sledenjem prilagajamo parametre zadnjih treh konvolucijskih slojev konvolucijske nevronske mreže, ki jo opišemo v Poglavju 3.1. S tem dosežemo, da je korelacija med templato in objektom sledenja večja in zmanjšamo korelacijo tistih delov slike, kjer ni objekta sledenja, vendar je korelacija visoka. To funkcijo izvajamo samo takrat, ko je detekcija zanesljiva in velja neenačba (3.16) iz Poglavja 3.4.

Najprej potrebujemo pozitivne in negativne vzorce, ki jih bomo uporabili za učno množico. Te vzorci so izseki iz iskalne slike $\mathbf{I}_t$. Vzamemo n_{samp} vzorcev, ki imajo dimenzije enake kot okvir lokalizacije na trenutni iteraciji $\mathbf{o}_t$. Središča teh vzorcev pa dobimo tako, da vzamemo prvih n_{samp} maksimalnih točk iz mape podobnosti $\mathbf{P}_t$ in jih preslikamo na koordinate slike $\mathbf{I}_t$, kot to prikazuje Slika 3.4. Te vzorce nato klasificiramo kot pozitivne ali kot negativne glede na njihov presek čez unijo (IoU) z okvirjem $\mathbf{o}_t$. Če je oznaka regije i-tega vzorca $\mathbf{r}_i$ in če velja neenačba

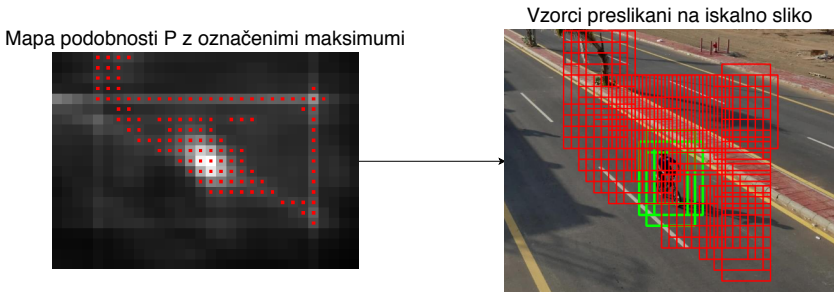
$$IoU(\mathbf{o}_t, \mathbf{r}_i) > \omega^+, \tag{3.10}$$

potem smatramo r_i kot pozitiven vzorec, če pa velja

$$IoU(\mathbf{o}_t, \mathbf{r}_i) < \omega^-, \tag{3.11}$$

pa kot negativen vzorec. Pozitivnim vzorcem dodamo še čez os y prezrcaljene izseke pozitivnih vzorcev, da dobimo večje število pozitivnih vzorcev ter zmanjšamo verjetnost prekomernega prileganja. Ker ni nujno, da se vsak od n_{samp} vzorcev klasificira v enega od klasifikacijskih razredov, vzamemo vseh skupaj n_{in} vzorcev, ki predstavljajo učno množico.

Za vsako regijo $\mathbf{r}_i$ izrežemo vzorec $\mathbf{R}_i$ iz slike $\mathbf{I}_t$ in jim z bikubično interpolacijo prilagodimo dimenzije na $8T_d \times 8T_d$. Na vhod konvolucijske nevronske mreže podamo gručo po n_{batch} vzorcev. To storimo n_{epoch} krat, s tem, da



Slika 3.4: Slika prikazuje način izbiranja vzorcev. Na levi je prikazana mapa podobnosti P_t, kjer so z rdečo označene točke prvih n_{samp} maksimumov. Na desni pa so prikazani vzorci, kjer so z zeleno označeni pozitivni, z rdečo pa negativni vzorci.

pred vsako iteracijo vzorce naključno premešamo. Pozitivne in negativne razvrstimo tako, da imamo približno trikrat več negativnih.

Za prilagajanje parametrov uporabimo stohastični gradientni spust s hitrostjo učenja λ_r. Za resnične oznake (*angl. truth labels*) pri pozitivnih vzorcih vzamemo njihov presek čez unijo z $\mathbf{o}_t$, pri negativnih pa je oznaka 0. Uporabimo kriterijsko funkcijo Softmax s križno entropijo, kjer so vhodne vrednosti l_i izračunani po definiciji

$$l_i = \frac{h_i}{\max(h)}, \tag{3.12}$$

kjer je h_i korelacija med templato $\theta(\mathbf{T}_t)$ in tenzorjem vzorca $\theta(\mathbf{R}_i)$, $\max(h)$ pa je maksimum teh ujemanj. S tem ko je resnična oznaka za negativne vzorce enaka 0, učimo parametre mreže na način, da bo razmerje med h_i in $\max(h)$ konvergirala proti 0 za vse negativne vzorce. Za pozitivne pa je resnična vrednost enaka $IoU(\mathbf{o}_t, \mathbf{r}_i)$, zato da tistim pozitivnim vzorcem z večjim prekrivanjem tudi bolj povečamo razmerje h_i proti $\max(h)$.

Pri implementaciji moramo previdno izbrati hiper parametre. Previsoka hitrost učenja lahko preveč prilagodi parametre na trenuten izgled objekta, ali pa se preveč prileže ozadju (*angl. background clutter*).

3.4 Merjenje gotovosti sledenja

Sledilniku dodamo funkcijo merjenja gotovosti sledenja. Če je na neki sliki gotovost detekcije slaba, sledilnik zazna odpoved sledenja. Tako postane dolgoročni sledilnik in nam daje možnost, da ob odpovedi povečamo verjetnost ponovne detekcije. Metoda zaznavanja odpovedi izhaja iz članka [20].

Moč detekcije izračunamo po definiciji predstavljeni v članku [4]. Naj bo $\max(\mathbf{P}_t)$ vrednost maksimuma na mapi podobnosti $\mathbf{P}_t$ in $PSR(\mathbf{P}_t)$ naj bo razmerje maksimuma proti regiji okoli maksimuma (*angl. peak-to-sidelobe ratio, PSR*). Za izračun $PSR(\mathbf{P}_t)$ vzamemo maksimum $\max(\mathbf{P}_t)$ in 11×11 veliko regijo $\mathbf{R}_{sl}$ okoli točke maksimuma. Če sta μ_{sl} in σ_{sl} povprečje in standardni odklon regije $\mathbf{R}_{sl}$, potem po enačbi

$$PSR(\mathbf{P}_t) = \frac{\max(\mathbf{P}_t) - \mu_{sl}}{\sigma_{sl}} \tag{3.13}$$

izračunamo vrednost razmerja PSR. Nato po definiciji

$$q = \max(\mathbf{P}_t) \cdot PSR(\mathbf{P}_t) \tag{3.14}$$

izračunamo moč detekcije v trenutni iteraciji. Sedaj pa primerjamo q s povprečno močjo detekcije zadnjih n_q iteracij, oznaka katere naj bo μ_q. Če je razmerje med tema dvema vrednostima večja od vnaprej določenega pragu ρ^-, definiran kot

$$\frac{\mu_q}{q} \geq \rho^-, \tag{3.15}$$

potem sledenje smatramo kot negotovo in zaznamo odpoved sledenja. Če zaznamo odpoved, potem povečamo regijo iskanja za faktor 2, ne prilagajamo okvirja in v naslednji iteraciji ne apliciramo gausovega filtra na mapo podobnosti.

