Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Grega Kres

Razpoznavanje znakov prstne abecede na osnovi računalniškega vida

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

doc. dr. Iztok Lebar Bajec
MENTOR

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00069/2011
Datum: 04.02.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **GREGA KRES**

Naslov: **RAZPOZNAVANJE ZNAKOV PRSTNE ABECEDE NA OSNOVI RAČUNALNIŠKEGA VIDA**

COMPUTER VISION BASED SIGN LANGUAGE RECOGNITION

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Veliko število tehnično zagnanih raziskovalcev se navdušuje nad videnjem prihodnosti, prikazane v znanstveno-fantastičnih filmih in prav ti raziskovalci nato premikajo meje znanosti s črpanjem idej iz teh filmov. Film Minority Report, na primer, prikazuje interakcijo z računalnikom na osnovi kretenj.

V diplomski nalogi zasnujte programsko opremo za razpoznavanje znakov ameriške prstne abecede. Osredotočite se predvsem na uporabo odprtokodne knjižnice OpenCV. Realizirajte dva pristopa, kjer v prvem za razpoznavo prikazanega znaka uporabljate analizo značilk, v drugem pa klasificirate na osnovi ujemanja s predlogo. Proučite predvsem razlike v hitrosti in natančnosti klasifikacije, in nakažite pristop implementacije razpoznavanja besed v primeru tekočega videa.

Mentor:

doc. dr. Iztok Lebar Bajec

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Grega Kres,

z vpisno številko 63050277,

sem avtor diplomskega dela z naslovom:

Razpoznavanje znakov prstne abecede na osnovi računalniškega vida

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Iztok Lebar Bajec
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 23.3.2011 Podpis avtorja:

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Grega Kres

Razpoznavanje znakov prstne abecede na osnovi računalniškega vida

POVZETEK

V diplomski nalogi predstavljamo metode za prepoznavo znakov ameriške prstne abecede s slik oziroma videa. Ameriško prstno abecedo smo izbrali zaradi dostopnosti gradiva in lastnosti same abecede (enoročna in majhno število znakov, ki vsebujejo gibanje). Metode so smiselno razdeljene v dve poglavji.

V prvem opisujemo tiste postopke, ki iz slike predhodno zajamejo značilke znaka nato pa operirajo na teh številčnih podatkih. Podrobno opisujemo segmentacijo slike, zajem značilk ter samo klasifikacijo, ki temelji na metodi najbližjih sosedov. Sledi še opis postopka izračuna podobnosti med znaki, ker gre za pomemben del metode najbližjih sosedov.

V drugem poglavju opisujemo postopke, ki znake klasificirajo na podlagi podobnosti med vhodnim znakom in predlogo, dobljeno iz učne množice. Za razliko od postopkov predstavljenih v prvem poglavju, ki uporabljajo značilke v številčni obliki, so predloge v slikovnem formatu. Razvili smo tri različne oblike klasifikacije, ki se razlikujejo po hitrosti in natančnosti.

Pri testiranju se osredotočamo na hitrost in natančnost klasifikacije. Testirali smo na podlagi posameznih slik in videa. Pri klasifikaciji na podlagi videa prihaja do zapletov, še posebej kadar klasificiramo v realnem času, s spletno kamero, zato v tem poglavju opisujemo še različne pristope za izboljšanje natančnosti.

V zaključku na kratko predstavimo korake, ki utegnejo izboljšati natančnost in bi bile primerne za nadaljnje raziskovanje.

Ključne besede: računalniški vid, prstna abeceda, ujemanje s predlogo, klasifikacija

University of Ljubljana
Faculty of Computer and Information Science

Grega Kres

Computer vision based sign language recognition

ABSTRACT

This thesis focuses on methods designed to recognize signs of American manual alphabet on still images and video. The American manual alphabet was chosen due to the availability of material and properties of the alphabet itself (singlehanded and low amount of signs requiring movement). Methods are divided into two chapters.

