Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Boštjan Bohte

Robustno sledenje s konvolucijskimi nevronskimi mrežami

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matej Kristan

Ljubljana, 2018

Zahvala

Zahvaljujem se mentorju izr. prof. dr. Mateju Kristanu za vodstvo pri izdelavi naloge in vsem ostalim, ki so pomagali pri izdelavi magistrske naloge.

Boštjan Bohte, 2018

Vsem.

"Our lives sometimes depend on computers performing as predicted."

— Philip Emeagwali

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CA	classification accuracy	klasifikacijska točnost
FP	false positive	nepravilno pozitivno
FN	false positive	nepravilno pozitivno
TN	true positive	pravilno negativno
TP	true positive	pravilno pozitivno
CNN	convolutional neural network	konvolucijska nevronska mreža
NN	neural net	nevronska mreža
FOR	false ommision rate	delež napačno omitiranih
FDR	false discovery rate	delež napačno zaznanih
FPR	false positive rate	delež napačno zaznanih kot pozitivnih
FNR	false negative rate	delež napačno zaznanih kot negativnih

Povzetek

Naslov: Robustno sledenje s konvolucijskimi nevronskimi mrežami

Sledenje objektov je proces lokalizacije objekta(ov) skozi sekvenco slik. Mnogo uspešnih sledilnikov za sledenje uporablja konvolucijske nevronske mreže, ki so sposobne razpoznavati objekte na višjem abstraktnem nivoju. Poznamo kratkoročne in dolgoročne sledilnike, pri katerih se slednji razlikujejo po značilnosti, ki ob primeru odpovedi sledenja ponovno nastavi sledilnik. V našem delu smo se osredotočili na izboljšavo sledilnika, ki z uporabo konvolucijske nevronske mreže sicer doseže hitro in natančno sledenje, vendar nima možnosti dolgoročnega sledenja. Predlagamo nov sledilnik z implementirano detekcijo odpovedi sledenja. Ta napove verjetnost pravilnega oziroma nepravilnega trenutnega sledenja. Na podlagi te detekcije pa nato implementiramo detektor objekta, ki v primeru zaznane odpovedi sledenja poišče sledeni objekt na sliki in ponastavi sledilnik. S tem izpolnimo zahteve dolgoročnega sledilnika. Za še bolj robustno dolgoročno sledenje predlagamo tudi dva načina posodabljanja predloge. Pri prvem načinu posodabljamo predlogo s shranjevanjem predlog v vrsto, pri drugem pa s postopnim posodabljanjem predloge z novimi primeri. Na trenutno edini podatkovni zbirki za dolgoročno sledenje predlagani sledilnik dosega najboljše rezultate, ki so 24% boljši od trenutno najboljšega objavljenega sledilnika.

Ključne besede

mreže, sledilnik, objekt, sledenje, detekcija, model

Abstract

Title: Robust tracking with convolutional neural networks

Visual object tracking is a process of object(s) localization through the sequence of images. Many successful trackers use convolutional neural networks for tracking. These networks are capable of recognizing object features on an abstract level. We can define trackers as short term and long term trackers with the latter having an additional function which reinitializes their tracking in case of a failure. In our work, we want to improve the tracker that uses convolutional neural network which is already accurate and fast but does not offer the possibility of a long term tracking. We propose a new tracker with a tracking failure detection. This detection predicts with plausibility if the tracker is tracking the object correctly or incorrectly. Based on implemented failure detection, we implement the object detection which finds the tracking object on the image and reinitializes the tracker in case of a predicted tracking failure. With these two features implemented, we fulfill the requirements for the long term tracker. For even more robust long term tracking we propose two methods of updating the template. With the first method, we save templates in an array while with the second method we gradually update the initial template with new examples. We scored the best result in the so far only existing database for long term tracking evaluation. The results are 24% better than those of the currently best published tracker.

Keywords

networks, tracker, object, tracking, detection, model

Poglavje 1

Uvod

Sledenje objektov je proces lokalizacije objekta(ov) skozi sekvenco slik. Uporaba je široko razširjena pri interakciji človek-stroj [1], na področjih, kot so varnostno nadzorovanje [2], kontrola prometa [3], obogatena resničnost [4] itd. Pri sledenju objekta iz realnega sveta se kompleksnost sledenja skozi čas povečuje, saj se spreminja tako objekt, kot tudi njegovo ozadje. Potrebne so metode, ki uspejo kljub transformaciji objekta in ozadja zaznati objekt ter njegove spremembe shranijo za nadaljnjo sledenje (glej [5]). Sledenje je lahko časovno zahtevno, zato je izbira metode pomembna glede na namen uporabe sledilnika. Če nam je pomembno natančno sledenje in ne njegov čas izvajanja, lahko izberemo počasnejši, a bolj natančen sledilnik. V primeru sledenja v realnem času pa moramo izbrati hiter sledilnik, ki pa ni nujno natančen.

Sledenje poteka na sekvenci slik, kjer pri začetni sliki inicializiramo sledilnik s predlogo sledečega objekta. Sledilnik nato poskuša samostojno slediti objektu z vsako naslednjo iteracijo. Da bi sledilnik čim dlje ohranil pravilno sledenje objekta, so bile razvite različne metode, ki pomagajo pri razpoznavnosti objekta ali predvidevajo njegove položaje. Zaradi vedno večje računske moči in učinkovitosti so se pojavile metode sledenja s konvolucijskimi nevronskimi mrežami [6, 7, 8, 9]. Njihova učinkovitost se kaže v tem, da lahko, za razliko od drugih metod, prepoznajo objekt na abstraktnem nivoju. Tako

lahko sledeni objekti med sledenjem spreminjajo svoj videz preko deformacij in so kljub temu dobro prepoznani.

1.1 Pregled sorodnih del

Veliko sledilnikov je že bilo implementiranih z uporabo konvolucijskih nevronskih mrež, na primer [6, 7, 8, 9]. Sledilnike se lahko uporablja z namenom kratkoročnega sledenja (angl., short-term tracking) ali sledenja na dolgi rok (angl., long-term tracking). Slednji mora detektirati lastno odpoved sledenja in ponovno detektirati sledeči objekt, tudi ko je ta že dlje časa odsoten. V tem razdelku bomo pregledali samo tiste metode, katerih lastnosti želimo imeti tudi v našem modelu.

V članku [6] za sledenje objekta predlagajo uporabo primerjalne funkcije, ki je sestavljena iz polno konvolucijske siamske nevronske mreže [10]. Ta sprejme na en vhod sliko objekta in na drug iskalno sliko ter vrne matriko, kjer vsaka celica predstavlja možno lokacijo, njene vrednosti pa verjetje (angl., likelihood) pojavitve objekta. Ker model v eni interakciji detektira in lokalizira objekt, je zato hitrejši od ostalih sodobnih sledilnikov. Model ne uporablja posodabljanja ter je kljub temu robusten pri številnih zahtevnih problemih, kot so večje spremembe podobe objekta, slaba osvetlitev, sprememba v velikosti in zameglitev pri premikanju. Ta sledilnik bomo v delu podrobneje preučili in na podlagi tega naredili novi dolgoročni sledilnik.

V članku [7] naučijo primerjalno funkcijo, ki med dvema slikama vrne visoko vrednost, če sta si podobni, sicer pa nizko. Model vzorči sliko, kjer je pojavitev objekta najbolj verjetna, ter primerja te vzorce s primerjalno funkcijo, nato pa izbere lokacijo tistega vzorca, ki je dosegel največjo vrednost. Dobra lastnost predlaganega modela je zmožnost ponovno detektirati objekt, tudi ko je ta že dlje časa odsoten. To pa je prvina sledilnikov na dolgi rok [11].

V članku [26] predlagajo uporabo dveh ločenih niti za sledenje, kjer se prva nit izvaja hitro, a ne tako natančno, medtem ko je druga nit počasnejša

in popravlja sledenje prve niti, v kolikor je to potrebno. Dobra lastnost uporabe dveh ločenih niti je ta, da ni potrebno počasnejše niti izvajati pri vsaki iteraciji, ampak samo na vsakih N slik. Tako se lahko za drugo nit uporabi model, ki natančno prepozna značilke objekta, pri tem pa še vedno ohrani sledenje v realnem času. Rezultati so pokazali, da so z omenjeno metodo dosegli boljše sledenje od vseh ostalih realno časovnih sledilnikov in tudi nekaj počasnejših sledilnikov.

1.2 Prispevki

V delu se osredotočamo na kratkoročni sledilnik SiamFC [6] in na podlagi tega naredimo nov dolgoročni sledilnik. Sledilnik SiamFC je kratkoročni sledilnik, ki hitro in robustno detektira sledeči objekt, vendar nima možnosti detektirati odpovedi sledenja. V našem delu nadgradimo sledilnik SiamFC z detekcijo odpovedi sledenja in detekcijo objekta, ki ob zaznani odpovedi sledenja ponovno poišče sledeni objekt. Tako dobimo nov sledilnik, ki je hiter in hkrati dovolj zanesljiv, da omogoča dolgoročno sledenje.

1.3 Struktura naloge

Magistrska naloga je razdeljena na šest poglavij. V Poglavju 2 predstavimo konvolucijske nevronske mreže, ki jih uporabljamo v našem delu. Nato v Poglavju 3 predstavimo sledilnik, ki uporablja arhitekturo polno konvolucijske siamske nevronske mreže za sledenje. V Poglavju 4 predlagamo izboljšave nadgradnje sledilnika iz Poglavja 3, kjer najprej implementiramo klasifikator, ki omogoča detekcijo odpovedi sledenja. Tega nato uporabimo pri metodah za izboljšanje sledenja, kot je uporaba detektorja objekta ob odpovedi sledenja ter posodabljanje predloge. V Poglavju 5 izvedemo eksperimentalno evalvacijo s testiranjem naših izboljšav na različnih testnih množicah. V zadnjem Poglavju 6 pa opišemo naše ugotovitve.

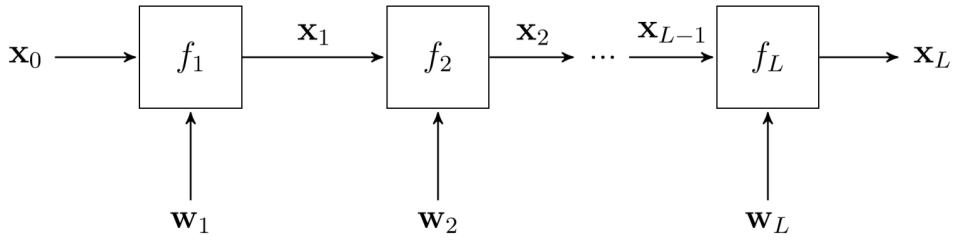
Poglavje 2

Konvolucijske nevronske mreže

Konvolucijske nevronske mreže (angl. convolutional neural networks, CNN) imajo pomemben vpliv na področju računalniškega vida in razumevanja slik na splošno. Njihova učinkovitost je tako dobra, da zamenjujejo obstoječe metode pridobivanja značilk iz slik, kot je na primer SURF [12]. Čeprav je večina CNN pridobljenih s kompozicijo enostavnih linearnih in nelinearnih filtrov, je njihova implementacija daleč od trivialne. Za učenje CNN je potrebno imeti veliko učno množico, ki velikokrat vsebuje tudi več milijonov primerkov. Da lahko izvajamo CNN na tako velikih učnih množicah, rabimo ustrezna orodja, ki optimizirajo uporabo in, kar je najpomembneje, izvajajo računanje na grafičnih karticah [13, 14]. V prvem delu poglavja opišemo nevronske mreže, nato opišemo CNN in vse njene funkcije, ki jih uporabljamo v našem delu, v nadaljevanju pa še opišemo križno korelacijo in arhitekturo siamske CNN.

2.1 Nevronske mreže

Nevronska mreža (angl. Neural Network, NN) je funkcija $g(\cdot)$, ki slika podatke $\boldsymbol{x}$ (na primer sliko) v vektor $\boldsymbol{y}$, ki je oznaka (angl. *label*) slike. Funkcija $g(\cdot) = f_L \circ ... \circ f_1$ je kompozicija zaporedij enostavnejših funkcij $f_l(\cdot)$, ki se imenujejo računski bloki ali nivoji. Naj bodo $\boldsymbol{x_1}, \boldsymbol{x_2}, .., \boldsymbol{x_L}$ izhodi vsakega



Slika 2.1: Slika prikazuje enostavnejši primer nevronske mreže, kjer so računski bloki zaporedno povezani. Slika povzeta po [14].

nivoja mreže in naj $\boldsymbol{x_0} = \boldsymbol{x}$ predstavlja vhod mreže. Vsak vmesni izhod $\boldsymbol{x_l} = f_l(\boldsymbol{x_{l-1}}; \boldsymbol{w_l})$ je izračunan iz prejšnjega izhoda $\boldsymbol{x_{l-1}}$ z uporabo funkcije $f_l(\cdot)$ in parametra $\boldsymbol{w_l}$, kot prikazuje Slika 2.1, kjer vidimo enostavnejši primer zaporedno povezanih računskih blokov nevronske mreže.

2.2 Konvolucijske nevronske mreže

Pri CNN imajo podatki prostorsko strukturo. Vsak $\boldsymbol{x_l} \in \mathbb{R}^{H_l \times W_l \times C_l}$ je 3D seznam, kjer prvi dve dimenziji predstavljata višino (H_l) in širino (W_l) ter sta interpretirani kot prostorski dimenziji. Tretja dimenzija C_l pa predstavlja število kanalov značilk. Torej seznam $\boldsymbol{x_l}$ predstavlja C_l dimenzionalni vektor značilk polja $H_l \times W_l$ za vsako prostorsko lokacijo. Četrta dimenzija N_l v seznamu združuje več primerkov, združenih v en paket (angl. batch), za učinkovito paralelno obdelavo. Mreža se imenuje konvolucijska zato, ker je funkcija $f_l(\cdot)$ lokalno in translacijsko neodvisen operator. Pri CNN je možno uporabiti tudi več prostorskih dimenzij, kjer na primer dodatni parameter pomeni čas, saj ni predpisanih restrikcij glede samega formata podatkov.

2.2.1 Vzvratni prehod

NN se pogosto uporablja kot klasifikatorje ali regresije. Če izhod $\boldsymbol{y}$ predstavlja vektor verjetnosti za vsak razred in če je $\boldsymbol{c}$ oznaka slike $\boldsymbol{x}$, lahko

s kriterijsko funkcijo $\ell(\boldsymbol{y}, \boldsymbol{c}) \in \mathbb{R}$, ki "kaznuje"napačne napovedi, izmerimo učinkovitost CNN. Parametre CNN se lahko nastavi ali nauči tako, da minimizirajo povprečno napako kriterijske funkcije nad učno množico, ki vsebuje vnaprej označene primere. Za učenje se uporablja metoda stohastičnega spusta po gradientu. Odvodi so izračunani z uporabo algoritma *vzvratni prehod*, ki uporablja učinkovit pristop verižnega odvajanja.

V CNN je nivo funkcija $\boldsymbol{y} = f(\boldsymbol{x})$ kjer sta vhod $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ in izhod $\boldsymbol{y} \in \mathbb{R}^{H' \times W' \times C'}$ tenzorja. Odvod funkcije $f(\cdot)$ vsebuje odvode vsake izhodne komponente $\boldsymbol{y_{i'j'k'}}$ glede na vhodno komponento $\boldsymbol{x_{ijk}}$, kjer je skupno število elementov enako $H' \times W' \times C' \times H \times W \times C$. Namesto da se odvodi računajo kot tenzor, je bolje zamenjati tenzor v matrično obliko s kopičenjem vhodnih in izhodnih tenzorjev v vektor. To stori operator *vec*, ki leksikografsko uredi elemente v vektor, na primer,

$$vec\, \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_{111}} \\ \boldsymbol{x_{211}} \\ \vdots \\ \boldsymbol{x_{H11}} \\ \boldsymbol{x_{121}} \\ \vdots \\ \boldsymbol{x_{HWC}} \end{bmatrix}. \tag{2.1}$$

S kopičenjem vhodov in izhodov se lahko vsak nivo $f(\cdot)$ ponovno interpretira kot vektorska funkcija $vecf$, katere odvod je konvencionalna Jakobijeva

matrika, na primer

$$\frac{d\,vec\,f}{d(vec\,\boldsymbol{x})^T} = \begin{bmatrix} \frac{\partial y_{111}}{\partial x_{111}} & \frac{\partial y_{111}}{\partial x_{211}} & \cdots & \frac{\partial y_{111}}{\partial x_{H11}} & \frac{\partial y_{111}}{\partial x_{121}} & \cdots & \frac{\partial y_{111}}{\partial x_{HWC}} \\ \frac{\partial y_{211}}{\partial x_{111}} & \frac{\partial y_{211}}{\partial x_{211}} & \cdots & \frac{\partial y_{211}}{\partial x_{H11}} & \frac{\partial y_{211}}{\partial x_{121}} & \cdots & \frac{\partial y_{211}}{\partial x_{HWC}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial y_{H'11}}{\partial x_{111}} & \frac{\partial y_{H'11}}{\partial x_{211}} & \cdots & \frac{\partial y_{H'11}}{\partial x_{H11}} & \frac{\partial y_{H'11}}{\partial x_{121}} & \cdots & \frac{\partial y_{H'11}}{\partial x_{HWC}} \\ \frac{\partial y_{121}}{\partial x_{111}} & \frac{\partial y_{121}}{\partial x_{211}} & \cdots & \frac{\partial y_{121}}{\partial x_{H11}} & \frac{\partial y_{121}}{\partial x_{121}} & \cdots & \frac{\partial y_{121}}{\partial x_{HWC}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial y_{H'W'C'}}{\partial x_{111}} & \frac{\partial y_{H'W'C'}}{\partial x_{211}} & \cdots & \frac{\partial y_{H'W'C'}}{\partial x_{H11}} & \frac{\partial y_{H'W'C'}}{\partial x_{121}} & \cdots & \frac{\partial y_{H'W'C'}}{\partial x_{HWC}} \end{bmatrix}. \tag{2.2}$$

Z matrično obliko je enostavno izračunati odvode tenzorja, vendar so te matrike lahko ekstremno velike. Tudi pri običajno velikih podatkih (na primer $H = H' = W = W' = 32$ in $C = C' = 128$) imamo približno 17×10^9 elementov v Jakobijevi matriki, za kar zahteva 68 GB prostora. Smisel algoritma vzvratnega prehoda je izračun odvodov brez zahteve po tolikšnem prostoru.