Če pa, v nasprotnem primeru, velja, da je razmerje med q in μ_q manjše od ρ^+, kot prikazuje neenačba

$$\frac{\mu_q}{q} \leq \rho^+, \tag{3.16}$$

takrat pa sledilnik predpostavi, da je trenutna detekcija objekta zelo dobra. Ko je detekcija zelo dobra uporabimo eno ali obe metodi za prilagajanje vizualnega modela na izgled objekta trenutne iteracije. Te dve metodi opišemo v Poglavju 3.3.2 in Poglavju 3.3.3.

Pri izbranih vrednosti za upragovanje velja $\rho^+ \leq \rho^-$. Vrednosti sta lahko enaki, takrat velja, da bomo zaznali ali odpoved, ali pa dobro detekcijo.

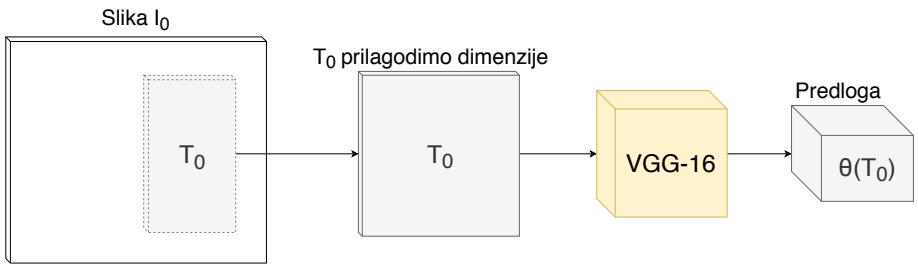
3.5 Dolgoročni sledilnik SiamRP-LT

Predlagano implementacijo sledilnika smo imenovali SiamRP-LT. Ime izhaja iz dejstva, da sledilnik uporabi siamsko konvolucijsko nevronsko mrežo, predlagane regije (*angl. region proposals*) in je dolgoročen (*angl. long-term*).

Delovanje sledilnika delimo na dva dela: (i) inicializacijo in (ii) sledenje. Pri inicializaciji sledilniku podamo iskalno sliko $\mathbf{I}_0$ in regijo $\mathbf{o}_0 = \{\mathbf{o}_0^{(i)}\}_{i=1:4}$, ki predstavlja koordinate štirih točk izseka $\mathbf{O}_0$ iz $\mathbf{I}_0$, ki vsebuje objekt sledenja. Da sledilnik dobi boljši kontekst ozadja, povečamo dimenzije $\mathbf{O}_0$ za p, ki je definiran z enačbo

$$p = \frac{\mathbf{O}_0^{(w)} + \mathbf{O}_0^{(h)}}{p_d}, \tag{3.17}$$

kjer je $\mathbf{O}_0^{(w)}$ širina, $\mathbf{O}_0^{(h)}$ višina izseka $\mathbf{O}_0$ in p_d je določen parameter. Tako dobimo izsek $\mathbf{T}_0$, iz katerega po transformaciji $\theta(\mathbf{T}_0)$ dobimo predlogo. Vendar pred transformacijo, moramo zaradi fiksne velikosti predloge $\theta(\mathbf{T}_0)$ prilagoditi dimenzije izseka $\mathbf{T}_0$ na $8T_d \times 8T_d$. Proces inicializacije je prikazan na Sliki 3.5.



Slika 3.5: Prikazan proces inicializacije.

Pri sledenju pa v t-ti iteraciji sledilnik prejme sliko $\mathbf{I}_t$. Iz optimizacijskih razlogov ne preiščemo celotne slike, temveč le njen izsek. Ker predvidevamo, da bo pozicija objekta nekje blizu kot v iteraciji $t-1$, je center izseka v točki, kjer je bil lokaliziran objekt na prejšnji sliki. Izsek je kvadrat, katerega stranice imajo dolžino definirano po enačbi

$$d = \tau \cdot \max(\mathbf{o}_{t-1}^{(w)}, \mathbf{o}_{t-1}^{(h)}), \tag{3.18}$$

kjer je τ faktor skaliranja, ki je vnaprej določen parameter. Ta parameter se podvoji, če je sledilnik v iteraciji $t-1$ zaznal odpoved sledenja po enačbi (3.15) iz Poglavja 3.4. Tako dobimo izsek $\mathbf{I}'_t$, na katerem lokaliziramo objekt, da dobimo okvir $\mathbf{o}_t$ in nato izračunamo zaupanje detekcije q. Če zaznamo odpoved sledenja, potem dimenzij okvirja ne prilagajamo, temveč samo njegovo pozicijo. Če pa je zaupanje dobro, pa prilagajamo predlogo $\theta(\mathbf{T}_t)$, kot opišemo v Poglavju 3.3.2 in oz. ali parametre zadnjih treh nivojev konvolucijske nevronske mreže, ki pa smo postopek opisali v Poglavju 3.3.3. Ker je $\mathbf{I}'_t$ izsek iz slike, moramo poziciji okvirja $\mathbf{o}_t$, ki jo dobimo na način opisan v Poglavju 3.2, prišteti še koordinate zgornjega levega kota izseka, da dobimo končno točko nove pozicije objekta sledenja na sliki $\mathbf{I}_t$. Končnemu okvirju pa zaradi povečanja dimenzij za p moramo že zmanjšati dimenzije za p^-, ki je definiran kot

$$p^- = \frac{\mathbf{o}_t^{(w)} + \mathbf{o}_t^{(h)}}{p_d + 2}. \tag{3.19}$$

Iteracijo sledenja smo opisali v pseudokodi v Algoritmu 1.

Algoritem 1: SiamRP-LT

Potrebuje: t-to sliko iz sekvence $\mathbf{I}_t$, templato $\theta(\mathbf{T}_t)$, rezultat lokalizacije $\mathbf{o}_{t-1}$ prejšnje iteracije, povprečje moči detekcije μ_q zadnjih n_q iteracij

Rezultat : lokalizacija objekta $\mathbf{o}_t$

Začni

[1] Povečaj dimenzije $\mathbf{o}_{t-1}$ za vrednost dobljeno po (3.17).

[2] Izreži izsek $\mathbf{I}'_t$, s sredino v točki centra $\mathbf{o}_{t-1}$ in dimenzijama stranic izračunanih po (3.18).