The first chapter describes methods based on feature extraction. The image is first segmented using color filters, then the features are extracted and converted into numerical form, and finally classification is performed. Classification is based on nearest neighbor search, which requires a metric to be defined so distance to neighboring examples can be calculated. The metric used is detailed in this chapter as well.

The second chapter describes methods based on template matching. Unlike methods used in the previous chapter, templates are not represented in numerical form but rather as binary images. A set of templates is constructed using a group of training images. An input image is then compared to every template in the set and the best match is returned. We have developed three alterations of the algorithm, each having different classification accuracy and speed.

We then focus on testing the speed and accuracy of the classification. Classification is tested using both still images and video. When testing is performed on video, certain problems occur, especially when capturing video in real-time using a web cam. Due to a generally lower quality of such capture, noise is introduced to the image, which severely affects classification accuracy. Certain methods are explored that help avoid the issue.

In the final chapter we propose improvements that could be the focus of further research.

Key words: computer vision, manual alphabet, template matching, classification

ZAHVALA

Zahvaljujem se mentorju doc. dr. Iztoku Lebar Bajcu za nasvete in pomoč pri izdelavi naloge.

— Grega Kres, Ljubljana, marec 2011.

KAZALO

1 Uvod

1.1 Motivacija

Cilj diplomske naloge je razviti aplikacijo, ki bo znala s slike prepoznati znak prstne abecede, oziroma iz videa razbrala zaporedje znakov in sestavila besedo. Celoten postopek je sestavljen iz sledečih korakov:

- zajem slike: zajem slike s spletno kamero;
- lokalizacija znaka: iskanje območja na sliki, kjer se nahaja znak;
- zajem značilk: opredelitev lastnosti znaka;
- klasifikacija: prepoznavanje znaka kot ene črke abecede;
- sestavljanje: sestavljanje črk v besedo.

1.2 Znakovni jezik

Znakovni jezik je jezik, ki namesto uporabe zvoka in slušnega zaznavanja, za prenos informacij uporablja kretnje in vidno zaznavanje. Pri tem se uporabljajo kretnje rok,

govorica telesa in obrazna mimika.

Uporablja se v okoljih, kjer je uporaba govornega jezika ovirana ali otežena, na primer v hrupnih okoljih ali okoljih, kjer je potrebna tišina. Nenazadnje pa tudi v okoljih, kjer uporaba govora ni mogoča, na primer v skupnosti gluhih in naglušnih ljudi.