Da bi razumeli vzvratni prehod, si predstavljajmo enostavno CNN, ki se zaključi v kriterijski funkciji $f_L(\cdot) = \ell_y(\cdot)$. Cilj je izračunati gradient napake izhoda $\boldsymbol{x_L}$ glede na vsak parameter $\boldsymbol{w_l}$:

$$\frac{df}{d(vec\,\boldsymbol{w_l})^T} = \frac{d}{d(vec\,\boldsymbol{w_l})^T}[f_L(.;\boldsymbol{w_L}) \circ \cdots \circ f_2(;\boldsymbol{w_2}) \circ f_1(\boldsymbol{x_0};\boldsymbol{w_1})]. \tag{2.3}$$

Z uporabo verižnega pravila in matrične notacije, predstavljene zgoraj, se odvod izračuna po enačbi

$$\frac{df}{d(vec\,\boldsymbol{w_l})^T} = \frac{d\,vec\,f_L(\boldsymbol{x_{L-1}};\boldsymbol{w_L})}{d(vec\,\boldsymbol{x_{L-1}})^T} \times \cdots \\ \cdots \times \frac{d\,vec\,f_{l+1}(\boldsymbol{x_l};\boldsymbol{w_{l+1}})}{d(vec\,\boldsymbol{x_l})^T} \times \frac{d\,vec\,f_l(\boldsymbol{x_{l-1}};\boldsymbol{w_l})}{d(vec\,\boldsymbol{x_l}^T)}, \tag{2.4}$$

kjer so odvodi izračunani na mestu, ki ga je določil vhod x_0 in trenutna vrednost parametra. V enačbi (2.4) je izhod x_l skalar in elementi ciljnega

odvoda $df/d(vec\,\boldsymbol{w_l})^T$ so enake velikosti kot vektor parametrov $\boldsymbol{w_l}$, kar pa je sprejemljiva velikost. Kljub temu pa ima Jakobijeva matrika preveliko velikost za hitro računanje z računalnikom. Da se izognemo eksplicitnemu računanju faktorjev, se jih lahko lotimo po naslednjem postopku.

Začnemo z množenjem izhoda zadnjega nivoja s tenzorjem $p_L = 1$ (tenzor je tukaj skalar, tako kot spremenljivka x_L).

$$\begin{aligned} p_L \times \frac{df}{d(vec\,\boldsymbol{w_l})^T} &= p_L \times \frac{d\,vec\,f_L(\boldsymbol{x_{L-1}};\boldsymbol{w_L})}{d(vec\,\boldsymbol{x_{L-1}})^T} \times \dots \\ &\quad \dots \times \frac{d\,vec\,f_{l+1}(\boldsymbol{x_l};\boldsymbol{w_{l+1}})}{d(vec\,\boldsymbol{x_l})^T} \times \frac{d\,vec\,f_l(\boldsymbol{x_{l-1}};\boldsymbol{w_l})}{d(vec\,\boldsymbol{w_l}^T} = \\ &= (vec\ , \boldsymbol{p_{L-1}})^T \times \dots \times \frac{d\,vec\,f_{l+1}(\boldsymbol{x_l};\boldsymbol{w_{l+1}})}{d(vec\,\boldsymbol{x_l})^T} \times \frac{d\,vec\,f_l(\boldsymbol{x_{l-1}};\boldsymbol{w_l})}{d(vec\,\boldsymbol{w_l}^T)}, \end{aligned} \quad (2.5)$$

kjer sta bila v tretji vrstici zadnja faktorja pomnožena in tvorita novi tenzor p_{L-1}, ki ima isto velikost kot spremenljivka $\boldsymbol{x_{L-1}}$. Faktor $\boldsymbol{P_{L-1}}$ se lahko eksplicitno shrani. Tak postopek ponavljamo ter izračunamo tenzorje $\boldsymbol{p_{L-2}}, \dots, \boldsymbol{p_l}$ dokler ni izračunan želeni odvod. Pri tem pa nikoli ne shranimo velikega tenzorja v spomin. Za vzvratni prehod si lahko predstavljamo projekcijo p kot linearizacijo celotne mreže od spremenljivke $\boldsymbol{y}$ do končnega vhoda. Projicirani odvod pa si lahko predstavljamo kot novi nivo $(d\boldsymbol{x}, d\boldsymbol{w}) = df(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{p})$ v obratni smeri računanja.

2.3 Računski bloki

V tem podpoglavju bomo opisali računske bloke, ki smo jih uporabili v našem delu. V našem primeru se računske bloke uporablja kot funkcije $\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{w})$, kjer seznama $\boldsymbol{x}$ in $\boldsymbol{w}$ predstavljata podatke in parametre, medtem ko $\boldsymbol{y}$ predstavlja izhodni seznam. V splošnem sta $\boldsymbol{x}$ in $\boldsymbol{y}$ realna 4D seznama, zapakirana v N map ali slik, kjer je $\boldsymbol{w}$ lahko poljubne oblike. Različne funkcije lahko, v kolikor je to potrebno, uporabljajo malo drugačno sintakso. Veliko funkcij lahko sprejme dodatni argument, nekatere ne rabijo parametrov, druge pa lahko sprejmejo več vhodov in parametrov.

2.3.1 Konvolucija

Konvolucijski blok računa konvolucijo vhodne mape $\boldsymbol{x}$ s skupino K več dimenzionalnih filtrov $\boldsymbol{f}$ in nagnjenju (angl. biases) $\boldsymbol{b}$, ki so definirani kot

$$\boldsymbol{x} \in \mathbb{R}^{H \times W \times D}, \boldsymbol{f} \in \mathbb{R}^{H' \times W' \times D \times D''}, \boldsymbol{y} \in \mathbb{R}^{H'' \times W'' \times D''}. \tag{2.6}$$

Rezultat konvolucije pa se nato izračuna kot

$$\boldsymbol{y}_{i''j''d''} = \boldsymbol{b}_{d''} + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d'=1}^{D} \boldsymbol{f}_{i'j'd} \times \boldsymbol{x}_{i''+i'-1, j''+j'-1, d''}. \tag{2.7}$$

Pri računanju konvolucije mora biti celoten filter znotraj mape, zato je izhodna mapa manjša od vhodne. Če tega ne želimo, moramo vhodni mapi dodati obrobe (angl. padding). Te dodajo ničle na robove vhodne mape $(P_h^-, P_h^+, P_w^-, P_w^+)$. V kolikor pa želimo imeti izhodno mapo manjšo, uporabimo podvzorčenje (S_h, S_w). Z uporabo obrob ali podvzorčenja se izhod izračuna kot

$$\boldsymbol{y}_{i''j''d''} = \boldsymbol{b}_{d''} + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d'=1}^{D} \boldsymbol{f}_{i'j'd} \times \boldsymbol{x}_{S_h(i''-1)+i'-P_h^-, S_w(j''-1)+j'-P_w^-, d', d''}. \tag{2.8}$$

Velikost izhodne mape pa se izračuna po enačbi

$$H'' = 1 + \lfloor \frac{H - H' + P_h^- + P_h^+}{S_h} \rfloor. \tag{2.9}$$

Pri tem pa mora biti vhod vsaj toliko velik, kot so veliki filtri, tako da je $H + P_h^- + P_h^+ \leq H'$.

Polno povezani bloki ali nivoji so linearne funkcije, kjer je vsak izhod povezan z vsemi vhodi. V našem delu ne razlikujemo med konvolucijskimi bloki in polno povezanimi bloki. Slednji je poseben primer, kjer ima izhodna mapa $\boldsymbol{y}$ dimenzije 1×1.

2.3.2 Nivo združevanja

Naloga bloka prostorskega združevanja je zmanjšanje lokacijske odvisnosti značilk [15]. Ta za vsak odziv v mapi, in njegovo določeno okolico, izračuna

maksimalno ali povprečno vrednost. Z združevanjem se zmanjša prostorska dimenzija vhodom v naslednji nivo. S tem se tudi omili preveliko prilagajanje učnim podatkom. Ker so vse operacije vnaprej določene, se ta blok ne uči. Primer maksimalnega združevanja prikazuje enačba

$$\boldsymbol{y}_{i''j''d} = \max_{1 \leq i' \leq H', 1 \leq j' \leq W'} \boldsymbol{x}_{i''+i-1, j''+j'-1, d}. \tag{2.10}$$

Pri obrobah in podvzorčenju je podobno kot pri konvolucijskem nivoju. Pri tem pa je odziv na obrobo drugačen. Pri maksimalnem združevanju imajo obrobe vrednost $-\inf$, medtem ko imajo pri povprečnem združevanju vrednosti 0.

2.3.3 Enota ReLu

Uporaba aktivacijske funkcije vpelje modelu nelinearnost, saj bi se v drugačnem primeru CNN obnašala kot enoslojni perceptron. Obstaja več vrst aktivacijskih funkcij, kot sta na primer sigmoidna funkcija in hiperbolični tangens. V našem delu pa uporabljamo aktivacijsko funkcijo ReLU (angl. Rectified Linear Unit), ki vrne rezultat po enačbi

$$\boldsymbol{y}_{ijd} = max\{\, 0, \boldsymbol{x}_{ijd}\} \,. \tag{2.11}$$

Pri globokih nevronskih mrežah, ki se učijo na velikih in kompleksnih učnih podatkih, se je uporaba funkcije ReLU izkazala za boljšo rešitev kot logistična sigmoidna funkcija ali hiperbolični tangens [16].

2.3.4 Paketna normalizacija

Paketna normalizacija [17] je drugačna od ostalih računskih blokov, ker izvaja operacije čez celotno skupino map, medtem ko jih večina izvaja za vsako mapo posebej. Ta normalizira vsak kanal mape $\boldsymbol{x}$, tako da izračuna povprečje prostorske lokacije in skupinske instance. Izhodno mapo se izračuna tako, da

spremenljivka T predstavlja velikost skupine $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{H \times W \times K \times T}$, $\boldsymbol{w} \in \mathbb{R}^K$, $\boldsymbol{b} \in \mathbb{R}^K$, nato pa se jo izračuna po enačbi

$$\boldsymbol{y}_{ijkt} = \boldsymbol{w}_k \frac{\boldsymbol{x}_{ijkt} - \boldsymbol{\mu}_k}{\sqrt{\boldsymbol{\sigma}_k^2 + \epsilon}} + \boldsymbol{b}_k, \tag{2.12}$$

kjer se $\boldsymbol{\mu}_k$ in $\boldsymbol{\sigma}_k^2$ izračunata po naslednjih enačbah:

$$\boldsymbol{\mu}_k = \frac{1}{HWT} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{t=1}^{T} \boldsymbol{x}_{ijt}, \tag{2.13}$$

$$\boldsymbol{\sigma}_k^2 = \frac{1}{HWT} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{t=1}^{T} (\boldsymbol{x}_{ijkt} - \boldsymbol{\mu}_k)^2. \tag{2.14}$$

2.3.5 Funkcija Softmax

Funkcija Softmax je razširjena logistična funkcija, ki realne vrednosti K dimenzionalnega vektorja $\boldsymbol{\sigma}(\boldsymbol{z})$ spremeni v vrednosti $[0, 1]$, ki se seštejejo v 1. V našem primeru nam te vrednosti predstavljajo napovedne verjetnosti posameznega razreda. Po navadi se jo uporabi na zadnjem nivoju CNNja in služi kot klasifikator. Izračuna se jo kot

$$\boldsymbol{y}_{ijk} = \frac{\mathrm{e}^{\boldsymbol{x}_{ijk}}}{\sum_{t=1}^{D} \mathrm{e}^{\boldsymbol{x}_{ijk}}}. \tag{2.15}$$

2.3.6 Kriterijska funkcija

Namen kriterijske funkcije $\ell(\boldsymbol{x}, \boldsymbol{c})$ je primerjava napovedi $\boldsymbol{x}$ s pravilno vrednostjo razreda oznake $\boldsymbol{c}$. Izguba je tretirana kot konvolucijski operator, v smislu, da je izguba neodvisno izračunana za vsako prostorsko lokacijo.

Logaritemska kriterijska funkcija dobi na vhod vektor $\boldsymbol{x}$, ki predstavlja verjetnosti za vsak razred posebej. Izguba je nato negativna logaritemska verjetnost pravilno napovedanega razreda

$$\ell(\boldsymbol{x}, \boldsymbol{c}) = -\log \boldsymbol{x}_c. \tag{2.16}$$

Odvod te funkcije pri vzvratnem razširjenju napake pa se izračuna po enačbi

$$\frac{\partial p\ell(\boldsymbol{x}, c)}{\partial \boldsymbol{x}_k} = -p\frac{\partial \log(\boldsymbol{x}_c)}{\partial \boldsymbol{x}_k} = -p\boldsymbol{x}_c\delta_{k=c}, \tag{2.17}$$

kjer je tenzor p skalar, saj vse kriterijske funkcije vrnejo skalarno vrednost.

2.4 Križna korelacija

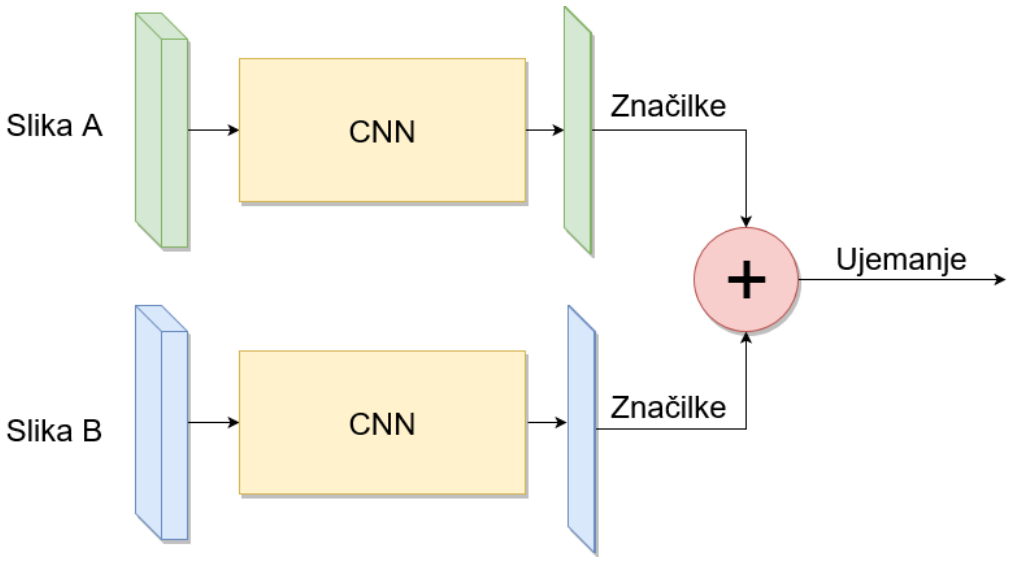
Križna korelacija je mera podobnosti dveh zaporedij $\boldsymbol{f}, \boldsymbol{g}$ kot funkcija pozicijskih sprememb ene relativno na drugo. Znana je tudi kot premikajoče okno vektorskega produkta. Za diskretne funkcije je križna korelacija definirana kot

$$(\boldsymbol{f} \star \boldsymbol{g})[n] \stackrel{\text{def}}{=} \sum_{m=-\inf}^{\inf} \boldsymbol{f}^*[m]\boldsymbol{g}[m+n]. \tag{2.18}$$

2.5 Arhitektura siamske CNN

Arhitektura siamske CNN je bila prvič predstavljena leta 1990 v delu [10], kjer so poskušali rešiti problem prepoznave ročnih podpisov z ujemanjem dveh slik. Vsebuje dve identični CNN s transformacijo $\varphi(\cdot)$, ki sprejmeta dva različna vhoda, nato pa se njuna izhoda povežeta s funkcijo $g(\cdot)$, ki vrne končni rezultat $f(\boldsymbol{x}, \boldsymbol{y}) = g(\varphi(x), \varphi(\boldsymbol{y}))$. Funkcija $g(\cdot)$ je lahko konvolucijska mreža ali pa je enostavna mera razdalje oziroma podobnosti. Uporaba dveh identičnih CNN ima dve ključni lastnosti. Prva lastnost je, da se zaradi enakih uteži in parametrov posamezne CNN zagotovi, da se dva podobna podatka na vhodu preslikata v podoben prostor značilk na izhodu. Druga lastnost pa je simetričnost, kar pomeni, da dobimo enake rezultate tudi v primeru, če zamenjamo njuna vhoda.