[3] Izseku $\mathbf{I}'_t$ skaliraj širino in višino po (3.1) in pošlji skozi CNN in obdaj z ničelno oblogo v širini $\lfloor \frac{T_d}{2} \rfloor$, dobimo $\theta(\mathbf{I}'_t)$.

[4] **while** *ponavljaj do konca pogoja* **do**

[5] Izračunaj mapo podobnosti $\mathbf{P}_t$ tako, da na vsaki točki (i,j) tenzorja $\theta(\mathbf{I}'_t)$ izreži izsek $\theta(\mathbf{I}'_t)^{(i,j)}$ in po (3.2) izračunaj ujemanje med izsekom in predlogo ter vrednost shrani v $p_{i,j} \in \mathbf{P}_t$

[6] **end**

[7] **if** *na prejšnji iteraciji nismo zaznali odpovedi* **then**

[8] Na $\mathbf{P}_t$ apliciramo Gaussov filter s sredino v točki centra $\mathbf{o}_{t-1}$ in standardnim odklonom σ.

[9] **end**

[10] Generiraj predlagane regije $\mathbf{a}_i^\phi$ z uporabo $\mathbf{o}_{t-1}$ in seznamoma $\mathbf{R}_v$ ter $\mathbf{R}_s$.

[11] **while** *za vsako predlagano regijo* $\mathbf{a}_i^\phi$ **do**

[12] Prilagodi dimenzije predlagane regije na $T_d \times T_d$.

[13] Izračunaj ujemanje med predlagano regijo in predlogo po (3.6) in vrednost shrani v seznam *score*.

[14] **end**

[15] Če velja neenačba (3.8) izberi za okvir $\mathbf{a}_{max}$, sicer obdrži okvir $\mathbf{o}_{t-1}$.

[16] Izračunaj moč detekcije q po enačbi (3.14).
[17] **if** *velja neenačba (3.16)* **then**
[18] Zaupanje sledenja je dobro
[19] Posodobi predlogo po enačbi (3.9) in/ali prilagodi parametre CNN
[20] **else if** *velja neenačba (3.15)* **then**
[21] Zaznana odpoved sledenja
[22] Povčaj τ iz enačbe (3.18) za faktor 2
[23] Zanemari prilagoditev okvirja
[24] Postavi $\mathbf{o}_t$ na točko, ki jo dobimo po (3.3) in dodeli izbrani okvir pomanjšan za vrednost dobljeno z (3.19)

Poglavje 4

Eksperimentalna evalvacija

V tem poglavju predstavimo metode analize uspešnosti dolgoročnega sledenja sledilnika SiamRP-LT. V Poglavju 4.1 opišemo strojno opremo in pa nastavitev parametrov, s katerimi izvajamo eksperimente. Nato v Poglavju 4.2, kjer opišemo postopek testiranja uspešnosti z različnimi modifikacijami sledilnika SiamRP-LT. Na koncu pa v Poglavju 4.3 opišemo rezultate primerjanja sledilnika SiamRP in metod objavljenih na tekmovanju VOT-LT2018 [16], ki je namenjeno ravno za primerjavo dolgoročnih sledilnikov.

4.1 Implementacijske podrobnosti

Sledilnik smo implementirali v programskem jeziku Python z uporabo knjižnic Tensorflow [1] in Keras [7]. Računalnik na katerem smo pognali evalvacijo vsebuje procesor Intel i7-4790K 4GHz in 16GB notranjega pomnilnika. Prilagajanje parametrov mreže in prehode skozi mrežo pa izvajamo z grafično kartico NVIDIA GeForce GTX 970 s 3,5GB pomnilnika.

Transformacija $\theta(\mathbf{x})$ predstavlja izhod zadnjega konvolucijskega nivoja mreže, ki jo opišemo v Poglavju 3.1. Pri detektiranju objekta uporabimo predlogo velikosti $T_d = 8$. Za izračun gotovosti detekcije iz Poglavja 3.4 uporabimo enake parametre kot v članku [20] s pragom za odpoved sledenja $\rho^- = 2{,}9$ ter pragom za dobro detekcijo $\rho^+ = 0{,}9$. Ko posodabljamo predlogo

na način opisan v Poglavju 3.3.2, pri enačbi (3.9) uporabimo parameter $\alpha = 0{,}02$. Velikost izseka slike, po katerem iščemo objekt, izračunamo po enačbi (3.18), kjer za parameter uporabimo $\tau = 2{,}5$. Za generiranje predlaganih regij, kot to opišemo v Poglavju 3.3.1, uporabimo seznam razmerij velikosti sklopov $\mathbf{R}_v = [0{,}5; 0{,}66; 1; 1{,}5; 2]$ in seznam razmerij dolžin stranic po sklopih $\mathbf{R}_s = [0{,}5; 0{,}66; 1; 1{,}5; 2]$. S tem generiramo 25 predlaganih regij. Pri enačbi za izračun faktorja pristranskosti (3.7), uporabimo $b_v = 10$ ter pri neenačbi (3.8), ki določi ali je izbrana predlagana regija dovolj dobra za posodobitev okvirja, uporabimo $\psi = 1{,}5$.

Če velja neenačba (3.16) potem prilagajamo parametre zadnjih treh konvolucijskih nivojev na način, kot ga opisujemo v Poglavju 3.3.3. Najprej vzamemo $n_{max} = 256$ vzorcev in jih glede na (3.10) označimo kot pozitivne, kjer je $\omega^+ = 0{,}7$ in glede na (3.11) kot negativne, kjer je $\omega^- = 0{,}3$. Pri učenju pa uporabimo naslednje parametre:

- izberemo $n_{in} = 128$ vzorcev, tako da je približno trikrat več negativnih,
- skozi mrežo jih pošiljamo po $n_{batch} = 64$,
- število iteracij je $n_{epoch} = 5$,
- hitrost učenja pa nastavimo na $\lambda_r = 0{,}00001$.

4.2 Testiranje modifikacij SiamRP-LT

Analizirali smo uspešnost dolgoročnega sledenja različnih modifikacij sledilnika SiamRP-LT. Postopek analize opišemo v Poglavju 4.2.1. Mere s katerimi smo primerjali uspešnost opišemo v Poglavju 4.2.2. V Poglavju 4.2.3 analiziramo vpliv na uspešnost sledenja različnih nastavitev generiranja predlaganih regij. Analizo predlaganih izboljšav za prilagajanje vizualnega modela na barvne spremembe pa opišemo v Poglavju 4.2.4.