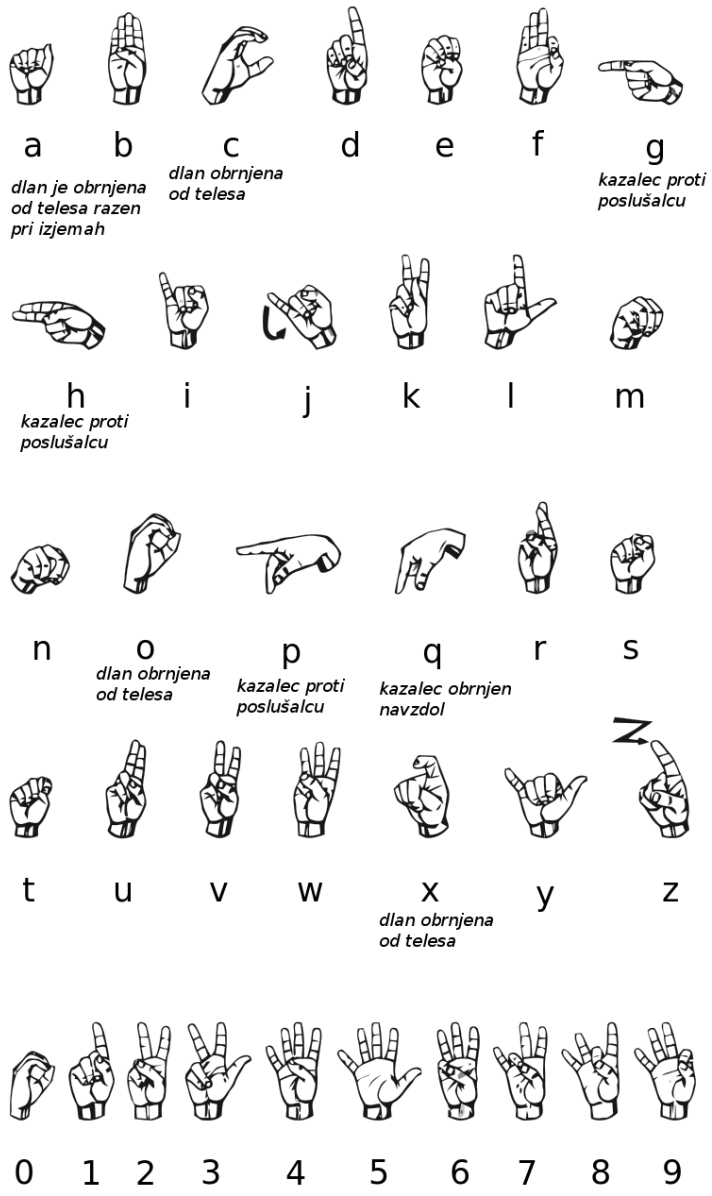
Znakovni jezik, ki ga uporabljajo gluhi, je lignvistično gledano ravno tako bogat in kompleksen kot govorni jeziki, med seboj pa sta popolnoma neodvisna. Ta neodvisnost je vidna v primerih držav, ki imajo skupni govorni jezik, znakovni jezik pa je različen (ZDA in Velika Britanija), kakor tudi v primerih, ko ima država več uradnih govornih jezikov a le en znakovni jezik (Južnoafriška Republika). Poleg tega ima znakovni jezik svoja slovnična pravila in drugačen vrstni red. Pogosto prepričanje, da je znakovni jezik le s kretnjami izražen govorni jezik je tako napačno.

1.3 Prstna abeceda

Za razliko od znakovnega jezika, ki je od govornega neodvisen, pa je prstna abeceda tesno povezana s standardno abecedo. Vsak znak predstavlja eno črko standardne abecede (glej sliko 1.1). Na nek način služi kot most med znakovnim in govornim jezikom. Črkuje se navadno imena krajev, držav, ljudi, kadar poudarjamo ali pa kadar neka beseda še nima svoje kretnje v znakovnem jeziku. Prstnih abeced je več, katero uporablja posamezen jezik pa je odvisno od jezika samega. V prstni abecedi velja nekaj osnovnih pravil:

- razen redkih izjem je dlan roke obrnjena proti poslušalcu,
- pri črkovanju je roka na istem mestu in ne poskakuje preveč,
- pri črkovanju več besed se presledek pokaže kot krajši premor,
- poskakovanje roke pri črkovanju pomeni podvojeno črko (na primer črka “D” pri besedi “oddaja”),
- jasnost je pomembnejša od hitrosti črkovanja.

Glede na osnovne lastnosti prstne abecede smo se zaradi lažjega iskanja znaka na vhodni sliki odločili uporabiti barvno rokavico. Bombažno rokavico bele barve smo pobarvali na sledeč način: palec smo pobarvali rdeče, kazalec modro, sredinec, prstanec in mezinec pa zeleno. Preostali del rokavice smo pustili bele barve.



Slika 1.1 Ameriška prstna abeceda.

1.4 Organizacija diplomske naloge

Glavni del diplomske naloge je sestavljen iz treh poglavij. V prvem poglavju so predstavljene metode, katerih klasifikacija temelji na posameznih značilkah, ki se zajamejo iz vhodnih slik. Poleg teh metod pa so opisani tudi postopki potrebni za zajem.