S to arhitekturo lahko na primer izračunamo podobnost dveh slik, kjer dvema podobnima slikama pripišemo visoko vrednost ujemanja in različnima slikama pripišemo nizko vrednost ujemanja. Slike gredo najprej čez vhoda CNN, kjer se na izhodu pretvorita v bolj abstraktne značilke. Podobnost med



Slika 2.2: Slika prikazuje primer uporabe arhitekture siamske CNN za izračun podobnosti vsebine dveh slik.

njima pa se lahko izračuna z enostavno funkcijo, na primer s križno korelacijo, saj se značilke nahajajo v istem prostoru, če so si slike vsebinsko podobne. Tak primer prikazuje Slika 2.2. V preteklosti so siamsko CNN predlagali za razpoznavanje obrazov [18], učenja ključnih značilk na sliki [19], prepoznavo ročnih podpisov [10] itd.

Poglavje 3

Sledenje s siamsko konvolucijsko mrežo

V delu [6] za sledenje predlagajo uporabo arhitekture siamske CNN. Naše delo temelji na njihovem sledilniku SiamFC, zato ga v tem poglavju podrobno opišemo. V prvem delu poglavja opišemo arhitekturo in njene prednosti. V drugem delu opišemo učenje siamske CNN. V tretjem delu pa opišemo sledenje z uporabo siamske CNN.

3.1 Arhitektura siamske CNN

Za sledenje se uporablja arhitektura siamske CNN, ki je polno konvolucijska glede na sliko predloge $\boldsymbol{x}$. Funkcija je polno konvolucijska, če se prevaja s prevodom oziroma če se vrednosti preslikajo tako, da se zaporedje vrednosti na vhodu ohrani tudi na preslikanem izhodu. Torej, če je L_τ operator prevajanja, sledi $(L_\tau \boldsymbol{x})[u] = \boldsymbol{x}[u - \tau]$. Funkcija $h(\cdot)$, ki povezuje vse signale, je polno konvolucijska s korakom k, če je

$$h(L_{k\tau x}) = L_\tau h(\boldsymbol{x}) \tag{3.1}$$

za vsak prevod τ.

Prednost polno konvolucijske funkcije je ta, da lahko na vhod podamo

Računski blok	**Velikost filtra**	**Stride**	**Primerek**	**Iskalna mapa**	**Št. kanalov**
			127×127	255×255	$\times 3$
Conv1	11×11	2	59×59	123×123	$\times 96$
pool1	3×3	2	29×29	61×61	$\times 96$
Conv2	5×5	1	25×25	57×57	$\times 256$
pool2	3×3	2	12×12	28×28	$\times 256$
Conv3	3×3	1	10×10	26×26	$\times 192$
Conv4	3×3	1	8×8	24×24	$\times 192$
Conv5	3×3	1	6×6	22×22	$\times 128$

Tabela 3.1: Tabela prikazuje arhitekturo polno konvolucijske siamske CNN sledilnika SiamFC.

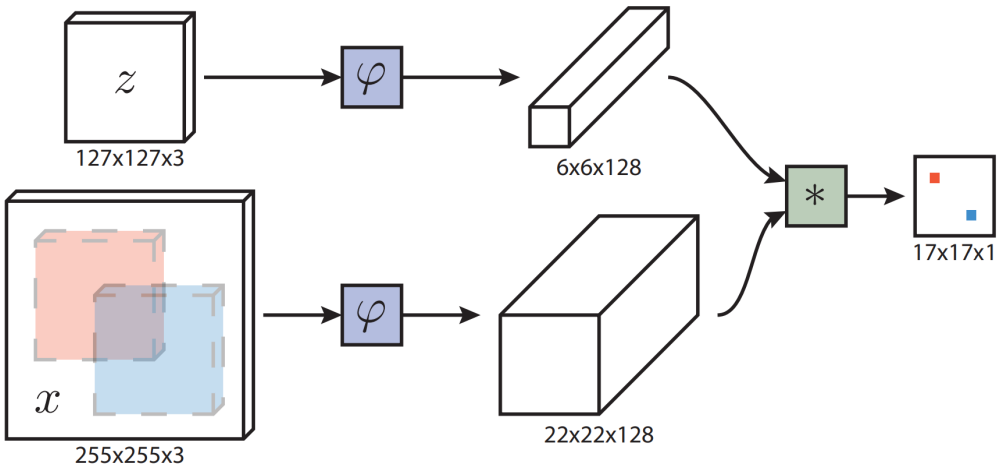
različno veliki sliki. Namesto slike primerka, ki je enake velikosti kot slika predloge, uporabimo kar poljubno veliko iskalno sliko, funkcija pa bo izračunala podobnost za vsak prevod posebej. Tako dobimo v eni iteraciji izračunane podobnosti primerka za več lokacij hkrati. To se doseže tako, da se združi vgrajeni konvolucijski funkciji $\varphi(\cdot)$ s križno korelacijo

$$f(\boldsymbol{z}, \boldsymbol{x}) = \varphi(\boldsymbol{z}) * \varphi(\boldsymbol{x}) + b\mathbb{1}, \tag{3.2}$$

kjer $b\mathbb{1}$ označuje signal, ki ima na vsaki lokaciji vrednost $b \in \mathbb{R}$.

Izhod take mreže ni en sam izhod, ampak je končno velika matrika $\boldsymbol{D} \subset \mathbb{Z}^2$, kjer vsaka vrednost v matriki $\boldsymbol{D}$ predstavlja podobnost predloge in primerka, koordinate pa izdajo njihovo lokacijo. Koordinate na matriki $\boldsymbol{D}$ je nato potrebno še pretvoriti v koordinate celotne slike. Delovanje take polno povezane CNN prikazuje Slika 3.1.

Arhitektura polno konvolucijske siamske CNN, ki jo uporablja omenjeni sledilnik, je podana v Tabeli 3.1. V času učenja je po vsakem ReLu dodana še paketna normalizacija. Aktivacijska funkcija ReLu je, z izjemo zadnjega, prisotna po vsakem konvolucijskem bloku. Pri omenjeni arhitekturi se ne uporablja obrob, saj bi s tem kršili pravilo polno konvolucijske mreže, ki je definirano v enačbi (3.2).



Slika 3.1: Slika prikazuje delovanje polno konvolucijske siamske CNN. Vhodni podatek $\boldsymbol{z}$ pomeni sliko predloge, $\boldsymbol{x}$ pa iskalno sliko. Končni rezultat je matrika $\boldsymbol{D}$. Slika je povzeta po [6].

3.2 Učenje

Za učenje omenjenega modela so v delu [6] uporabili zbirko podatkov ImageNet Large Scale Visual Recognition Challenge [20] (ILSVRC), ki je opisana kot novi izziv za detekcijo objektov na video posnetkih. Podatkovna zbirka video posnetkov vključuje skoraj 4.500 posnetkov, ki so porazdeljeni na učno in testno množico ter tvorijo skupaj več kot milijon označenih slik. Označeni objekti so klasificirani v 30 različnih razredov živali in vozil. V delu omenijo, da je uporaba takšnih množic zaradi njihove velikosti in različnih domen posnetkov, ki jih najdemo v testnih množicah za sledilnike, koristna za raziskave, ki se ukvarjajo s sledenjem. Posledično se jo lahko primerno uporablja za učenje CNN, brez da bi se preveč prilagodili testnim množicam za sledilnike.

3.2.1 Priprava učne množice

Učenje je potekalo na negativnih in pozitivnih parih z uporabo logistične kriterijske funkcije:

$$\ell(y, v) = \log(1 + exp(-yv)), \tag{3.3}$$

kjer je v realno število enega para in oznake $y \in \{ +1, -1\}$. Pozitivni par je primerjava dveh enakih objektov, negativni par pa je primerjava objekta z ozadjem. Ker je končni rezultat matrika rezultatov, se iz enega primera iskalne slike in slike predloge naredi več pozitivnih in negativnih primerov. Kriterijska funkcija celotne mape se nato izračuna kot povprečje vseh individualnih kriterijskih funkcij na mapi

$$L(\boldsymbol{y}, \boldsymbol{v}) = \frac{1}{|\boldsymbol{D}|} \sum_{u \in \boldsymbol{D}} \ell(\boldsymbol{y}[u], \boldsymbol{v}[u]). \tag{3.4}$$

Pari so pridobljeni z označenimi sekvencami slik, kjer izluščijo primere predloge in iskalne slike. Velikosti obeh označenih objektov se normalizira, pri tem pa se ne spremeni razmerja med višino in širino. Faktor skaliranja s je izbran tako, da je skalirani označeni okvir z višino h in širino w enak konstanti $A = 127^2$. Izračuna se ga kot

$$s(w + 2p) \times s(h + 2p) = A, \tag{3.5}$$

pri čemer p pomeni dodatno zajetje slike in je pri predlogi določena na polovico povprečne dimenzije $P = (w + h)/4$. V kolikor je slika manjša od določene, se jo zapolni s povprečno barvo te slike, kot je prikazano na Sliki 3.2.

V delu so za učenje uporabili celotno zbirko podatkov, kar je naneslo več kot 2 milijona označenih okvirjev. Matriko se nato napolni s pozitivnimi in negativnimi primeri. Vse lokacije, ki so znotraj radija R, pripadejo pozitivnemu razredu, ostali pa negativnemu in so določene po enačbi

$$\boldsymbol{y}[u] = \begin{cases} +1, & \text{če } k|u - c| \leq R \\ -1, & \text{drugače} \end{cases}. \tag{3.6}$$

Slika 3.2: Slika prikazuje primere zapolnitve predloge ali iskalne slike z njuno povprečno barvo. Rdeči pravokotnik predstavlja označeni okvir objekta. Slika povzeta po [6].

Izgube pozitivnih in negativnih primerov v matriki rezultatov se uteži, da se doseže enakost med številom primerov razreda. Za iskalno sliko uporabijo dimenzije 255×255, za sliko predloge pa 127×127. S takšnim razmerjem je nato izhodna matrika velikosti 17×17.

3.3 Sledenje

Sledilnik SiamFC za sledenje uporablja arhitekturo polno konvolucijske siamske CNN, ki smo jo opisali v Poglavju 3.1. Mreža sprejme iskalno sliko velikosti 255×255 in predlogo velikosti 127×127. Na začetku dobi sledilnik označeni okvir objekta, ki je lahko poljubno velik in ga je potrebno skalirati s faktorjem s, kot ga opisuje enačba (3.5). Enako kot predloga, se mora tudi iskalna slika skalirati s faktorjem s. V kolikor slika ne pokriva celotne predpisane velikosti, se ta prostor zapolni z njuno povprečno barvo. Ob sledenju se v naslednji sliki iz trenutno znane lokacije obreže iskalna slika, ki je štirikrat

večja od trenutne velikosti predloge. Predlogo in iskalno sliko se pošlje na vhoda polno konvolucijske siamske CNN, kjer se izračuna matrika $\boldsymbol{D}$, ki je omenjena v Poglavju 3.1. Vrednosti se pomnožijo s Hannovim oknom [21], tako da se bolj oddaljene lokacije od trenutno zaznane kaznuje z znižanjem vrednosti. Za prilagajanje velikosti pa se izračuna več iskalnih map različnih velikosti, kjer se tudi tukaj kaznujejo po skaliranju bolj oddaljene vrednosti. Po najdeni maksimalni uteženi vrednosti se določi trenutna zaznana lokacija in velikost objekta. V vsaki naslednji iteraciji sledilnik ponovi postopek in sledi, vse dokler ga ne izklopimo.

Pozitivna lastnost sledilnika je predvsem hitrost in učinkovitost, za kar je zaslužena uporaba polno konvolucijske siamske CNN. Ta v eni iteraciji poda vsa ujemanja različnih lokacij in skal na iskalni sliki. Težave, ki jih vidimo, pa so, da sledilnik kljub temu, da objekta ni prisotnega v iskalni sliki, vseeno izbere najprimernejšo lokacijo in velikost sledenega objekta. Tako se ob odpovedi sledenja lahko le po naključju postavi nazaj na sledeni objekt. Naslednja težava pa je, da se za predlogo vedno uporablja samo začetna slika objekta, tudi ko sledilnik že dlje časa sledi. Pri tem je objekt na sliki lahko že precej drugačen od začetne predloge in je s tem oteženo ujemanje na iskalni sliki. V naslednjem poglavju predlagamo naše izboljšave in izboljšamo sledilnik.

Poglavje 4

Implementacija našega sledilnika

Cilj magistrske naloge je bil izdelati nov dolgoročni sledilnik. Ta mora imeti sposobnost zaznati svojo lastno odpoved sledenja in se ob taki odpovedi ponastaviti na pravi objekt. Poleg dolgoročnega sledenja pa si želimo tudi hitro in natančno izvajanje sledilnika. Za izhodišče smo izbrali sledilnik SiamFC [6], ki smo ga opisali v Poglavju 3. V tem poglavju opišemo izdelavo našega dolgoročnega sledilnika. V prvem delu se osredotočimo na izdelavo modela detekcije odpovedi sledenja, ki zazna, ali sledilnik še sledi objektu. V drugem delu opišemo implementacijo detekcije objekta, ki ob primeru zaznane odpovedi sledenja ponastavi sledilnik na objekt, ki mu sledimo. V zadnjem delu poglavja pa opišemo morebitne izboljšave sledenja s posodabljanjem predloge.

4.1 Detekcija odpovedi sledenja

V našem delu predlagamo implementacijo modela, ki bo zaznal, ali sledilnik še sledi objektu. Odločili smo se za uporabo CNN. V tem razdelku sprva opišemo predlagano arhitekturo CNN, nato pa opišemo, na kakšen način smo pridobili učno množico s primeri pravilnega in nepravilnega sledenja.

Na koncu še opišemo postopek učenja omenjene CNN.

4.1.1 Arhitektura CNN za detekcijo odpovedi

Arhitektura je prikazana na Sliki 4.1 in je povzeta v Tabeli 4.1. Za to arhitekturo je bilo potrebno razviti lastni računski blok, ki je na koncu iz polno povezanega konvolucijskega bloka izračunal maksimalno in povprečno vrednost. Ti dve vrednosti pa se nato uporabita pri končni napovedi. Omenjeni računski blok smo poimenovali MeanMax in ga bomo podrobneje opisali v naslednjem razdelku.

Računski blok MeanMax

Vrednosti polno povezanega bloka so shranjene v kanalih map, ki so velikosti 1×1. Pri tem metode, kot so maksimalno in povprečno združevanje, ne pridejo v poštev, saj vrednosti računajo po mapi in ne po kanalih. V nalogi smo zato razvili svoj računski blok, ki je vrednosti izračunal po kanalih in podal naprej maksimalno in povprečno vrednost.

Računski blok deluje tako, da dobi na vhod vrednosti polno povezanega konvolucijskega bloka, ki je velikosti $H \times W \times D$. Pri tem so vrednosti $W \times H = 1 \times 1$. Povprečno vrednost pa se izračuna po enačbi

$$\boldsymbol{y}_{mean} = \frac{1}{D} \sum_{d=1}^{D} \boldsymbol{x}_{1,1,d}, \tag{4.1}$$

maksimalno pa po enačbi

$$\boldsymbol{y}_{max} = max\{\ \boldsymbol{x}\}. \tag{4.2}$$

Končna velikost izhodne mape je nato $\boldsymbol{y} \in \mathbb{R}^{2\times1\times1}$. Izhoda nato povežemo na zadnji konvolucijski blok, ki sprejme ti dve vrednosti in na izhod vrne verjetnosti napovedi za posamezni razred.