4.2.1 Protokol analize uspešnosti na zbirki LTB35

Vse eksperimente smo izvedli na sekvencah zbirke LTB35 [21]. Zbirka vsebuje 35 dolgih sekvenc, kjer objekt sledenja izginja iz pogleda kamere. Povprečno 12-krat na sekvenco objekt izgine za dalj časa. Tarče so anotirane z okvirji, katerih stranice so poravnane s koordinatnimi osmi. Scenariji, ki so vsebovani v sekvencah te zbirke: (i) Polna okluzija, (ii) Tarča izven pogleda, (iii) Delna okluzija, (iv) Premikanje kamere, (v) Hitro premikanje, (vi) Spreminjanje velikosti, (vii) Spreminjanje razmerja stranic, (viii) Sprememba pogleda, (ix) Podobni objekti.

Eksperimenti potekajo tako, da na prvi sliki inicilaiziramo sledilnik, nato pa pustimo, da sledilnik teče do konca sekvence nenadzorovano. Nenadzorovano v smislu, da ob odpovedi sledenja sledilnik ne ponastavimo in ponovno inicializiramo, kot pri tekmovanju VOT-ST2018 [16] za kratkoročne sledilnike. Naloga dolgoročnega sledilnika je, da ob odpovedi sledenja tarčo ponovno detektira, ko se ta prikaže.

Za primerjavo rezultatov eksperimentov smo uporabili orodje *vot-toolkit*, ki nam ga ponuja komisija tekmovanja VOT [16]. S tem orodjem izrišemo grafe iz meritev, ki jih opišemo v Poglavju 4.2.2. Med sledenjem, za vsako sliko poročamo koordinate lokalizacije in moč detekcije. Moč detekcije q, izračunano po enačbi (3.14), poročamo kot $1/q$, tako da večji q pomeni slabšo moč detekcije. Mere s katerimi smo primerjali različne nastavitve, so enake kot jih predlagajo v članku [21] in smo jih opisali v Poglavju 4.2.2.

4.2.2 Performančne mere

Mere, ki jih uporabimo za primerjanje so namenjene merjenju dolgoročnosti sledilnika in jih predstavijo v članku [21]. Izmerijo, kako dobro sledilnik zaznava okluzijo ali odsotnost objekta in kako hitro ga detektira ob ponovni pojavitvi. Predstavijo tri performančne mere: (i) priklic (*angl. recall*), (ii) natančnost (*angl. precision*) in (iii) F-mera. Mere izračunamo na naslednji način, ki je predstavljen v članku [21].

Naj bo G_t resničen okvir objekta (*angl. ground truth*), $A_t(\tau_q)$ okvir, ki ga predlaga sledilnik, q_t je zaupanje detekcije na t-ti iteraciji sledenja in pa τ_q, ki predstavlja prag detekcije. Če objekta ni na sliki potem $G_t = \emptyset$, če pa sledilniku zaupanje sledenja pade in velja $q_t < \tau_q$, potem je $A_t(\tau_q) = \emptyset$. Pri izračunu priklica in natančnosti uporabimo mero $IoU(G_t, A_t(\tau_q))$, ki predstavlja presek čez unijo (*angl. intersection over union*) in je definirana z enačbo

$$IoU(A, B) = \frac{A \cap B}{A \cup B}. \tag{4.1}$$

Naj N_g predstavlja število slik v sekvenci, kjer je objekt prisoten, se pravi $G_t \neq \emptyset$, in naj N_p predstavlja število slik v sekvenci, kjer sledilnik ne zazna odpovedi, pomeni da $A_t(\tau_q) \neq \emptyset$.

Natančnost predstavlja povprečen presek čez unijo na slikah, kjer sledilnik ni zaznal odpovedi oziroma na slikah, kjer velja $A_t(\tau_q) \neq \emptyset$. Izračunamo jo z enačbo

$$Pr(\tau_q) = \frac{1}{N_p} \sum_{t \in \{t:\ A_t(q_t) \neq \emptyset\}} IoU(G_t, A_t(\tau_q)). \tag{4.2}$$

Priklic pa predstavlja povprečen presek čez unijo na slikah, kjer je objekt prisoten in ga izračunamo z enačbo

$$Re(\tau_q) = \frac{1}{N_g} \sum_{t \in \{t:\ G_t \neq \emptyset\}} IoU(G_t, A_t(\tau_q)). \tag{4.3}$$

Rezultata enačbe (4.2) in (4.3) združimo, da izračunamo F-mero, po definiciji

$$F(\tau_q) = \frac{2Pr(\tau_q)Re(\tau_q)}{(Pr(\tau_q) + Re(\tau_q))}. \tag{4.4}$$

F-mera je primarna meritev za ocenjevanje uspešnosti dolgoročnega sledilnika pri naših eksperimentih.

Pri vsakem eksperimentu izmerimo še hitrost delovanja. Hitrost se izmeri kot povprečno število obdelanih slik na sekundo.

4.2.3 Analiza nastavitev predlaganja regij

V tem poglavju analiziramo vpliv števila, velikosti ter razmerja stranic predlaganih regij na dolgoročnost sledenja. Eksperimente smo izvedli po proto-

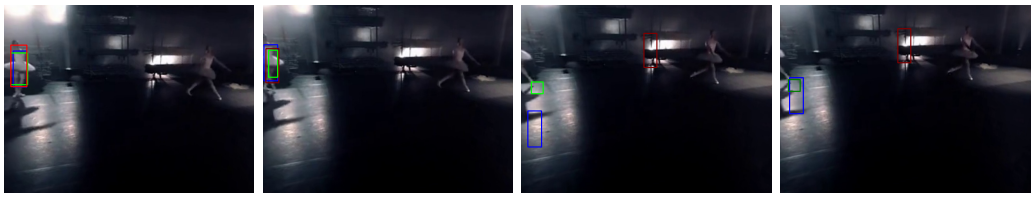
kolu, ki ga opišemo v Poglavju 4.2.1. Za ta eksperiment smo sledilnik pognali brez posodabljanja predloge in parametrov mreže, ker nas pri tem eksperimentu zanima zgolj delovanje posodabljanja okvirja. Izmerili smo rezultate treh nastavitev generiranja predlaganih regij. Nastavitve smo predstavili v Tabeli 4.1. Ostali parametri so enaki kot smo jih opisali v Poglavju 4.1.

Sledilnik	**Razmerja ploščin**	**Razmerja dolžin stranic**
SiamRP-LT$_1$	0,5; 1; 2	0,5; 1; 2
SiamRP-LT$_2$	0,33; 0,5; 1; 2; 3	0,5; 0,66; 1; 1,5; 2
SiamRP-LT$_3$	0,5; 0,66; 1; 1,5; 2	0,5; 0,66; 1; 1,5; 2

Tabela 4.1: Nastavitve seznamov razmerij dolžin stranic in ploščin za generiranje predlaganih regij. SiamRP-LT$_1$ generira skupno 9 predlaganih regij, SiamRP-LT$_2$ in SiamRP-LT$_3$ pa 25.