Metode, predstavljene v drugem poglavju klasificirajo na osnovi predlog. Namesto primerjanja posameznih značilk slike, te metode primerjajo vhodno sliko s predlogo. Klasifikacija se nato vrši glede na na podobnost s predlogami.

V tretjem poglavju so predstavljeni testi metod in rezultati. Testirali smo klasifikacijsko natančnost in hitrost, ter opisali vzroke za dobljene rezultate.

1.5 Uporabljena orodja

Pri razvoju aplikacije smo uporabili programski jezik Python in knjižico OpenCV. Ker gre za interpretni jezik, je izbira na prvi pogled nekoliko nenavadna, saj je hitrost ključnega pomena, interpretni jeziki pa so zaradi zasnove delovanja počasni. Vendar Python uporablja vmesnik do C/C++ verzije knjižice, zato izbira jezika ne vpliva na hitrost izvajanja OpenCV funkcij.

OpenCV je odprtokodna knjižica namenjena uporabi v računalniškem vidu. Napisana je v jezikih C in C++ in je na voljo je v operacijskih sistemih Windows, Linux in Mac OS X. Zasnovana je bila z namenom učinkovitosti s poudarkom na aplikacijah v realnem času [2].

Njen cilj je zagotoviti preprosto ogrodje, ki dovoljuje hitro grajenje naprednih aplikacij z uporabo računalniškega vida. Vsebuje preko petsto funkcij, ki obsegajo široko področje, od robotike, stereo vida do analize medicinskih slik in iskanja napak na tovarniških izdelkih. Nekatere izmed funkcij knjižice OpenCV, ki smo jih uporabili so:

- cvThreshold: pragovna funkcija, ki na podlagi praga (angl. threshold) ustvari binarno sliko;
- cvConvertImage: funkcija za pretvorbo med različnimi barvnimi formati;
- cvCreateCameraCapture: zajem slike s spletne kamere;
- cvCreateFileCapture: zajem slike z diska;
- cvSmooth: filtriranje z mediano;

- cvInRangeS: barvno filtriranje;
- CBlobResult: analiza blobov;
- cvMatchTemplate: iskanje ujemanja s predlogo.

2 Klasifikacija na podlagi značilk

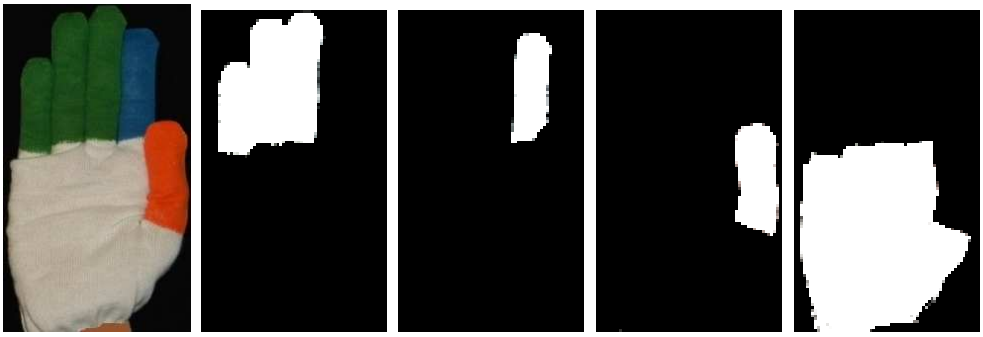
2.1 Zajem značilk

Postopki opisani v tem poglavju temeljijo na primerjanju značilk zajetih na vhodni sliki z značilkami zajetimi na primerih iz učne množice. Te značilke so zapisane v numerični obliki, iz slik pa se zajamejo z uporabo analize blobov (povezanih regij slike, ki so temnejše oziroma svetlejše od okolice). Analiza takih regij je podrobneje opisana v poglavju 2.1.2.

Postopke lahko razdelimo na dva konceptualna dela. Prvi del je zajem značilk iz slike, drugi del pa sama klasifikacija, ki s pomočjo primerjanja značilk poišče tisti primer znaka iz učne množice, ki je vhodnemu znaku najbolj podoben.