Pri vzvratnem prehodu pa dobimo iz zadnjega konvolucijskega bloka na funkciji dva odvoda d_{mean} d_{max}. Ta predstavljata popravek napake pri vzvratnem prehodu, ki jih je izračunal zadnji konvolucijski blok. Odvoda nato

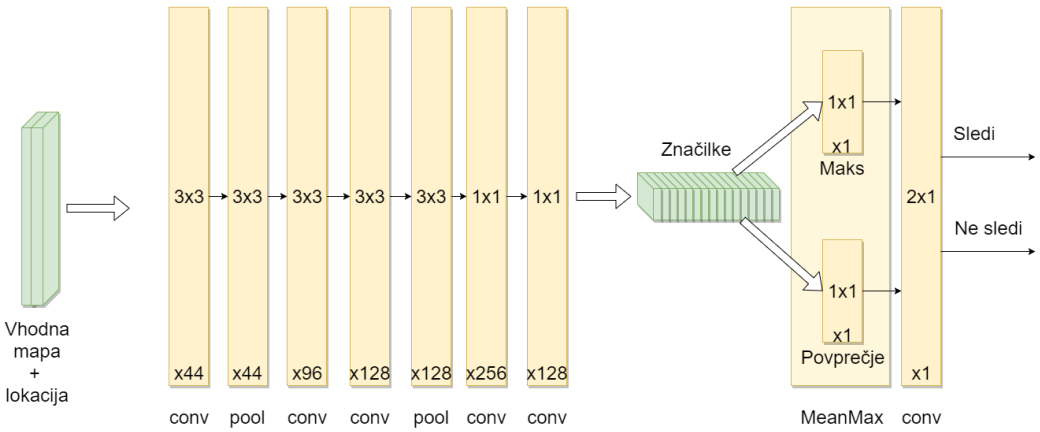
Računski blok	Velikost filtra	Stride	Velikost mape	Št. kanalov
			17×17	$\times 2$
Conv1	3×3	1	15×15	$\times 44$
pool1	3×3	2	7×7	$\times 44$
Conv2	3×3	1	5×5	$\times 96$
Conv3	3×3	1	3×3	$\times 128$
pool3	3×3	1	1×1	$\times 128$
Conv4	1×1	1	1×1	$\times 256$
Conv5	1×1	1	1×1	$\times 128$
MeanMax		1	2	$\times 2$

Tabela 4.1: Tabela prikazuje arhitekturo CNN za napoved odpovedi sledenja.

pomnožimo z vrednostmi na vhodu bloka MeanMax. d_{mean} pomnožimo z vsemi vrednostmi na vhodu, d_{max} pa samo z največjo vrednostjo na vhodu. Tako prenesemo napako nazaj na naslednji konvolucijski blok, kjer se njihove uteži ustrezno popravijo in zmanjšajo napako z vsako iteracijo.

4.1.2 Učna množica

Učno množico smo pridobili s shranjevanjem izhoda polno konvolucijske siamske CNN sledilnika SiamFC [6], pri sledenju posnetkov iz zbirke podatkov ILSVRC. Učni primer prikazuje Slika 4.2. Prva težava pri pridobivanju učnih podatkov je ta, da se sledilnik poskuša čim bolj prilegati najboljšemu trenutnemu ujemanju. Pri odpovedi sledenja začne sledilnik slediti ozadju. Ker sledilnik omogoča velikosti zajetega objekta, se lahko ta tako spremeni, da sledeno ozadje doseže dobro ujemanje s predlogo. Druga težava je nasprotna prvi – sledilnik slučajno pravilno sledi objektu, vendar je objekt prekrit ali kako drugače deformiran, da se ga ne da nikakor prepoznati. Tretja težava je ta, da je na sliki lahko prisotnih več enakih objektov in da sledilnik CNN ne zazna razlik med tovrstnimi objekti. Tovrstni primeri škodijo učenju, saj



Slika 4.1: Slika prikazuje arhitekturo CNN klasifikatorja odpovedi sledenja.

pri prvi in tretji težavi dobremu odzivu pripišemo negativno oznako, medtem ko pri drugi slabemu odzivu pripišemo pozitivno oznako. Z označenimi okvirji na slikah lahko zanesljivo izberemo dobre pozitivne primere sledenja, medtem ko lahko za negativne primere samo omilimo njihovo število slabih primerov. Izmed množice primerov smo zato za učenje uporabili le tiste, ki so ustrezali našim kriterijem. Za pozitivno oznako je moral imeti napovedan okvir vsaj 70% pravilne velikosti, središčna lokacija pa je morala biti znotraj pravega okvirja. Za negativne primere je moral biti pravokotnik prav tako vsaj 70% pravilne velikosti, napovedani okvir pa se ni smel prekrivati z označenim okvirjem. S tem smo omilili prvo in drugo težavo, medtem ko tretje težave ni mogoče odpraviti s tem načinom. Odpravili bi jo lahko tako, da bi ročno odstranili takšne primere iz učne množice, a nam obsežnost učne množice tega ne omogoča. S tem postopkom smo pridobili 168.528 primerov, med katerimi se prvi polovici sledi pravilno, drugi pa nepravilno.

4.1.3 Učenje

Učna množica je razdeljena na dva razreda: pravilno in nepravilno sledenje. Od modela želimo, da vrne verjetnost za vsak posamezni razred. Izmed vseh primerov, torej 168.528, smo za učenje uporabili 141.816 primerov, za validacijo pa 26.712. Pri obema množicama je en razred predstavljal njuno polovico. Učno množico uporabimo za učenje modela, z validacijsko pa testiramo njegovo uspešnost. Pri razvoju CNN smo preizkušali različne načine in nato izbrali tistega, ki je imel najboljši rezultat na validacijski množici. Za ocenjevanje posameznih modelov smo uporabili kriterije, ki so ocenili, kako dobro deluje naučeni model. Napovedi razredov smo razdelili na negativne in pozitivne. Slednja pomeni, da sledilnik še sledi objektu, negativna napoved pa pomeni, da sledilnik ne sledi objektu. Tako smo lahko uporabili klasifikacijsko napako (CE, angl. classification error), delež napačno zaznanih kot negativnih (FPR, angl. false positive rate) in delež napačno zaznanih kot pozitivnih (FPR, angl. false positive rate). Te napake se izračuna po omenjenem zaporedju:

$$\ell(x, c) = 1\Big[c \neq \operatorname{argmax} x\Big], \tag{4.3}$$

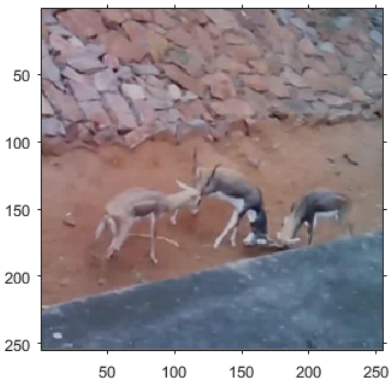
$$\ell(x, c) = 1\Big[c = \operatorname{argmax} x \wedge \text{sledi} \neq L(\operatorname{argmax} x)\Big], \tag{4.4}$$

$$\ell(x, c) = 1\Big[c = \operatorname{argmax} x \wedge \text{ne sledi} \neq L(\operatorname{argmax} x)\Big], \tag{4.5}$$

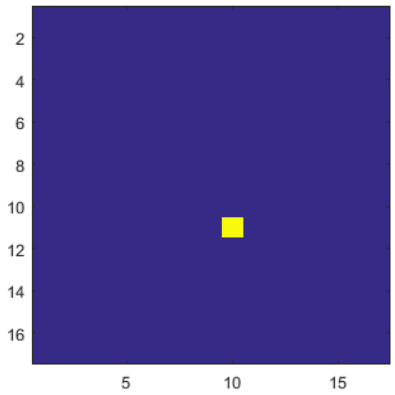
kjer je x napovedan razred in c pravilen razred. Pri tem je pomemben delež napačno zaznanih kot negativnih, saj v tem primeru napačno ocenimo, da sledilnik več ne sledi objektu in nehote sprožimo ukrepe za ponovno detekcijo objekta.

Odkrili smo, da arhitektura, ki je podana v Tabeli 4.1, dosega najboljše rezultate na validacijski množici. Rezultate učne in testne množice prikazuje Slika 4.3. Pri učenju smo pri validacijski množici dosegli slabše rezultate od učne, vendar se je napaka sorazmerno zmanjševala z napako učne množice pri vsaki iteraciji. Tako vemo, da se nismo preveč prilagodili učni množici.

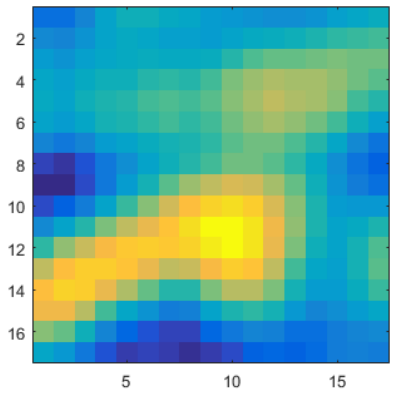
(a) Predloga



(b) Iskalna slika

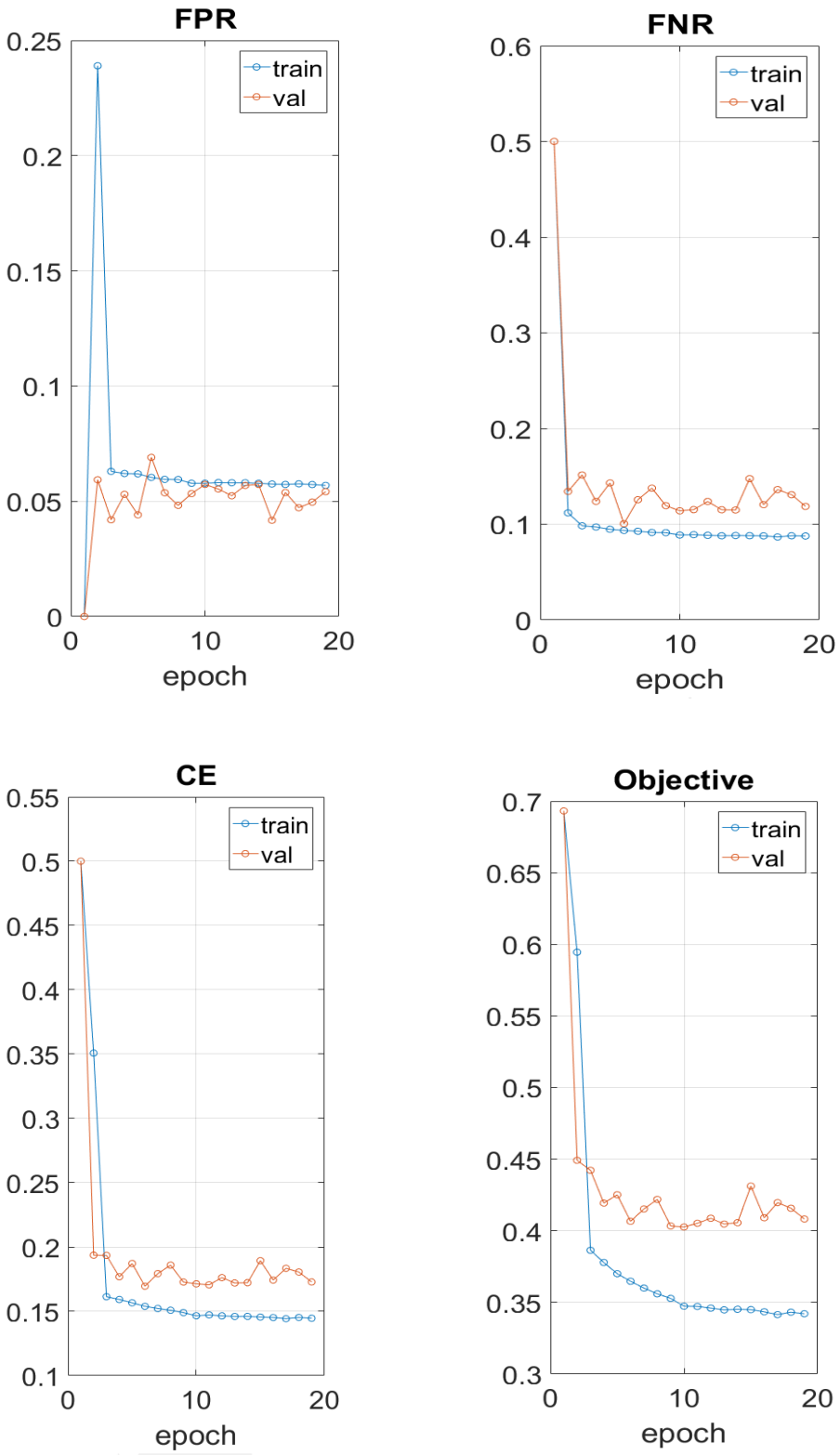


(c) Napovedana lokacija



(d) Odzivna mapa

Slika 4.2: Slika prikazuje vhod modela CNN sledilnika SiamFC (a) in (b) ter njen izhod (c) in (d), ki je sestavljena matrika iz odzivne mape in njene napovedane lokacije objekta.



Slika 4.3: Slika prikazuje rezultate učne (train) in validacijske (val) množice naše arhitekture CNN.

4.2 Detekcija objekta

Z implementacijo klasifikatorja odpovedi sledenja smo lahko razvili model detekcije objekta, ki v primeru odpovedi sledilnika ponovno poišče iskani objekt. Mnogi sledilniki za detekcijo objekta uporabljajo vzorčenje, vendar je tak pristop počasen, saj zahteva veliko računskih operacij, kot je izrez slike, pomik na grafično kartico ali procesor itd. Če želimo ohraniti hitrost izvajanja, moramo uporabiti drug pristop, ki bo hitro detektiral objekt na sliki. Pri tem sledilnik SiamFC [6] že deluje kot tak detektor, vendar ne na celotni sliki, saj izbere samo del slike za iskalno sliko, glede na prejšnjo zaznano lokacijo objekta. V našem delu smo spremenili že naučeno CNN iz sledilnika SiamFC in jo spremenili v detektor, ki poišče objekt na celotni sliki. Tak pristop je računsko učinkovit, saj se slika, predstavljena kot matrika, v celoti prenese na grafično kartico, kjer se optimizirano poračuna z ostalimi matričnimi filtri. S tem se izognemo večkratnemu izrezovanju slike in pošiljanju na grafično kartico, ki je ozko grlo prenosa podatkov. Tako smo ohranili hitrost sledilnika in ga obenem nagradili za dolgoročno sledenje. V naslednjem podpoglavju opišemo implementacijo detekcije objekta.

4.2.1 Uporaba SiamFC kot detektorja

Dobra lastnost polno konvolucijske siamske CNN je ta, da lahko uporabimo poljubno veliko iskalno sliko, ki pa mora biti vsaj toliko velika kot slika predloge. Sledilnik SiamFC uporablja iskalno sliko velikosti 255×255, ki pa je taka zgolj zaradi učinkovitosti sledenja in ni razloga, da ne bi bila večja. V našem delu smo uporabili model sledilnika SiamFC [6] in povečali iskalno sliko na celotno sliko. V Poglavju 3 smo omenili, da je potrebno sliko predloge in iskalno sliko skalirati s faktorjem s, kot ga opisuje enačba (3.5) ter zapolniti morebitni manjkajoči prostor z njuno povprečno barvo. Slednjega koraka pri iskalni sliki ni potrebno narediti, saj je slika lahko poljubne velikosti, paziti moramo le, da je skalirana s faktorjem s.