Tabela 4.2 nam pokaže izmerjeno F-mero in hitrost testiranja sledilnikov SiamRP-LT$_1$, SiamRP-LT$_2$ in SiamRP-LT$_3$. Vidimo, da je SiamRP-LT$_3$ dosegel največjo F-mero, za približno 21% več kot SiamRP-LT$_1$ in 26% več kot SiamRP-LT$_2$. Zanimivo je, da SiamRP-LT$_2$ deseže najslabše rezultate, slabše kot SiamRP-LT$_1$, ki generira manj regij. Razlog za to je, da se ob izginjanju objekta okvir izrazito pomanjša preden sledilnik zazna odpoved, kot to prikazuje Slika 4.1. Ko objekt izgine ga sledilnik težko ponovno detektira, ker je okvir precej manjši od objekta in pa ker regija iskanja postane precej manjša kot celotna slika iskanja, tako da regija iskanja niti ne zajema objekta sledenja.

Hitrost delovanja je prikazana na drugem stolpcu Tabele 4.2 in predstavlja število obdelanih slik na sekundo. SiamRP-LT$_1$ je najhitrejši, ker generira 9 predlaganih regij in tako izvede 9 primerjav s predlogo, v primerjavi z ostalima dvema, ki izvedeta 25 primerjav. SiamRP-LT$_3$ je približno 10% počasnejši, vendar glede na boljše rezultate pri F-meri, priklicu in natančnosti, je to zanemarljivo. SiamRP-LT$_2$ pa je najpočasnejši. Razloga tega je, ker sledenje odpove in tako pri velikem številu slik zazna odpoved, s tem pa se podvoji velikost regije iskanja in tako se poveča časovna zahtevnost.



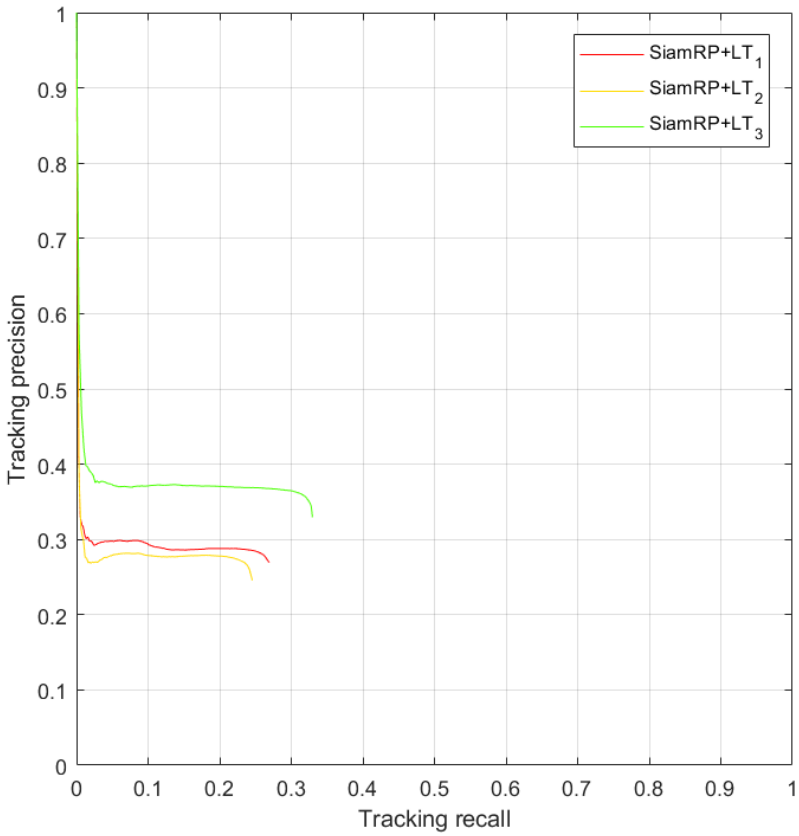
Slika 4.1: Prikazuje zaporedje slik, kjer objekt sledenja izgine iz pogleda kamere. SiamRP-LT$_1$ predstavlja moder, SiamRP-LT$_2$ zelen in SiamRP-LT$_3$ rdeč okvir. Zatemnjena barva okvirja pa predstavlja negotovo detekcijo.

Eksperiment	**F-mera**	**Hitrost [slike/s]**
SiamRP-LT$_1$	**0.27**	**9.78**
SiamRP-LT$_2$	0.25	8.08
SiamRP-LT$_3$	**0.34**	**8.78**

Tabela 4.2: Izmerjena F-mera in hitrost na zbirki LTB35 [21] različnih nastavitev generiranja predlaganih regij. Rdeča predstavlja najboljši rezultat, modra drugo, zelena pa tretje mesto.

Slika 4.2 prikazuje povprečno krivuljo natančnost-priklic vseh sekvenc iz LTB35 [21]. Bližje kot je krivulja desnemu zgornjemu kotu, boljše je dolgoročno sledenje. Iz grafa se vidi, da je nastavitev sledilnika SiamRP-LT$_3$ boljša od ostalih dveh, ker doseže večjo preciznost in priklic od ostalih dveh predlaganih sledilnikov.

S temi eksperimenti smo ugotovili, da število predlaganih regij lahko precej vpliva na uspešnost sledenja. Vendar bolj ključna za uspešnost je nastavaitev velikosti in dolžin stranic. Dobra nastavitev je, da so razmaki med ploščinam in dolžinam strani predlaganih regij manjši, da se med sledenjem okvir lahko posodobi večkrat, vendar je sprememba velikosti in razmerja stranic te posodobitve manjša.



Slika 4.2: Slika prikazuje povprečno krivuljo preciznost-priklic eksperimentov analize.

4.2.4 Analiza prilagajanja vizualnega modela

V tem poglavju analiziramo predlagani metodi za prilagajanje vizualnega modela med sledenjem, ki jih opišemo v Poglavju 3.3.2 in Poglavju 3.3.3. Eksperimente smo izvedli po protokolu, ki ga opišemo v Poglavju 4.2.1. Parametre smo uporabili enake, kot jih opišemo v Poglavju 4.1. Za generiranje predlaganih regij pa smo uporabili nastavitev iz Poglavja 4.2.3, ki je dosegla najvišjo F-mero.