2.1.1 Segmentacija slike

Vhodno sliko smo segmentirali s pomočjo barvnih filtrov. Te smo določili na podlagi vnaprej znanih barv rokavice. Vsak filter ima zgornjo in spodnjo mejo, ki so določene tako, da se intervali med seboj ne prekrivajo. Vsak slikovni element vhodne slike torej pripada eni od petih množic; ustreza enemu od štirih filtrov ter je del ene od barvnih regij ali pa ne ustreza nobenemu od filtrov in je del ozadja. Rezultat segmentacije so



Slika 2.1 Segmentacija znaka za črko "B".

štiri binarne slike, kjer vrednost 0 predstavlja ozadje, vrednost 1 pa barvno regijo, ki je ustrezala barvnemu filtru. Na sliki 2.1 si v zaporedju od leve proti desni sledijo: vhodna slika znaka za črko "B" in rezultat segmentacije – zelena, modra, rdeča in bela barvna regija vhodne slike, ki so vidne kot bel objekt na črnem ozadju.

2.1.2 Iskanje objektov s pomočjo analize blobov

Iskanje objektov se izvede s pomočjo analize blobov posameznih barvnih regij. Na vsaki izmed štirih vhodnih slik, ki predstavljajo posamezne barvne regije, se poišče objekte, ki niso del ozadja (so bele barve) in so primerno veliki: večji od 0.5% in manjši od 50% površine slike. Vsak objekt je povezana celota, njihovo število je neodvisno od števila barvnih regij. Eno barvno regijo lahko sestavlja tudi več nepovezanih objektov (kadar se dva objekta stikata, se obravnavata kot en sam objekt) ali pa noben, na primer kadar določena barva na sliki ni vidna.

Slika 2.2 Povezava med barvno regijo in objekti regije.

Na sliki 2.2 je na levi strani znak, ki predstavlja črko "W". Na njej so vidne vse štiri barvne regije (zelena, modra, rdeča in bela). Na desni strani pa je zelena barvna regija razdeljena na tri objekte, ki jo sestavljajo. V primeru črke "B" (slika 2.1) zeleno regijo

predstavlja en sam objekt.

Analiza blobov za delovanje uporablja algoritem imenovan *linearno-časovni algoritem za označevanje komponent z uporabo tehnike sledenja robovom* (angl. A linear-time component labeling algorithm using contour tracing technique) [1]. Algoritem začne obdelavo slike v zgornji vrstici, pri najbolj levem elementu in nadaljuje proti desni in navzdol. Sestavljajo ga štirje koraki:

- ko algoritem pride do zunanjega roba objekta mu sledi, dokler ne pride nazaj do začetne točke; vse točke roba se označijo[1];
- ko pride do zunanjega roba, kateremu je že pripisana oznaka, nadaljuje po vrstici in vsem elementom iste barve priredi isto oznako;
- če algoritem odkrije neoznačen notranji rob, mu sledi, dokler ne pride na začetno točko, vse točke roba pa označi;
- v kolikor odkrije označen notranji rob, nadaljuje po vrstici in vse elemente iste barve označi.

Zgornji postopek naredi le en prehod preko slike in vsakemu slikovnemu elementu pripiše že obstoječo označbo, ali pa novo označbo, v kolikor gre za novo odkrit blob.