Po izračunani križni korelaciji smo dobili končno veliko matriko mape,

kjer vsaka vrednost pomeni ujemanje te lokacije s sliko predloge. Koordinate največje vrednosti v polju pomenijo lokacijo detektiranega objekta, ki pa jih je potrebno še preslikati v koordinate slike. Če imamo iskalno sliko velikosti $w \times h$, mapo rezultatov velikosti $w' \times h'$ in zaznane koordinate $[x', y']$, se prave koordinate $[x, y]$ izračuna po naslednjih enačbah:

$$x = \frac{x' \frac{w-127+1}{w'} + \frac{127}{2}}{s}, \tag{4.6}$$

$$y = \frac{y' \frac{h-127+1}{h'} + \frac{127}{2}}{s}. \tag{4.7}$$

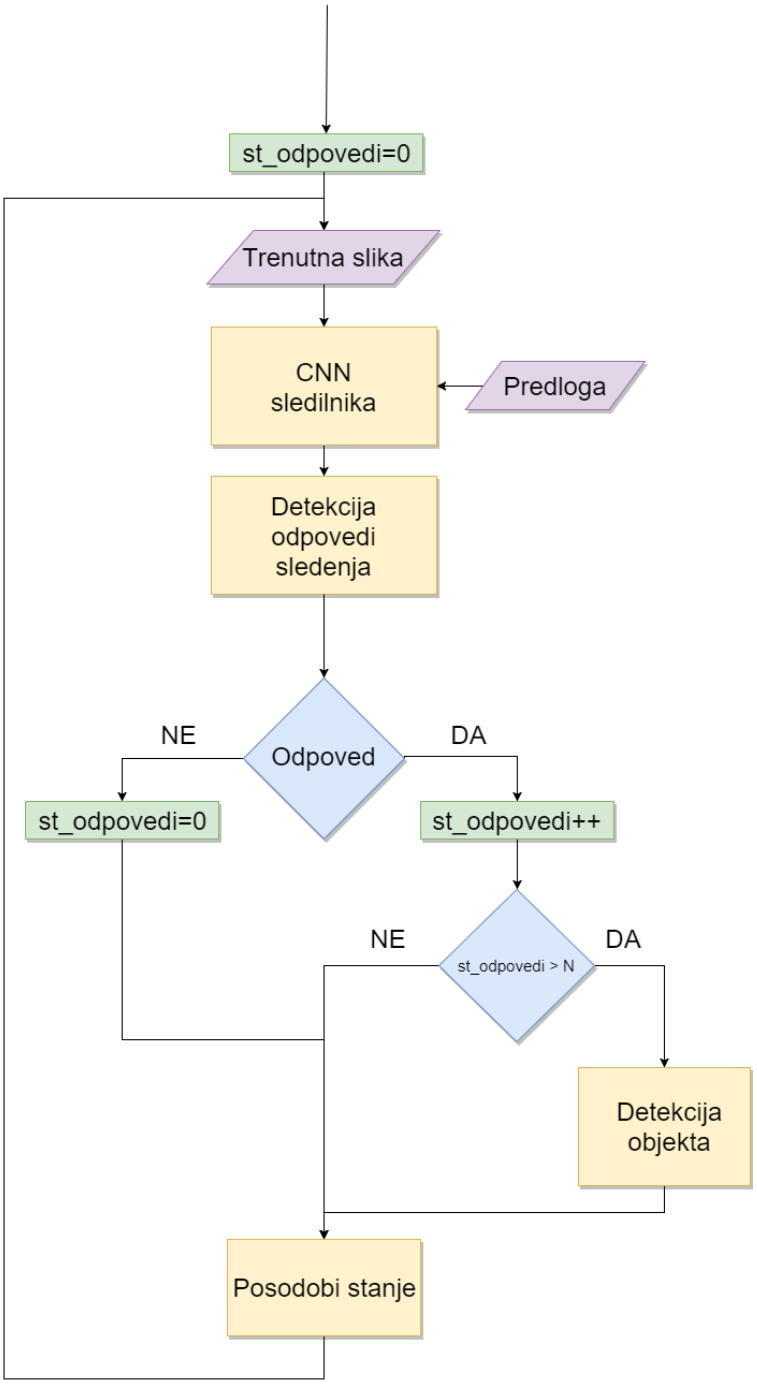
4.2.2 Sledilnik z detekcijo objekta

Z združitvijo implementacije detekcije odpovedi sledenja in detekcije objekta smo izdelali novi dolgoročni sledilnik, ki smo ga poimenovali Hitri sledilnik z detekcijo odpovedi (HSDO). Diagram poteka tega sledilnika prikazuje Slika 4.4. Sledilnik deluje tako, da se ob detekciji odpovedi sledenja sproži detekcija objekta, ki ponastavi sledilnik nazaj na pravi objekt. Pri tem želimo, da se ob pravilnem sledenju detekcija objekta čim manjkrat sproži, saj bi tako lahko iz pravilnega objekta prešel napovedni okvir na neki drug objekt. Zato želimo, da se detekcija objekta sproži samo takrat, ko klasifikator odpovedi sledenja z veliko verjetnostjo napove odpoved sledenja. Pri tem pa se poveča verjetnost za napako, da se detekcija objekta ne bo sprožila ob nepravilnem sledenju. Nastavitev ustreznega praga je zato ključnega pomena, saj lahko sicer poslabšamo sledenje. Da bi zmanjšali posledice napak klasifikatorja odpovedi sledenja in s tem preskoke iz pravilnega objekta na drug objekt, smo uporabili dva pristopa.

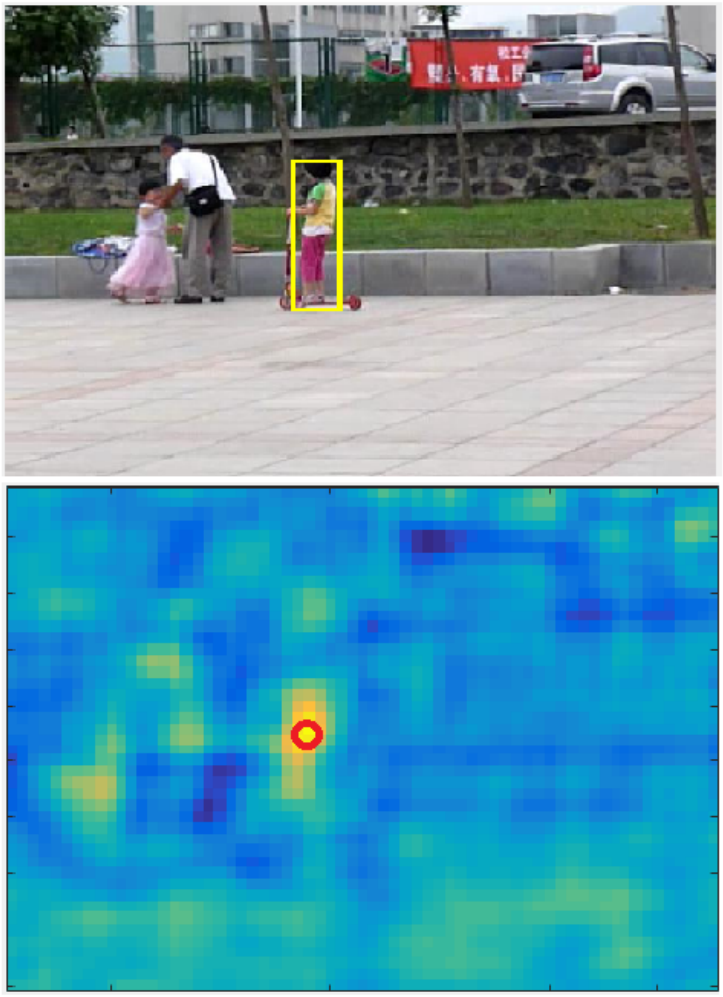
V prvem pristopu smo napovedane verjetnosti odpovedi sledenja utežili s funkcijo gostote, ki bolj oddaljenim točkam od trenutno znane lokacije objekta, zmanjša njune vrednosti. Funkcija gostote $f(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ izračuna gostoto normalne porazdelitve za vsako polje matrike $\boldsymbol{x}$ s predpisanim povprečjem $\boldsymbol{\mu}$ in standardnim odklonom $\boldsymbol{\sigma}$. Parametre funkcije smo nastavili

tako, da smo za povprečno vrednost $\boldsymbol{\mu}$ določili zadnjo znano lokacijo objekta. Standardni odklon $\boldsymbol{\sigma}$ smo določili kot kvadratno velikost maksimalne dolžine slike $max(w, h)^2$. Velikost matrike $\boldsymbol{x}$ pa je bila enaka vhodni iskalni sliki modela CNN. Izračunano funkcijo gostote normaliziramo na vrednosti intervala $[0, 1]$ in skaliramo na velikost izhodne matrike modela CNN. Prav tako normaliziramo tudi izhodno matriko na vrednosti intervala $[0, 1]$, nato pa pomnožimo njene istoležeče elemente z izračunano funkcijo gostote. Primer delovanja detektorja prikazuje Slika 4.5. Na njej vidimo sliko sledenega objekta in rezultat detekcije objekta, ko pravilno poišče sledeni objekt.

V drugem pristopu pa smo določili minimalno zaporedno število napovedanih odpovedi sledenja, preden se detekcija objekta sproži in poišče objekt. Pri sledenju objekta pogosto prihaja do nenadnih, a ne trajnih sprememb na sliki ali objektu, ki deformirajo objekt na sliki. Take spremembe so na primer: sprememba osvetlitve, zakrivanje objekta, zameglitev zaradi premika (angl. motion blur) itd. Z določitvijo potrebnega minimalnega števila napovedanih odpovedi sledenja omilimo vpliv teh kratkotrajnih motilcev, saj minejo še preden je zaznanih dovolj zaporednih odpovedi sledenja. Pomembno je, da ne določimo prevelikega minimalnega števila odpovedi, saj s tem vplivamo na pravilno sprožitev detekcije objekta.



Slika 4.4: Slika prikazuje diagram poteka našega dolgoročnega sledilnika HSDO.



Slika 4.5: Zgornja slika prikazuje sledenje objektu z označeno lokacijo. Spodnja slika pa prikazuje mapo rezultatov detekcije objekta in detektirano maksimalno vrednost. Na sliki je razvidno, da je detektor pravilno zaznal iskani objekt.

4.3 Posodabljanje predloge

Za boljšo dolgoročno sledenje smo naredili eksperiment s posodabljanjem predloge med sledenjem objekta. To deluje tako, da si sledilnik med sledenjem shranjuje slike sledenega objekta in te uporabi za nadaljnje sledenje. Posodabljanje predloge omogoči sledilniku boljšo odpornost na spremembe sledečega objekta skozi čas. V eksperimentu smo poskusili dva pristopa posodabljanja predloge, ki pa se ju lahko poleg sledilnika HSDO uporablja tudi pri sledilniku SiamFC [6].

V prvem pristopu smo predlogo posodabljali po naslednjem postopku. Na začetku dobi sledilnik predlogo, ki je ročno označena in velja za pravilno predstavitev objekta. Tekom sledenja si s klasifikatorjem odpovedi sledenja beležimo napovedane odpovedi sledenja. Če med časom določenih zaporednih napovedi ni prišlo do napovedane odpovedi sledenja, predpostavimo, da sledilnik še vedno sledi objektu in je ta tudi viden. Takrat vzamemo sliko sledenega objekta in ustvarimo novo predlogo, ki bo skupaj s starimi shranjenimi predlogami služila za nadaljnje sledenje. Ko zapolnimo predpisano število shranjenih predlog, odstranimo tisto, ki je najdlje časa v vrsti. Pri nadaljnjem sledenju za končno odzivno mapo izračunamo povprečje odzivnih map vseh shranjenih predlog.

V drugem pristopu smo predlogo poskusili posodabljati tako, da pri napovedanem pravilnem sledenju posodobimo predlogo T s trenutno predlogo T' po enačbi $T = T\alpha + T'(1 - \alpha)$, kjer je α parameter, ki določi delež vpliva prvotne predloge z novo. Takšno posodabljanje je po navadi uspešno, saj ima ena predloga majhen vpliv na celotno predlogo T in tako ena slaba vrednost ne pokvari sledenja.

Sledilniku, ki uporablja prvi pristop, pripnemo oznako +Pv, kot na primer HSDO+Pv. Za drugi pristop pa sledilniku pripnemo oznako +Pt.

Poglavje 5

Eksperimentalna evalvacija

V Poglavju Eksperimentalna evalvacija testiramo naše implementacije na različnih poskusih. V prvem delu testiramo detekcijo odpovedi sledenja, ki smo jo predstavili v Poglavju 4.1. V drugem delu testiramo predlagani različici posodabljanja predloge, ki smo ju omenili v Poglavju 4.3. V tretjem delu testiramo našo različico dolgoročnega sledilnika HSDO, ki smo ga omenili v Poglavju 4.2. Na koncu pa testiramo izgubo hitrosti z dodajanjem posameznih izboljšav, ki smo jih opisali v Poglavju 4.

5.1 Testiranje detekcije odpovedi sledenja

Namen testiranja detekcije odpovedi sledenja je analiza zanesljivosti klasifikatorja. Prav tako nas tudi zanima, katere vrednosti morajo imeti parametri za dano nalogo, kot je na primer za sprožitev detekcije objekta. Detekcijo odpovedi smo testirali tako, da smo pognali sledilnik na podatkovni bazi VOT2016 [22], ki vsebuje označene okvirje sledečega objekta. Testna množica vsebuje 60 različnih sekvenc, ki so bile izbrane na podlagi njihove raznovrstnosti. Pri vsakem označenem okvirju so poleg lokacije in velikosti okvirja zabeleženi tudi atributi, ki predstavljajo:

- zakrivanje;
- spremembe osvetlitve;

- premikanje objekta;
- sprememba velikosti objekta;
- premikanje kamere;
- drugo.

S temi atributi lahko bolj podrobno analiziramo sledilnik, tako da izvemo, ob katerih spremembah je sledilnik/model najbolj občutljiv in kje dobro deluje. Pri sledenju smo uporabili samo obstoječi sledilnik SiamFC, brez drugih metod in izboljšav, ter zapisali poleg napovedanega okvirja tudi napoved našega klasifikatorja. Pri testiranju nas je zanimalo, kakšna je zanesljivost napovedi, če sledilnik sledi objektu in kakšna je zanesljivost, če sledilnik sledi napačnemu objektu. Prag, da sledilnik še sledi objektu, smo določili tako, da mora biti prekrivanje napovedanega okvirja z dejanskim vsaj 50%. Na podlagi tega smo lahko določili vrednosti pravilno pozitivnih (TP, angl. true positive), pravilno negativnih (TN, angl. true negative), nepravilno pozitivnih (FP, angl. false positive) in nepravilno negativnih (FN, angl. false negative), v odvisnosti od praga napovedane verjetnosti klasifikatorja. Detekcijo odpovedi sledenja uporabljamo za dve nalogi – posodabljanje predloge in detekcijo objekta.

Če želimo detekcijo odpovedi uporabljati za posodabljanje predloge, nam je pomembno, da model čim manjkrat napačno napove vrednost, da sledilnik še sledi objektu. Imeti želimo torej čim manjše število zabeleženih FP ob sprejemljivi vrednosti TP. Ta vrednost se imenuje delež napačno odkritih (FDR, angl. false discovery rate) in se izračuna kot

$$FDR = \frac{FP}{TP + FP}. \tag{5.1}$$

V primeru ponovne detekcije objekta po odpovedi sledenja si želimo, da bi imel model čim manjše število napačno napovedanih odpovedi sledenja. V tem primeru želimo imeti čim manjše število FN ob sprejemljivem številu

TN. Ta vrednost se imenuje delež napačno omitiranih (FOR, angl. false ommision rate) in se izračuna kot

$$FOR = \frac{FN}{TN + FN}. \tag{5.2}$$

Ti dve meri uporabimo samo v tem podpoglavju za analizo detekcije odpovedi.

5.1.1 Rezultati

Rezultati za uporabo detekcije odpovedi sledenja so prikazani na Sliki 5.2 in povzeti v Tabeli 5.1. Če želimo, da se detektor objekta sproži ob vsaj 90% pravilnih odpovedi sledenja, lahko pričakujemo, da bo klasifikator odpovedi sledenja napačno napovedal pravilno sledenje pri več kot 45% primerov. Za posodabljanje predloge si želimo imeti čim manjši delež obeh napak FDR in FOR in ta prag je pri 75% napovedi, da sledilnik pravilno sledi.

Ocenili smo tudi napaki za posamezni atribut, ki jih prikazuje graf na Sliki 5.3 in povzema Tabela 5.2. Za napako FOR smo uporabili prag 75% napovedane vrednosti klasifikatorja, da pravilno sledi objektu, saj nas ta zanima za posodabljanje predloge. Rezultati so pokazali, da je največ napak pri gibanju objekta. Za napako FDR smo uporabili prag 10% napovedi klasifikatorja, da sledilnik pravilno sledi, kar pomeni, da je klasifikator z napovedjo 90% napovedal, da je sledilnik odpovedal. Rezultati so pokazali, da je pri vseh atributih podobna napaka. Atribut za zakrivanje objekta smo izključili iz analize, saj ne moremo vedeti, kdaj je objekt prekrit v celoti ali pa samo delno ter tako ne moremo oceniti pravilnosti klasifikatorja.