Izvedli smo štiri eksperimente s štirimi sledilniki, katerih imena in nastavitve so sledeče:

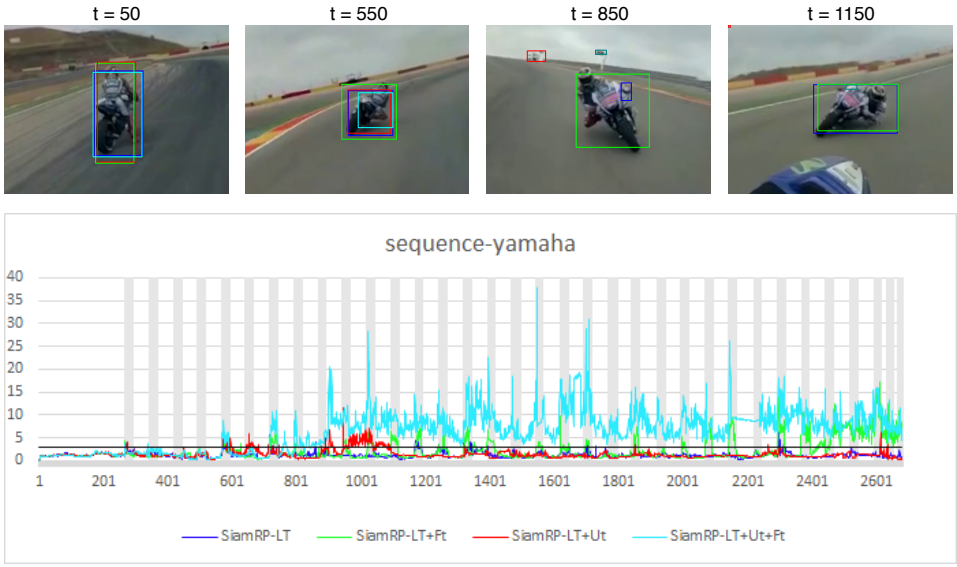
- **SiamRP-LT**: sledilnik brez predlaganih metod, za primerjavo
- **SiamRP-LT+Ut**: sledilnik s posodabljanjem predloge, kot opišemo v Poglavju 3.3.2
- **SiamRP-LT+Ft**: sledilnik s finim prilagajanjem parametrov mreže, kot opišemo v Poglavju 3.3.3
- **SiamRP-LT+Ut+Ft**: sledilnik s posodabljanjem predloge in finim prilagajanjem parametrov mreže

Da smo se izognili prekomernemu prilagajanju parametrov mreže, smo funkcijo finega prilagajanja omejili, da se po izvajanju lahko ponovno izvede šele po 40 zaporednih slikah. Pravtako, se izkaže, da se pri uporabi finega prilagajanja mreže poveča naklonjenost izbire manjših predlaganih regij, zato smo sledilnikoma SiamRP-LT+Ft in SiamRP-LT+Ut+Ft podvojili faktor naklonjenosti (3.7) tako, da smo nastavili $b_v = 5$.

Izmerjeni F-mera in hitrost vsakega eksperimenta povzame Tabela 4.3. Sledilnik brez prilagajanja SiamRP-LT ima povprečno F-mero 0.34. Če temu sledilniku dodamo fino prilagajanje parametrov, kot to predstavlja sledilnik SiamRP-LT+Ft, potem izmerimo F-mero 0.38, kar je 11% boljše kot pri SiamRP-LT. Vendar SiamRP-LT+Ft deluje s hitrostjo 4.7 slik na sekundo, kar je za 47% počasneje kot pa pri SiamRP-LT, ki obdela 8.78 slik na sekundo.

Če pa med sledenjem posodabljamo predlogo, pa pri SiamRP-LT+Ut izmerimo F-mero 0.22, pri SiamRP-LT+Ut+Ft pa 0.20. V primeru, da sledilniku SiamRP-LT dodamo posodabljanje predloge, se njegova izmerjena F-mera poslabša za približno 32%, če pa enako funkcionalnost dodamo sledilniku SiamRP-LT+Ft, pa se F-mera poslabša za 47%.

Razlog za slabšo uspešnost dolgoročnega sledenja sledilnikov, ki posodabljajo predlogo je, da se z vsako posodobitvijo prvotna predstavitev tarče zamegli. S tem pa postane detekcija sledenega objekta in zaznavanje odpovedi sledenja nepredvidljivo. Elementi tenzorja predloge se lahko spremenijo v taki meri, da sledilnik ne zazna odpovedi, temveč je kljub napačni lokalizaciji siguren. Včasih celo velja neenačba (3.16) in sledilnik posodablja



Slika 4.3: Prikazuje izmerjeno gotovost na sekvenci *yamaha* za vsakega od testiranih sledilnikov. Graf je sivo obarvan tam, kjer objekt sledenja izgine iz pogleda. Večja kot je vrednost bolj negotova je detekcija, če pa preseže črno vodoravnico pa sledilnik zazna odpoved. Nad grafom pa so rezultati lokalizacije v iteracijah 50, 550, 850, 1150.

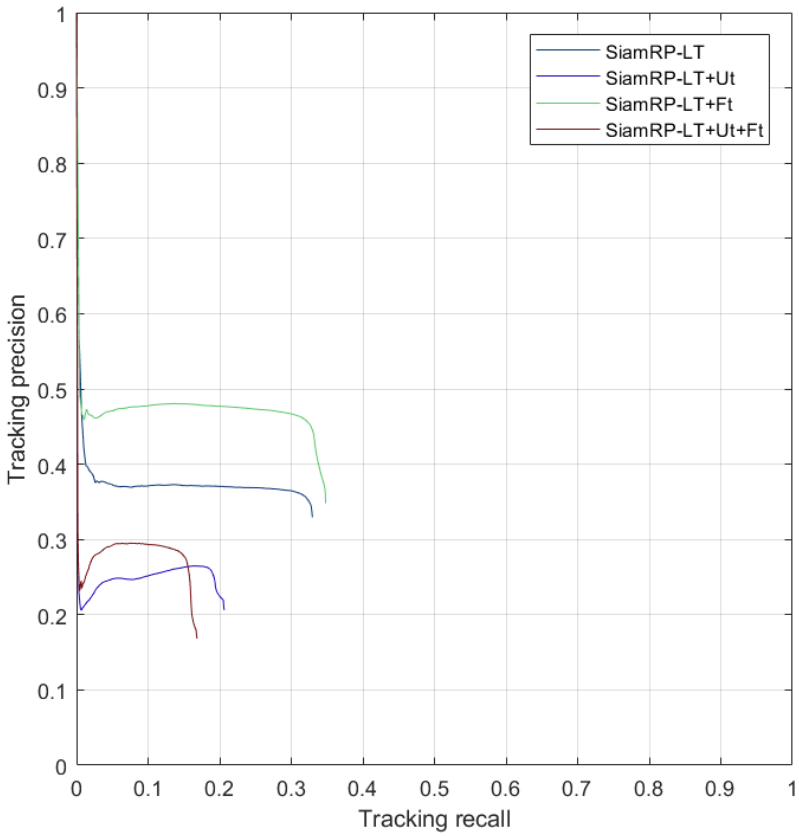
predlogo, ko lokalizacija ni uspešna. Takrat sledenje hitro odpove, ker sledilnik začne izbirati manjše predlagane regije, ker predstavitev izgleda objekta

na predlogi počasi izginja in izbiranje predlaganega okvirja postane nepredvidljivo. Pri sledilniku SiamRP-LT+Ut+Ft je to še bolj problematično, saj parametre mreže takrat ne prilagajamo na izgled tarče. To je prikazano na Sliki 4.3, kjer se okvir pri SiamRP-LT+Ut in SiamRP-LT+Ut+Ft ob izgubljeni tarči izrazito pomanjša. Ko je enkrat okvir precej manjši od tarče, se skoraj ne mora več popraviti, da bi zajemal celotno tarčo, ker takrat ponavadi zazna odpoved, sledilnik pa ob zaznani odpovedi okvirja ne posodablja. Prav tako pa ni zagotovila, da bo takrat korelacija večjih predlaganih regijah s predlogo večja kot pri ostalih predlaganih regijah.