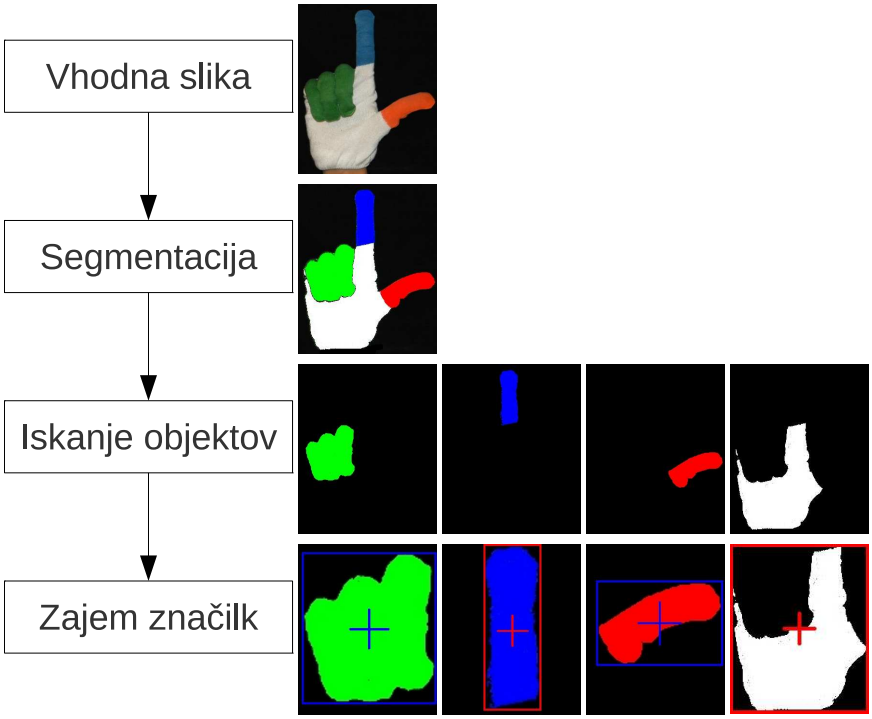
2.1.3 Zajem značilk najdenih objektov

Slika 2.3 prikazuje korake zajema značilk. Vhodno sliko najprej segmentiramo, poiščemo objekte, nato zajamemo značilke posameznih objektov. Na slikah pri zadnjem koraku so prikazani objekti in nekatere njihove značilke. Barvni del slike predstavlja glavni objekt, črni del pa ozadje, ki ga za voljo dodatnega kriterija razločevanja med objekti obravnavamo kot svoj objekt, t.i. objekt ozadja.

Torej, ko s pomočjo iskanja blobov odkrijemo vse objekte, ki so na sliki vidni, jim določimo značilke:

- *višina in širina objekta*: višina se izračuna na podlagi razdalje med y-koordinatama najvišje in najnižje, širina pa na podlagi razdalje med x-koordinatama najbolj leve in najbolj desne točke objekta;
- *površina objekta*: dejanska površina, ki jo določa število slikovnih elementov, ki so del objekta;

[1] Označevanje je namenjeno ločevanju med več objekti, ki so lahko na sliki vidni.



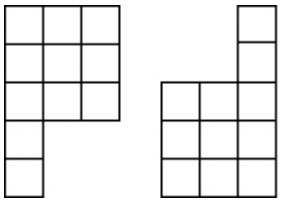
Slika 2.3 Zajem značilk.

- *centroid*: točka, ki predstavlja središče prostorskega obsega[2] objekta;
- *razlika površin*: količnik dejanske površine objekta in površine prostorskega obsega;
- *razdalja in kot* glede na modro regijo: evklidska razdalja med središčema glavnega objekta in največjega glavnega objekta modre regije ter kot, ki ga oklepa premica, ki poteka skozi središči, z ordinatno osjo;
- *razdalja in kot* glede na rdečo regijo: evklidska razdalja med središčema glavnega objekta in največjega glavnega objekta rdeče regije ter kot, ki ga oklepa premica, ki poteka skozi središči, z ordinatno osjo.

Vendar navedene značilke za ocenjevanje podobnosti dveh objektov še niso dovolj. Zamislimo si objekt nepravilne oblike, ki ga nato zavrtimo za 180° okoli središča (slika 2.4). Če z izbranimi značilkami objekta medsebojno primerjamo, pridemo do zaključka,

[2]Območje znotraj najmanjšega možnega pravokotnika, ki še zajema vse elemente objekta, s stranicami vzporednimi koordinatnima osema (angl. bounding box).

da sta 100% identična, ker so vse značilke enake. Potrebujemo torej nek dodaten kriterij, s katerim bomo lahko tudi takšne objekte ločili.