Z opazovanjem detekcije odpovedi sledenja smo odkrili, da klasifikator dobro napove izgubo sledenja, ko sledeči objekt nenadoma izgine iz vidnega polja (na primer zaradi prekrivanja), in slabo, kadar sledilnik sledi napačnemu objektu, ki je podoben sledenemu. Za omenjeno napako krivimo model CNN sledilnika, ki ne more tako podrobno ločiti objektov. Primere detekcije odpovedi sledenja prikazuje Slika 5.1. Na njej vidimo, da model dobro prepozna

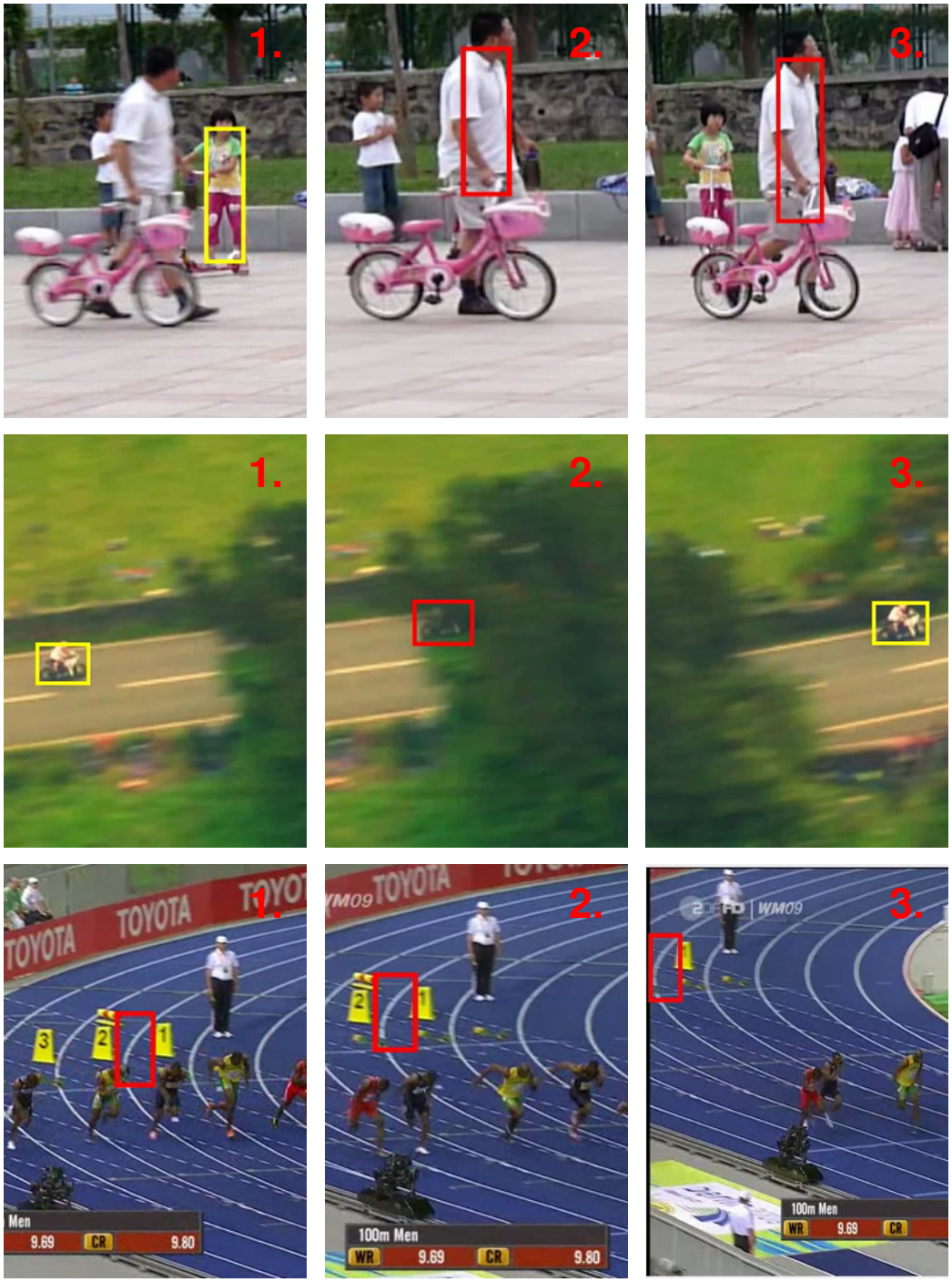
Namen	**FOR**	**FDR**	**Prag**
detekcija odpovedi	0.10	0.46	10%
posodabljanje predloge	0.26	0.26	75%

Tabela 5.1: Tabela prikazuje napaki FOR in FDR pri določenih pragih napovedi klasifikatorja, da sledilnik uspešno sledi.

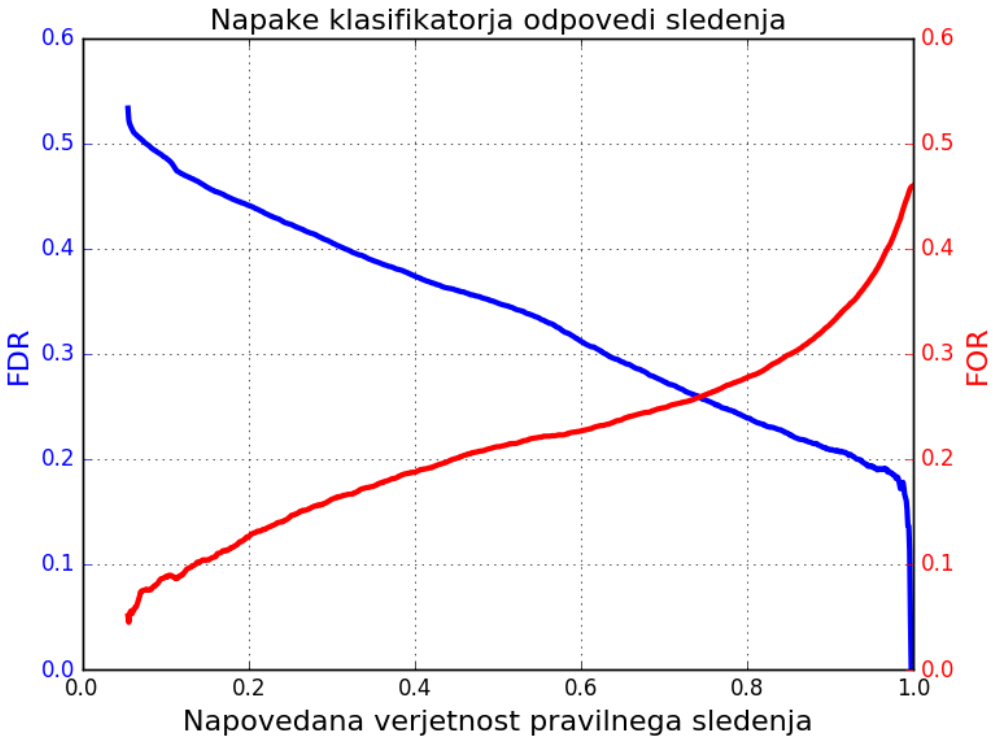
Atribut	**FOR(75% prag)**	**FDR(10% prag)**
Sprememba svetlosti	0,27	0,10
Gibanje	0,55	0,11
Premikanje kamere	0,31	0,04
Sprememba svetlosti	0,25	0,06

Tabela 5.2: Tabela prikazuje napaki FOR in FDR pri različnih atributih.

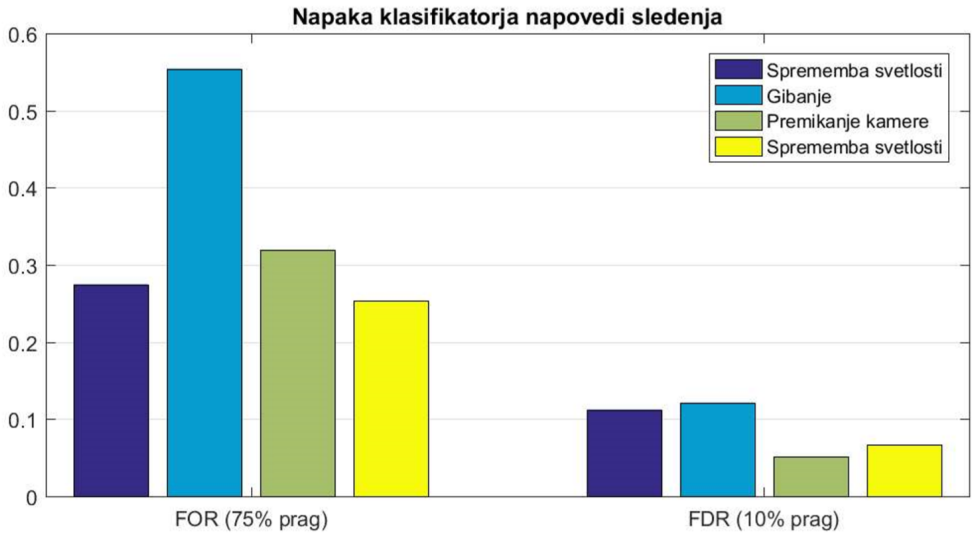
različne načine odpovedi, kot je na primer prekritje objekta in odpoved sledenja.



Slika 5.1: Slika prikazuje primere napovedi odpovedi sledenja (rdeča barva). Primer prekritja objekta prikazujeta prvi dve vrstici, primer odpovedi sledenja pa prikazuje zadnja vrstica.



Slika 5.2: Slika prikazuje graf napaki FDR in FOR v odvisnosti od praga napovedi klasifikatorja, da sledilnik uspešno sledi.



Slika 5.3: Slika prikazuje graf napak FDR in FOR pri različnih atributih.

5.2 Rezultati posodabljanja predloge

V tem podpoglavju testiramo naša predloga posodabljanja predloge, ki smo ju omenili v Poglavju 4.3. Posodabljanje predloge testiramo samo na sledilniku SiamFC [6], naš sledilnik HSDO pa izpustimo, saj ne želimo vmešavati detekcije objekta v rezultate. Testiramo torej referenčni sledilnik SiamFC, ki služi za primerjavo rezultatov, SiamFC+Pv in SiamFC+Pt. Pri sledilnikoma SiamFC+Pv in SiamFC+Pt poženemo eksperiment dvakrat. Prvič, ko ima detekcija odpovedi sledenja vpliv, in drugič, ko ga nima, torej je prag klasifikatorja odpovedi sledenja nastavljen na vrednost 0. Zanima nas namreč tudi, kakšen vpliv ima detekcija odpovedi sledenja na posodabljanje predloge.

Najprej smo morali pri posameznem omenjenem predlogu posodabljanja predloge nastaviti ustrezne parametre za optimalno posodabljanje predloge. Za nastavitev praga napovedi klasifikatorja odpovedi sledenja smo v Poglavju 5.1 že ugotovili, da je primeren prag 75% zanesljivosti, da sledilnik pravilno sledi. Torej vsakič, ko klasifikator z vsaj 75% zanesljivostjo oceni sledenje kot pravilno, je napoved klasificirana kot pravilno sledenje.

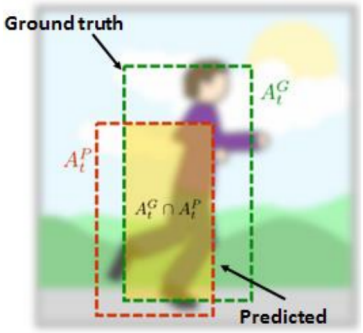
Pri predlogu +Pv smo v preliminarni evalvaciji ugotovili, da se sledenje izboljša že, če je shranjena samo ena dodatna predloga in se s številom dodatnih shranjenih predlog sledenje dodatno izboljšuje. Preveliko število shranjenih predlog pa začne poslabševati rezultat, saj se shrani preveč slabih primerov predlog, ki ne zajemajo celotne informacije o predlogi. Podobno lahko zaznamo tudi pri zaporednem številu uspešnih sledenj. Če jih nastavimo preveč, se predloga ne shrani dovolj pogosto in se s tem poslabša prilagodljivost na spremembe objekta. V kolikor jih je premalo, pa se predloga nezanesljivo shrani. Potrebno je ugotoviti pravo razmerje med zaporednim uspešnim sledenjem in številom shranjenih predlog. Ugotovili smo, da je najboljše razmerje 5 zaporednih napovedi in 5 shranjenih predlog.

Pri predlogu +Pt pa smo v preliminarni evalvaciji ugotovili, da je za stabilno sledenje potrebno imeti shranjeno prvotno predlogo, ki se jo upošteva skupaj s posodobljeno predlogo. Iz obeh predlog se izračuna povprečje, kjer ima slednja 80% vpliv, prvotna predloga pa 20% vpliv. Parameter α smo

nastavili na število $0,05$.

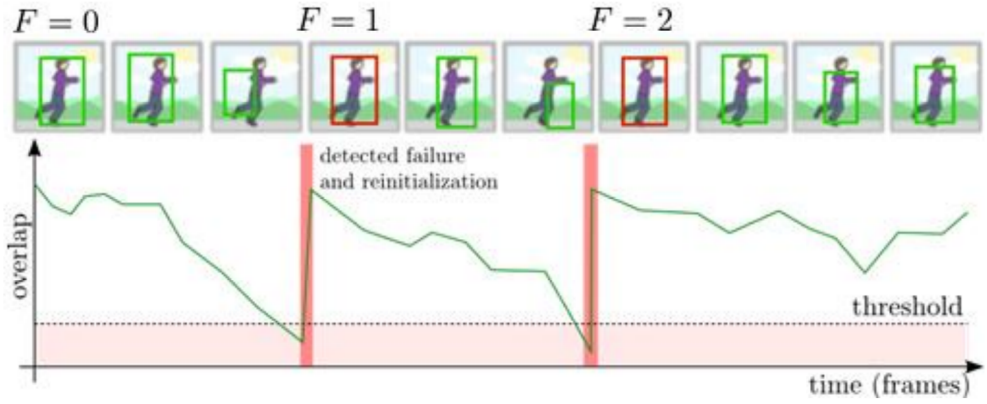
5.2.1 Testiranje na podatkovni zbirki VOT

Posodabljanje predloge smo testirali z izbranimi sekvencami slik podatkovne zbirke VOT2016 [22]. Pri ocenjevanju sledilnika s posodabljanjem predloge smo izbrali glavni eksperiment VOT, ki ob primeru izgube objekta ponovno ponastavi sledilnik. Takšen pristop je primeren za kratkoročne sledilnike, saj se ti ne morejo sami ponastaviti. Meri se natančnost in robustnost. Natančnost nam pove, kolikšen je povprečen delež prekrivanja med označenim okvirjem objekta in napovedanim okvirjem, kot prikazuje Slika 5.4.



Slika 5.4: Slika prikazuje primer prekrivanja označenega okvirja (angl. groundtruth) z napovedanim okvirjem (angl. predicted). Slika povzeta po [23].

Robustnost nam pove napako, kolikokrat je sledilnik odpovedal in ga je bilo potrebno ponovno ponastaviti. Odpoved se šteje takrat, ko je trenutno prekrivanje pod določenim pragom, kot prikazuje Slika 5.5. V našem primeru se šteje za odpoved sledenja, ko je delež prekrivanja enak 0 oziroma takrat, ko se okvirja ne prekrivata.



Slika 5.5: Slika prikazuje primer odpovedi sledenja, ko je prekrivanje označenega okvirja z napovedanim okvirjem enako 0. Slika povzeta po [23].

Rezultati glavnega eksperimenta VOT2016

Rezultati glavnega eksperimenta VOT2016 [22] so povzeti v tabeli 5.3. Ugotovili smo, da oba predloga izboljšata rezultate na glavnem eksperimentu VOT2016, pri čemer se predlog +Pv odreže bolje v robustnosti, +Pt pa v natančnosti. Detekcija odpovedi sledenja pa je imela le manjši vpliv na boljše rezultate.

Pri SiamFC+Pv z detekcijo odpovedi sledenja SiamFC+Pv(det) smo zabeležili 18 manj odpovedi sledenja, torej iz 104 na 86. Povprečje odpovedi sledenj se je zmanjšalo za 6 iz 25,50 na 19,33, uteženo povprečje pa za 8 iz 30,03 na 22,41. Prekrivanje se je pri tem zmanjšalo za približno 5%, iz 0,52 na 0,47. Brez vpliva detekcije odpovedi sledenja smo imeli le malenkost slabše rezultate. Zabeležili smo povprečno 1 odpoved več in za 1% manjšo natančnost kot z uporabo detekcije odpovedi sledenja. Skupni rezultat robustnosti in natančnosti pove mera pričakovano prekrivanje, ki pa se je tudi izboljšala iz 23,5% na 24,7%. Robustnost se je izboljšala za 21%, natančnost pa se je poslabšala za 10%.