Sledilnik	**F-mera**	**Hitrost [slike/s]**
SiamRP-LT	**0.34**	**8.78**
SiamRP-LT+Ut	0.22	**8.43**
SiamRP-LT+Ft	**0.38**	4.70
SiamRP-LT+Ut+Ft	0.20	4.70

Tabela 4.3: Izmerjena F-mera in hitrost na zbirki LTB35 [21] eksperimentov. Rdeča pisava predstavlja najboljši rezultat, modra drugo, zelena pa tretje mesto.

Slika 4.4 prikazuje krivuljo preciznost-priklic za vsakega od eksperimentov. Bližje kot je krivulja zgornjemu desnemu kotu, boljše je sledenje. Na grafu se vidi vpliv različne modifikacije. Če dodamo sledilniku SiamRP-LT fino prilagajanje parametrov mreže, potem se preciznost kot priklic povečata, vendar v zameno za hitrost delovanja. Saj je učenje konvolucijske nevronske mreže časovno in prostorsko precej zahtevno.



Slika 4.4: Slika prikazuje povprečno krivuljo preciznost-priklic za sledilnike iz analize.

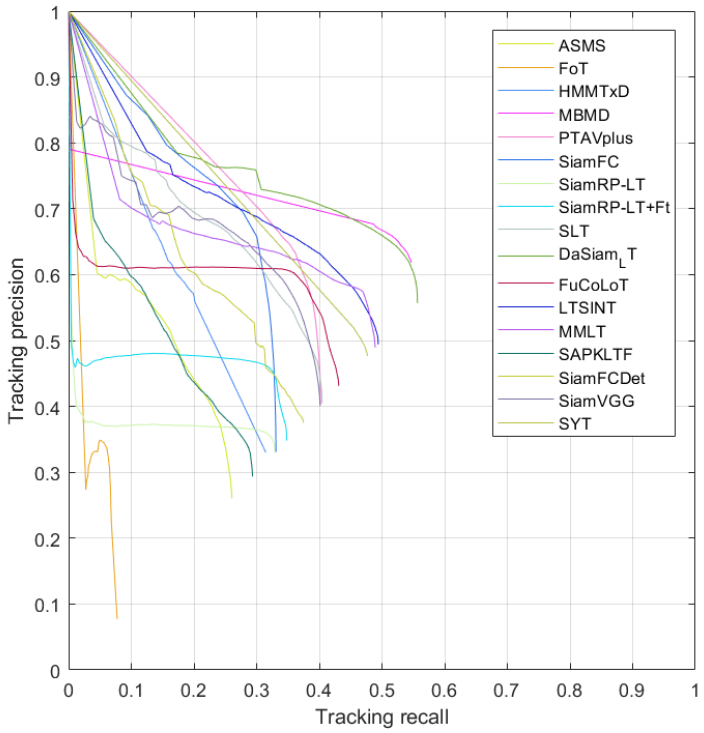
4.3 Primerjava s sorodnimi deli

Sledilnika SiamRP-LT in SiamRP-LT+Ft smo primerjali s 15 metodami, ki so bile objavljene na tekmovanju VOT-LT2018 [16] za dolgoročne sledilnike. Meritve uspešnosti so F-mera, natančnost in priklic, ki jih izračunamo na način, kot ga opišemo v Poglavju 4.2.2. Eksperimenti potekajo po protokolu, ki ga opišemo v Poglavju 4.2.1. Tekmovanje VOT-LT2018 [16] je sestavljeno iz dveh eksperimentov: (i) uspešnost dolgoročnega sledenja, (ii) uspešnost ponovne detekcije. Mi smo izvedli le eksperiment za merjenje uspešnosti dolgoročnega sledenja in rezultate primerjali z ostalimi. Rezultati metod tekmovanja so na voljo na uradni spletni strani VOT tekmovanja in so bolj podrobno predstavljeni v članku [15].

V Tabeli 4.4 so prikazane F-mera in hitrost sledilnikov SiamRP-LT in SiamRP-LT+Ft in ostalih metod iz tekmovanja VOT-LT2018 [16]. Slika 4.5 pa prikazuje krivuljo preciznost-priklic za vsak sledilnik iz Tabele 4.4. Glede na F-mero se je SiamRP-LT uvrstil na dvanajsto mesto, SiamRP-LT+Ft pa si z sledilnikom SiamFCDet deli enajsto mesto. Od zmagovalnega sledilnika MBMD, je F-mera SiamRP-LT manjša za 41%, pri SiamRP-LT+Ft pa za 35%. Glede na hitrost delovanja, pa je SiamRP-LT deveti najhitrejši, SiamRP-LT+Ft pa trinajsti najhitrejši, vendar moramo upoštevati, da uporabljena strojna oprema ni enaka kot pri metodah tekmovanja.

	Sledilnik	**F-mera**	**Hitrost [slike/s]**
1.	MBMD	**0.58**	2.71
2.	DaSiam_LT	**0.58**	20.81
3.	MMLT	0.52	6.15
4.	LTSINT	0.51	15.97
5.	SYT	0.48	17.77
6.	PTAVplus	0.46	2.01
7.	FuCoLoT	0.46	6.78
8.	SiamVGG	0.44	33.17
9.	SLT	0.43	14.89
10.	SiamFC	0.41	38.75
11.	SiamFCDet	0.38	5.75
12.	SiamRP-LT+Ft	0.38	4.70
13.	SiamRP-LT	0.34	8.78
14.	HMMTxD	0.32	3.53
15.	SAPKLTF	0.31	17.92
16.	ASMS	0.29	**98.88**
17.	FoT	0.11	**117.56**

Tabela 4.4: Prikazuje F-mero in hitrost dveh sledilnikov SiamRP-LT in SiamRP-LT+Ft in metod iz tekmovanja VOT-LT2018 [16]. Rdeča pisava predstavlja najboljši rezultat, modra drugo, zelena pa tretje mesto.



Slika 4.5: Slika prikazuje povprečno krivuljo preciznost-priklic eksperimentov za vse sledilnike iz tekmovanja VOT-LT2018 in sledilnikov SiamRP-LT in SiamRP-LT+Ft.

Poglavje 5

Sklepne ugotovitve

V delu predlagamo metodo sledenja poljubnih objektov na sekvenci slik. Metoda temelji na uporabi siamske arhitekture konvolucijske nevronske mreže, kjer s predlogo poiščemo lokacijo sledenega objekta. S križno korelacijo izmerimo ujemanje med predlogo in tenzorjem značilk slike po prehodu skozi konvolucijsko nevronsko mrežo. Predloga je tenzor značilk, ki predstavlja izsek objekta sledenja iz prve slike po prehodu skozi konvolucijsko nevronsko mrežo. Predlagamo metodo posodabljanja okvirja, kjer uporabimo neko določeno število predlaganih regij in za okvir izberemo tisto, katere korelacija s predlogo je največja. Dodali smo še merjenje sigurnosti pravilne detekcije, kar nam omogoči izvajanje dolgoročnega sledenja. Predlagamo tudi dve metodi za prilagajanje na barvno predstavitev tarče in ozadja, kjer ena posodablja predlogo, druga pa fino prilagaja parametre mreže, da poveča korelacijo med predlogo in izgledom tarče, ko je detekcija dovolj gotova.

Analiziramo kako na uspešnost dolgoročnega sledenja vplivajo različne nastavitve velikosti in razmerij stranic predlaganih regij. Rezultati so pokazali, da je dobra nastavitev predlaganih regij taka, da so razmaki velikosti in razmerij stranic manjši. Tako se okvir bolj pogosto posodobi, vendar prilagoditev predstavlja manjšo spremembo. Analizirali smo tudi kako se obneseta predlagani metodi za prilagajanje na vizualno predstavitev tarče. Izkazalo se je, da posodabljanje predloge precej poslabša uspešnost dolgoročnega sle-

denja. Fino prilagajanje parametrov mreže pa poveča preciznost in priklic sledenja, vendar je bolj časovno in prostorsko zahtevno.

Predlagano metodo smo primerjali z metodami, ki so se udeležile tekmovanja VOT-LT2018 [16]. Metoda SiamRP-LT+Ft, ki uporablja fino prilagajanje, se je uvrstila na 11. mesto od 15 objavljenih metod.

5.1 Nadaljnje delo

Po analizi delovanja smo razmišljali kako bi povečali uspešnost sledenja. Glede na uspeh pri dodajanju finega prilagajanja parametrov merže, bi lahko to funkcijo izboljšali tako, da bi med sledenjem hranili pozitivne in negativne vzorce in tako zbrali večjo učno množico. To bi zmanjšalo verjetnost prevelike prilagoditve parametrov in bi bila mreža manj občutljiva na spremembo izgleda sledenega objekta.

Ker je SiamRP-LT precej počasen, bi lahko za pohitritev uporabili manj globoko mrežo. Sedaj uporabimo mrežo, ki je naučena za klasifikacijo slik v 1000 različnih razredov. Lahko pa bi uporabili bolj plitvo mrežo, ki bi jo naučili klasifikacije v 2 razreda, ospredje in ozadje. Nato pa bi med sledenjem s finim prilagajanjem naučili mrežo na prepoznavanje specifičnega objekta, ki mu sledimo.

Tretja možno izboljšava pa je uporaba regresorja. Naučili bi regresor, ki bi predlagal zamik in skalo predlaganih regij. Tako bi izboljšali prilagajanje okvirja, kar bi zvišalo natančnost sledenja.

Literatura

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] L Ambrosio and G Dal Maso. A general chain rule for distributional derivatives. *Proceedings of The American Mathematical Society - PROC AMER MATH SOC*, 108:691–691, 03 1990.

[3] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. *arXiv preprint arXiv:1606.09549*, 2016.

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550, June 2010.

[5] L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[6] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[7] François Chollet et al. Keras. `https://keras.io`, 2015.

[8] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

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

[11] Rajendra Kachhava, Vivek Shrivasta, Rajkumar Jain, and Ekta Chaturvedi. Security system and surveillance using real time object tracking and multiple cameras. *Advanced Materials Research*, 403-408:4968–4973, 11 2011.

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

[13] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94, Oct 1999.

[14] L. Kong and R. Dai. Object-detection-based video compression for wireless surveillance systems. *IEEE MultiMedia*, 24(2):76–85, Apr.-June 2017.

[15] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pfugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, Gustavo Fernandez, and et al. The sixth visual object tracking vot2018 challenge results, 2018.

[16] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, Nov 2016.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[18] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[19] Fei-Fei Li, Andrej Karpathy, and Justin Johnson. Cs231n: Convolutional neural networks for visual recognition. 2017. [Dostopano: 20. 1. 2019].

[20] Alan Lukezic, Luka Cehovin Zajc, Tomás Vojír, Jiri Matas, and Matej Kristan. FCLT - A fully-correlational long-term tracker. *CoRR*, abs/1711.09594, 2017.

[21] Alan Lukezic, Luka Cehovin Zajc, Tomás Vojír, Jiri Matas, and Matej Kristan. Now you see me: evaluating performance in long-term visual tracking. *CoRR*, abs/1804.07056, 2018.

[22] J. Mooser, S. You, and U. Neumann. Real-time object tracking for augmented reality combining graph cuts and optical flow. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 145–152, Nov 2007.

[23] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

[24] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. *CoRR*, abs/1510.07945, 2015.

[25] Akshay Rangesh and Mohan M. Trivedi. No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras & lidars. *CoRR*, abs/1802.08755, 2018.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 06 2015.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[29] Jack Valmadre, Luca Bertinetto, João F. Henriques, Ran Tao, Andrea Vedaldi, Arnold W. M. Smeulders, Philip H. S. Torr, and Efstratios Gavves. Long-term tracking in the wild: A benchmark. *CoRR*, abs/1803.09502, 2018.

[30] Anh Vo. Deep learning – computer vision and convolutional neural networks, 2018. [Dostopano: 25. 1. 2019].

[31] Naiyan Wang, Siyi Li, Abhinav Gupta, and Dit-Yan Yeung. Transferring rich feature hierarchies for robust visual tracking. *CoRR*, abs/1501.04587, 2015.

[32] Yang Wang, Bogdan Georgescu, Terrence Chen, Wen Wu, Peng Wang, Xiaoguang Lu, Razvan Ionasec, Yefeng Zheng, and Dorin Comaniciu. *Learning-Based Detection and Tracking in Medical Imaging: A Probabilistic Approach*, volume 7, pages 209–235. 01 2013.

[33] J. Xu, C. Zhang, and S. Goto. Object tracking by detection for video surveillance systems based on modified codebook foreground detection and particle filter. In *2011 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*, pages 1–6, Dec 2011.