Pri SiamFC+Pt z detekcijo odpovedi sledenja SiamFC+Pt(det) smo zabeležili 13 manj odpovedi sledenja, torej iz 104 na 91. Povprečje odpovedi sledenj se je zmanjšalo za 5 iz 25,50 na 21,00, uteženo povprečje pa za 6 iz

30,03 na 24,44. Prekrivanje se je pri tem zmanjšalo za približno 1% iz 0,52 na 0,51. Brez vpliva detekcije odpovedi sledenja pa smo imeli tudi tukaj le malenkost slabše rezultate. Zabeležili smo povprečno 1 odpoved več in za 1% manjšo natančnost kot z uporabo detekcije odpovedi sledenja. Zaradi dobre natančnosti pa smo tukaj dobili najboljši rezultat pričakovanega prekrivanja, ki je bil 0,2471. Robustnost se je izboljšala za 14%, natančnost pa se je poslabšala za 2%.

Kljub temu, da uporaba detekcije odpovedi sledenja ni preveč vplivala na rezultate posodabljanja predloge, smo vseeno odkrili primere, ki potrjujejo njeno koristnost. Tak primer prikazuje Slika 5.6. Na sliki vidimo dva plesalca, kjer sledilnik sledi plesalki in se vrtita tako, da plesalec občasno prekrije plesalko. Sledilnik brez klasifikatorja odpovedi sledenja bi shranil predlogo, ko je plesalec pred plesalko. S tem bi se zmanjšala raznolikost med sledeno plesalko in plesalcem na predlogi. Z uporabo detekcije odpovedi sledenja pa preprečimo takšna posodabljanja tako, da predlogo posodobimo samo takrat, ko vemo, da je sledeni objekt viden na sliki in tudi pravilno sleden.

Slika 5.6: Slika prikazuje sledenje z uporabo detekcije odpovedi sledenja, kjer rdeča barva pomeni napovedano odpoved sledenja.

Sledilnik	**Povprečje**	**Uteženo povprečje**	**Združeno**
Natančnost			
SiamFC	**0, 52**	**0, 51**	**0, 52**
SiamFC+Pv(det)	0, 47	0, 47	0, 47
SiamFC+Pv	0, 47	0, 46	0, 46
SiamFC+Pt(det)	0, 51	0, 51	0, 51
SiamFC+Pt	0, 50	0, 50	0, 50
Robustnost			
SiamFC	25, 50	30, 03	104, 00
SiamFC+Pv(det)	**19, 33**	**22, 41**	**86, 00**
SiamFC+Pv	20, 33	23, 08	**86, 00**
SiamFC+Pt(det)	21, 00	24, 44	91, 00
SiamFC+Pt	21, 50	25, 69	91, 00
Pričakovano prekrivanje			
SiamFC	0, 2353		
SiamFC+Pv(det)	0, 2466		
SiamFC+Pv	0, 2396		
SiamFC+Pt(det)	**0, 2471**		
SiamFC+Pt	0, 2441		

Tabela 5.3: Tabela prikazuje rezultate glavnega eksperimenta VOT2016 [22], kjer odebeljena števila prikazujejo boljše rezultate.

5.3 Testiranje dolgoročnega sledenja

Našo implementacijo dolgoročnega sledilnika HSDO smo poleg rezultatov sledilnika SiamFC [6] primerjali še z rezultati pred kratkim objavljenega sledilnika PTAV [26], ki za sledenje uporablja podoben pristop. Za podatkovno zbirko smo vzeli TC-128 [24] in UAV20L [25], pri katerem je slednja namenjena prav testiranju dolgoročnih sledilnikov. Implementacijo sledilnika HSDO smo opisali v Poglavju 4.2, SiamFC smo opisali v Poglavju 3, PTAV pa omenimo v Poglavju 1.1. Testirali smo rezultate sledilnikov HSDO, HSDO+Pv, SiamFC, SiamFC+Pv in PTAV, pri čemer oznaka +Pv pomeni sledenje s posodabljanjem predloge, ki ga omenimo v Poglavju 4.3.

Pri dolgoročnem sledenju testiramo sledilnik tako, da mu na začetku sekvence podamo označen okvir, ki predstavlja predlogo objekta, nato pa mora sledilnik slediti vse do konca sekvence. V primeru odpovedi se sledilnik ne ponastavi, kot je pri glavnem eksperimentu VOT2016 [22]. Po koncu vseh sekvenc se izračuna uspešnost prekrivanja in razdalje sredinske točke napovedanega okvirja v odvisnosti od praga.

5.3.1 Podatkovna zbirka TC-123

Podatkovna zbirka TC-128 [24] sestavlja 128 barvnih sekvenc, primernih za testiranje sledilnikov. Služi za spodbujanje vključitve barvnih sekvenc namesto sivinskih pri sledenju. Ker vsebuje krajše sekvence slik, je bolj primerna za evalvacijo kratkoročnih sledilnikov, saj dolgoročnost ne pride do izraza. Kljub temu, da želimo testirati dolgoročno sledenje, smo vseeno testirali sledenje na tej podatkovni zbirki, ker smo imeli rezultate sledilnika PTAV [26] in smo jih lahko podrobno primerjali z našimi rezultati.

Podrobni rezultati podatkovne zbirke TC-123 so prikazani v Sliki 5.3.1 in povzeti v Tabeli 5.4. Naš sledilnik HSDO ima povprečno prekrivanje 0,5164, SiamFC pa 0,5081, kar je približno 2% izboljšava sledilnika HSDO. S posodabljanjem predloge se pri obeh sledilnikih SiamFC+Pv in HDSO+Pv rezultati poslabšajo od obstoječih na vrednosti 0,5065 in 0,5046. Podobne rezultate

dobimo tudi pri razdalji središčne točke napovedanega okvirja in središčne točke označenega okvirja. Sledilnik HSDO ima povprečni rezultat 0,7241, SiamFC pa ima 0,7115, kar je tudi tukaj za približno 2% boljši rezultat sledilnika HSDO. S posodabljanjem predloge tudi tukaj poslabšamo rezultate pri obeh sledilnikih. Ugotovili smo, da z uporabo detekcije objekta dobimo boljše rezultate od osnovnega sledilnika SiamFC, vendar slabše od sledilnika PTAV. Kombinacija s posodabljanjem predloge pa se izkaže kot slabša rešitev pri obeh sledilnikih SiamFC in HSDO.

Kljub temu, da so bili rezultati sledenja s posodabljanjem predloge uspešni pri glavnem eksperimentu VOT2016 [22], se je pri tem izkazalo za neuspešno. Dobre rezultate pri glavnem eksperimentu VOT2016 pripisujemo temu, da se ob odpovedi sledenja sledilnik ponovno ponastavi, torej ob prisotnosti "brezhibnega"detektorja. Tako ugotovimo, da je za učinkovito posodabljanje predloge potrebna brezhibna detekcija odpovedi sledenja in objekta. V tem primeru postane posodabljanje predloge nepotrebno, saj z brezhibnimi modeli ni potrebno zmanjšati števila njihovih posredovanj pri sledenju, kar je bil tudi osnovni namen.

Z uporabo detekcije objekta smo pričakovali boljše rezultate, saj so bili rezultati v naslednji opisani podatkovni zbirki veliko boljši. Ugotovili smo, da ima podatkovna zbirka TC-128 drugačen nabor posnetkov kot pa UAV20L, ki otežujejo delo detektorja objekta. Testna množica TC-128 vsebuje veliko primerov, kjer so na sliki tudi drugi objekti, ki so podobni sledenemu objektu. Tako se ob nepravilni napovedi klasifikatorja odpovedi sledenja pogosto zgodi, da detekcija objekta ponastavi sledilnik na drug objekt, podoben objektu, ki mu sledimo. V takšnih primerih se z uporabo detekcije objekta sledenje poslabša. Primer take napake prikazuje Slika 5.8. Na sliki vidimo, da sledilnik pravilno sledi objektu in to tudi potrjuje klasifikator pravilnega sledenja. V naslednjem koraku klasifikator nepravilno napove odpoved sledenja in se s tem vključi tudi detekcija objekta. Ta najde trenutno najboljše ujemanje na sliki in postavi napovedani okvir na to lokacijo. Na tej lokaciji pa je drug objekt, ki je podoben pravemu objektu. V takem primeru sledilnik

Ime sledilnika	Preciznost	Uspešnost prekrivanja
PTAV	0,7385	0,5447
HSDO	0,7241	0,5164
HSDO+Pv	0,7115	0,5049
SiamFC	0,7096	0,5081
SiamFC+Pv	0,7079	0,5069

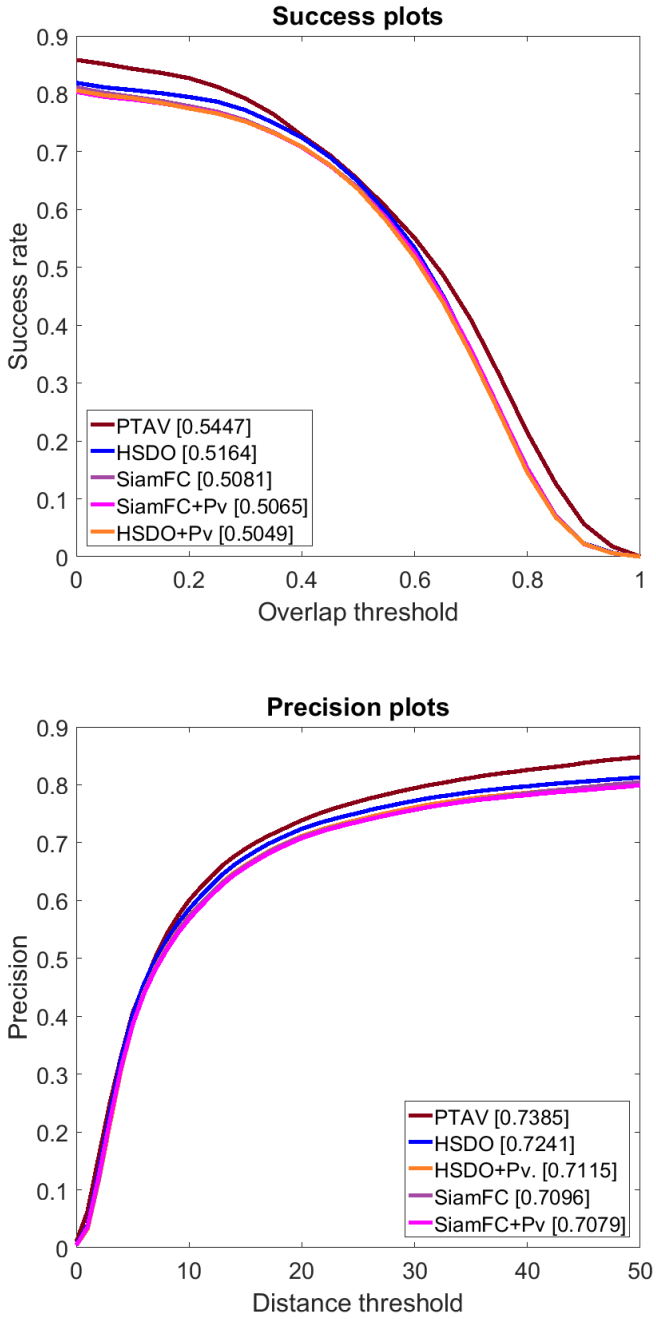
Tabela 5.4: Tabela prikazuje rezultate sledilnikov pri sledenju na podatkovni zbriki TC-123.

še naprej sledi nepravilnemu objektu, saj klasifikator napoveduje pozitivno sledenje in s tem onemogoča detekciji objekta, da se sproži in ponastavi nazaj na pravilni objekt.

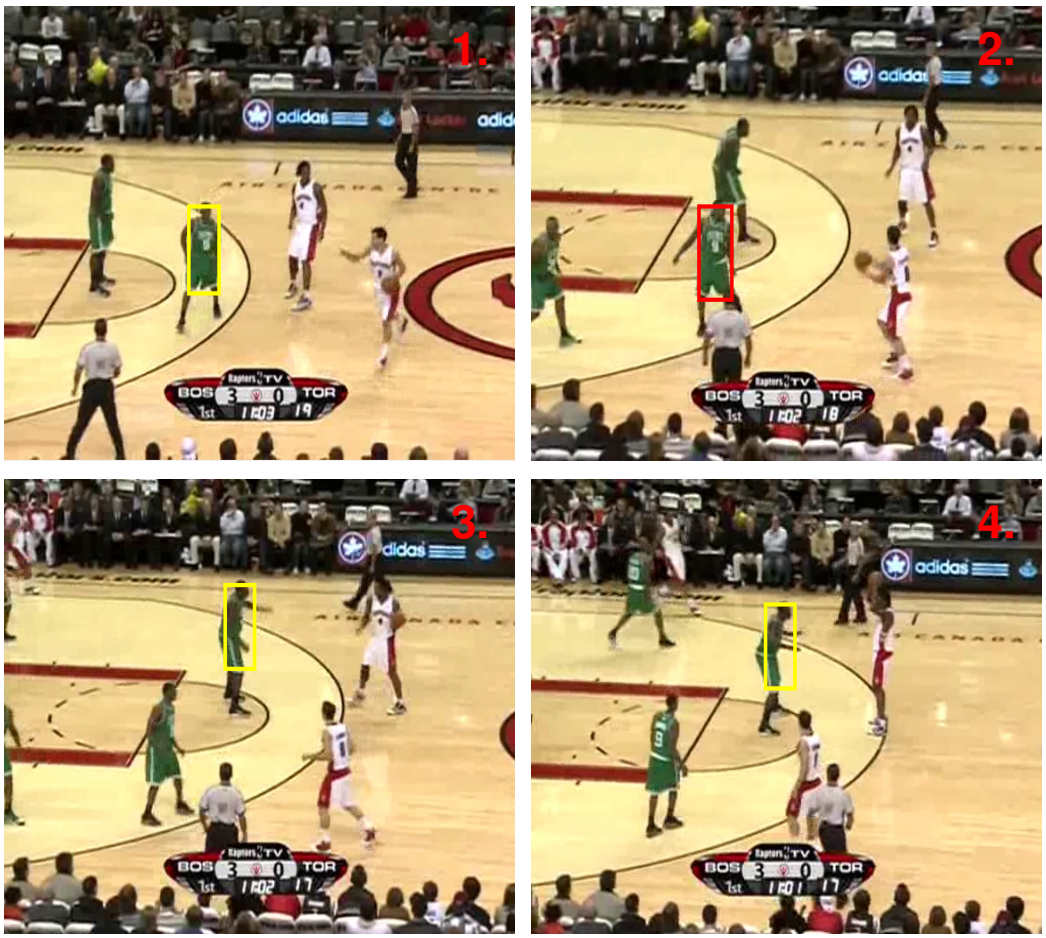
5.3.2 Analiza dolgoročnega sledenja

Podatkovna zbirka UAV20L [25] sestavlja 20 dolgih sekvenc, ki so primerne za dolgoročno sledenje iz zraka. Pri tej podatkovni zbirki pride detekcija objekta bolj do izraza, saj je z daljšimi sekvencami in z odsotnostjo sledenega objekta nujno potrebna za nadaljnje sledenje. Podatkovna zbirka vsebuje tudi primere, ko je sledeni objekt nekaj časa odsoten in se nato spet pojavi. Tak primer prikazuje Slika 5.9. Na njej vidimo primer, ko objekt za nekaj časa izgine iz objektiva kamere, zato začne sledilnik slediti najbolj primernemu objektu. Pri tem model zazna odpoved sledenja in se sproži detekcija objekta. Ko sledeni objekt ponovno pride v objektiv in je zaznan z dovolj veliko verjetnostjo, se sledilnik ponastavi na ta sledeni objekt.

Podrobni rezultati podatkovne zbirke UAV20L so prikazani v Sliki 5.3.2 in povzeti v Tabeli 5.5. Naš sledilnik HSDO je dosegel najboljši rezultat. Povprečno prekrivanje HSDO sledilnika je 0,4913, prekrivanje SiamFC sledilnika pa je 0,4277, kar je 14% izboljšanje. Prav tako so bili rezultati boljši od sledilnika PTAV, ki je dosegel povprečno prekrivanje 0,4230, kar je za 16%



Slika 5.7: Slika prikazuje rezultate sledenja testne množice TC-128.



Slika 5.8: Slika prikazuje primer, ko detektor objekta preskoči iz pravilno sledenega objekta na nepravilnega. Rdeč pravokotnik pomeni, da je klasifikator napovedal odpoved sledenja.

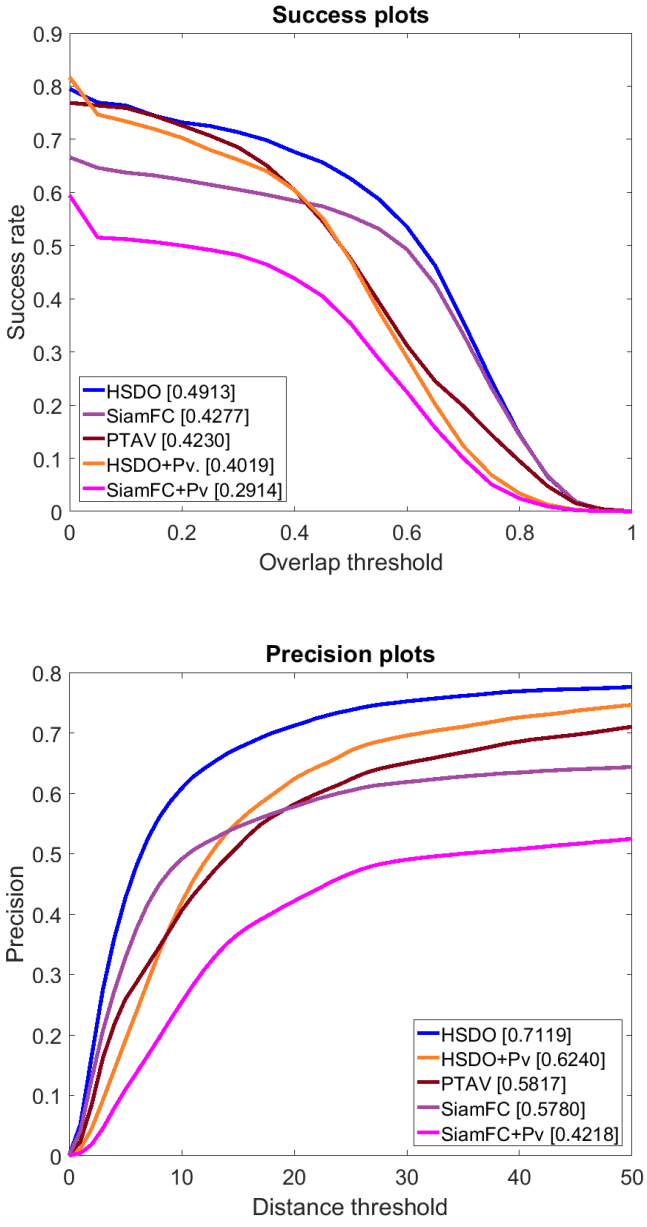
Slika 5.9: Slika prikazuje delovanje detektorja objekta ob zaznani odpovedi sledenja.

boljši rezultat. Podobne rezultate dobimo tudi pri razdalji središčne točke napovedanega okvirja z označenim okvirjem. Sledilnik HSDO ima povprečni rezultat 0,7119, SiamFC pa ima 0,5780, kar je za približno 23% izboljšanje sledilnika HSDO. Prav tako sledilnik HSDO za približno 24% presega sledilnik PTAV [26]. S posodabljanjem predloge tudi tukaj poslabšamo rezultate pri obeh sledilnikih. Rezultati so potrdili uporabnost detekcije objekta, saj

Ime sledilnika	**Preciznost**	**Uspešnost prekrivanja**
HSDO	0,7119	0,4913
SiamFC	0,6240	0,4277
PTAV	0,5817	0,4230
HSDO+Pv	0,5780	0,4019
SiamFC+Pv	0,4218	0,2914

Tabela 5.5: Tabela prikazuje rezultate sledilnikov pri sledenju na podatkovni zbriki UAV20L.

dobimo zanesljivo boljše rezultate od sledilnika SiamFC in prav tako PTAV. Kombinacija s posodabljanjem predloge se tudi tukaj izkaže kot slabša rešitev pri obeh sledilnikih SiamFC in HSDO.



Slika 5.10: Slika prikazuje rezultate sledilnikov, testiranih na podatkovni zbirki UAV20L.

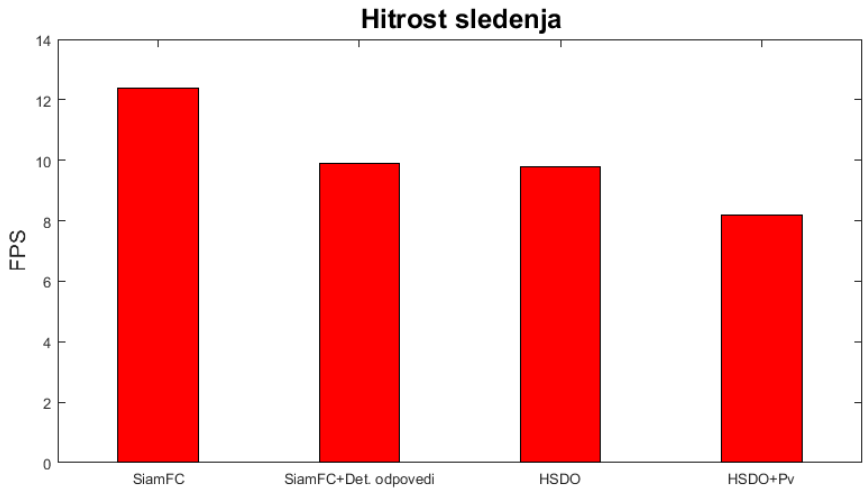
5.4 Analiza hitrosti delovanja

Kljub dobrim rezultatom je bil naš cilj tudi ohraniti hitrost izvajanja sledenja. Hitrost smo analizirali tako, da smo vzeli sledilnik SiamFC [6] in mu postopoma dodajali funkcionalnosti iz Poglavja 4. Izračunali smo povprečno število procesiranih okvirjev na sekundo (fps, angl: frames per second) na podatkovni zbirki UAV20L [25] za vsako dodano funkcionalnost. Za izvajanje sledenja smo uporabili prenosni računalnik s procesorjem Intel Core i7-4710HQ, grafično kartico NVIDIA GeForce GTX 860M in delovnim spominom velikosti 16 GB.

Rezultati analize hitrosti za posamezno dodano funkcionalnost so prikazani na Sliki 5.11 in povzeti v Tabeli 5.6. Ugotovili smo, da izgubimo največ hitrosti pri sami detekciji odpovedi sledenja. Z uporabo modela detekcije objekta izgubimo najmanj hitrosti, saj se ga uporablja samo takrat, ko je zaznana odpoved sledenja. Z uporabo posodabljanja predloge pa dodatno izgubimo na hitrosti, saj se namesto ene predloge računa 5 predlog. Skupna izguba hitrosti vseh združenih funkcionalnosti je približno 35%. Kljub izgubi hitrosti, ki je bila predvidena, lahko še vedno trdimo, da sledilnik ohranja to pozitivno lastnost.

Sledilnik	**Hitrost (FPS)**
SiamFC	12,4
SiamFC+Det. odpovedi	9,9
HSDO	9,8
HSDO+Pv	8,2

Tabela 5.6: Tabela prikazuje hitrosti izvajanja posameznih funkcionalnosti.



Slika 5.11: Slika prikazuje hitrosti izvajanja posameznih funkcionalnosti.

.

Poglavje 6

Sklepne ugotovitve

V delu smo predstavili nov dolgoročni sledilnik. Izhajali smo iz pred kratkim objavljenega sledilnika SiamFC [6], ki temelji na polno konvolucijskih siamskih CNN, zaradi katerih je sledilnik hiter in hkrati natančen, vendar ni primeren za dolgoročno sledenje.

Naš dolgoročni sledilnik opremimo z detekcijo odpovedi sledenja in detekcijo objekta. Detekcija odpovedi sledenja nam pove, kdaj sledilnik pravilno sledi in kdaj je odpovedal. V primeru odpovedi pa se sproži detekcija objekta, ki ponovno poišče iskani objekt. Ker sledilnik pri iskanju objektov uporablja polno konvolucijske siamske CNN, je zato hiter in hkrati natančen pri prepoznavanju objektov.

V delu najprej predlagamo implementacijo detekcije odpovedi sledenja. Tako izvemo, če sledilnik še pravilno sledi ali ne. S to nadgradnjo smo lahko nato uporabili metode, ki omogočajo dolgoročno sledenje.

Z detekcijo odpovedi sledenja smo lahko implementirali detekcijo objekta, ki je ob primeru odpovedi sledenja poiskala objekt na celotni sliki. S tem je sledilnik postal primeren za dolgoročno sledenje, saj ima možnost ponastavitve sledenja ob primerih lastne odpovedi.

Predlagali smo tudi posodabljanje predloge, da bi s tem izboljšali robustnost sledenja. Tako bi zmanjšali število potrebnih sprožitev detekcij objekta in zmanjšali napako sledenja. Predlagali smo dva pristopa. Prvega s

shranjevanjem predlog v vrsto in drugega s postopnim posodabljanjem predloge z novimi primeri.

Najprej smo analizirali detekcijo odpovedi sledenja. Pognali smo eksperiment nenadzorovanega sledenja na podatkovni zbirki VOT2016 [22] tako, da smo beležili napovedi in te primerjali z napovedanimi in označenimi okvirji. Ugotovili smo, da je točnost napovedi odvisna od nastavljenega praga. Za sprožitev detekcije objekta smo ugotovili, da mora biti prag napovedi nastavljen na 15% verjetnost, da sledilnik pravilno sledi, torej mora klasifikator odpovedi sledenja napovedati odpoved sledenja z vsaj 85% verjetnostjo. Za posodabljanje predloge pa mora biti prag nastavljen na 75%. Tako se detektor objekta pravilno sproži v 90% primerov, vendar se pri 46% ne sproži takrat, ko bi se moral. Pri posodabljanju predloge pa klasifikator detekcije odpovedi sledenja napove pravilno v 74% primerov v obeh primerih.

V nadaljevanju smo analizirali predloge izboljšav posodabljanja predloge. Najprej smo testirali posodabljanje na podatkovni zbirki VOT2016, kjer smo pognali glavni eksperiment. Ugotovili smo, da posodabljanje predloge dobro deluje, če vedno pravilno sledimo in se ob odpovedi sledilnik takoj ponastavi. Najboljši rezultat, shranjevanje predlog v vrsto, je imel 21% izboljšanje robustnosti, vendar za 10% slabšo natančnost. S predlogom postopnega posodabljanja pa smo dosegli za 14% boljšo robustnost, vendar samo za 2% slabšo natančnost. Kljub dobrim rezultatom na glavnem eksperimentu VOT2016 smo na podatkovnima zbirkama TC-123 [24] in UAV20L [25] dosegli občutno poslabšanje sledenja, tako da smo posodabljanje predloge ovrgli iz naše implementacije sledilnika.

Implementacijo našega dolgoročnega sledilnika smo testirali na dveh podatkovnih zbirkah TC-123 in UAV20L. Zanimala nas je natančnost v povprečnem prekrivanju in v oddaljenosti središčne točke napovedanega okvirja od dejanskega. Poleg rezultatov SiamFC smo naše rezulate primerjali še z rezultati sledilnika PTAV [26], ki je najsodobneši sledilnik. Pri podatkovni zbirki TC-123, ki je namenjena bolj za kratkoročne sledilnike, smo dosegli le manjšo izboljšavo od sledilnika SiamFC, in sicer za slabih 2%. Sledilnik

PTAV pa je imel boljše rezultate. Pri podatkovni zbirki UAV20L, ki je namenjena testiranju dolgoročnih sledilnikov in je pri tem tudi edina na voljo, pa smo dosegli občutno izboljšanje v sledenju. V primerjavi s sledilnikom SiamFC smo pri povprečnem prekrivanju dosegli 14% izboljšanje, pri oddaljenosti središčne točke okvirja pa 23% izboljšanje. Boljši smo bili tudi od trenutno najboljšega sledilnika PTAV, kjer smo zabeležili 24% izboljšanje v prekrivanju okvirjev in 16% v oddaljenosti središčne točke.

Naš cilj je bil tudi ohraniti hitrost sledenja, ki ga ima sledilnik SiamFC. Merili smo povprečno hitrost sledenja z dodajanjem posameznih funkcionalnosti. Odkrili smo, da se hitrost sicer zmanjša za približno 25%, vendar to ni tako veliko, če ga primerjamo z drugimi sledilniki, ki so še vedno mnogo počasnejši od naše različice.

V delu smo dosegli zastavljen cilj. Naredili smo dolgoročni sledilnik, ki se tudi hitro izvaja. Čeprav na zbirkah za kratkoročno sledenje ni bilo velikih izboljšav, smo na dolgoročni zbirki dosegli dobre rezultate, ki se kosajo z rezultati najsodobnejših sledilnikov.

6.1 Nadaljnje delo

Glavni cilj nadaljnjega dela bi lahko bila uspešna združitev posodabljanja predloge in detekcije odpovedi, ki bi uspešno sodelovala in zajela vse njihove pozitivne lastnosti, robustnost ter dolgoročnost. Pri posodabljanju predloge pride do poslabšanja natančnosti postavitve okvirja objekta, zato bi bilo dobro ločiti sistem za lokacijo in postavitve okvirja, tako da bi slednjega izvajal nek drugi model, ki bi bil zadolžen samo za to. Ugotovili smo tudi, da sledilnik slabo deluje, če sta si na sliki dva objekta podobna in namesto pravemu začne slediti napačnemu. To bi izboljšali z boljšo arhitekturo CNN, ki bi bolj natančno ločila objekte. Lahko bi uporabili arhitekturo CNN dela [7], ki velja za bolj natančno pri razpoznavanju objektov.

Literatura

[1] C. Manresa, J. Varona, R. Mas, F. J. Perales, Hand tracking and gesture recognition for human-computer interaction, ELCVIA Electronic Letters on Computer Vision and Image Analysis 5 (3) (2005) 96–104.

[2] A. Sanin, C. Sanderson, B. C. Lovell, Improved shadow removal for robust person tracking in surveillance scenarios, in: Pattern Recognition (ICPR), 2010 20th International Conference on, IEEE, 2010, pp. 141–144.

[3] Y. Wan, Y. Huang, B. Buckles, Camera calibration and vehicle tracking: Highway traffic video analytics, Transportation Research Part C: Emerging Technologies 44 (2014) 202–213.

[4] G. F. Welch, History: The use of the kalman filter for human motion tracking in virtual reality, Presence: Teleoperators and Virtual Environments 18 (1) (2009) 72–91.

[5] Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-learning-detection, IEEE transactions on pattern analysis and machine intelligence 34 (7) (2012) 1409–1422.

[6] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. Torr, Fully-convolutional siamese networks for object tracking, in: European Conference on Computer Vision, Springer, 2016, pp. 850–865.

[7] R. Tao, E. Gavves, A. W. Smeulders, Siamese instance search for tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1420–1429.

[8] J. Fan, W. Xu, Y. Wu, Y. Gong, Human tracking using convolutional neural networks, IEEE Transactions on Neural Networks 21 (10) (2010) 1610–1623.

[9] L. Wang, W. Ouyang, X. Wang, H. Lu, Visual tracking with fully convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3119–3127.

[10] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using ašiamese"time delay neural network, in: Advances in Neural Information Processing Systems, 1994, pp. 737–744.

[11] F. Pernici, A. Del Bimbo, Object tracking by oversampling local features, IEEE transactions on pattern analysis and machine intelligence 36 (12) (2014) 2538–2551.

[12] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Computer vision and image understanding 110 (3) (2008) 346–359.

[13] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning., in: OSDI, Vol. 16, 2016, pp. 265–283.

[14] A. Vedaldi, K. Lenc, Matconvnet: Convolutional neural networks for matlab, in: Proceedings of the 23rd ACM international conference on Multimedia, ACM, 2015, pp. 689–692.

[15] D. Scherer, A. Müller, S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, Artificial Neural Networks–ICANN 2010 (2010) 92–101.

[16] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[17] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.

[18] M. Khalil-Hani, L. S. Sung, A convolutional neural network approach for face verification, in: High Performance Computing & Simulation (HPCS), 2014 International Conference on, IEEE, 2014, pp. 707–714.

[19] S. Bell, K. Bala, Learning visual similarity for product design with convolutional neural networks, ACM Transactions on Graphics (TOG) 34 (4) (2015) 98.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International Journal of Computer Vision 115 (3) (2015) 211–252.

[21] F. J. Harris, On the use of windows for harmonic analysis with the discrete fourier transform, Proceedings of the IEEE 66 (1) (1978) 24–25.

[22] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojír, G. Häger, A. Lukežič, G. Fernández, et al., The visual object tracking vot2016 challenge results, in: European Conference on Computer Vision, Springer, 2016, pp. 777–823.

[23] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, L. Čehovin, A novel performance evaluation methodology for single-target trackers, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (11) (2016) 2137–2155. doi:10.1109/TPAMI.2016.2516982.

[24] P. Liang, E. Blasch, H. Ling, Encoding color information for visual tracking: Algorithms and benchmark, IEEE Transactions on Image Processing 24 (12) (2015) 5630–5644.

[25] M. Mueller, N. Smith, B. Ghanem, A benchmark and simulator for uav tracking, in: European Conference on Computer Vision, Springer, 2016, pp. 445–461.

[26] H. Fan and H. Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In *Int. Conf. Computer Vision*, pages 5486–5494, 2017.