Slika 2.4 Razločevanje objektov.

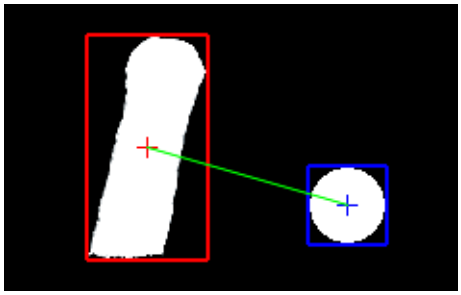
Ker so objekti nepravilnih oblik, kar v našem primeru pomeni, da je njihova površina manjša kot površina prostorskega obsega, lahko poiščemo območja v notranjosti prostorskega obsega, ki objektu ne pripadajo (na binarni sliki so to slikovni elementi, ki imajo vrednost 0 oziroma so črne barve).

Ko tako območje odkrijemo, ga obravnavamo kot nov podobjekt (objekt ozadja). Izmerimo njegove lastnosti in jih dodamo ostalim značilkam trenutno obravnavanega objekta. V kolikor je takih podobjektov (objektov ozadja) več, upoštevamo le površinsko največjega. Lastnosti objekta ozadja, ki nas zanimajo so:

- *površina, razdalja in kot*: dejanska površina objekta ozadja, evklidska razdalja med središčem trenutno obravnavanega/glavnega objekta in središčem objekta ozadja ter kot, ki ga oklepa premica, ki poteka skozi središči, z ordinatno osjo.

Ker so poleg lastnosti posameznih barvnih regij pomembne tudi lastnosti celote, vse barvne regije združimo v en objekt in zajamemo njegove značilke. Obravnavamo ga kot vsak drug objekt.

Na sliki 2.5 so prikazane nekatere značilke objektov. Črna barva predstavlja ozadje, bela barva pa objekte. Na sliki sta prikazana dva objekta iste barvne regije, ki imata z rdečim oziroma modrim križcem označeno središče. Pravokotnika okoli obeh objektov predstavljata prostorski obseg, zelena črta med središčema pa razdaljo med njima.



Slika 2.5 Značilke objektov.

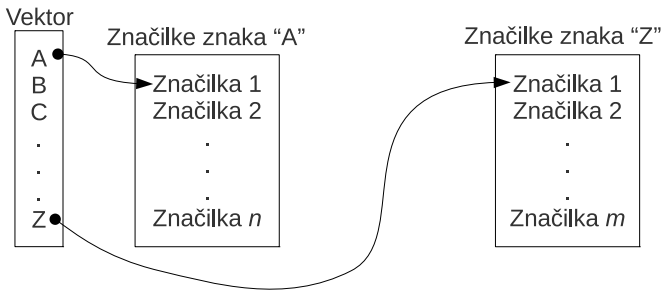
2.2 Klasifikacija

Naloga klasifikatorja je, da nek objekt, ki je opisan z množico značilk oziroma atributov, uvrsti v enega od razredov, ki so na voljo. V našem primeru so objekti znaki Ameriške prstne abecede, razredi pa črke standardne abecede.

2.2.1 Iskanje najbližjega soseda

Klasifikator imenovan *iskanje najbližjega soseda* (ang. nearest neighbor search) klasificira znake na podlagi znanja, pridobljenega iz učne množice. Realizirali smo njegovo najpreprostejšo obliko, ki si samo zapomni vse učne primere. Pri klasifikaciji se nato iz učne množice poišče najbolj podoben primer [3].

Bazo znanja smo zapisali kot vektor, sestavljen iz šestindvajsetih elementov, vsaki črki abecede pripada en element. Učna množica, na podlagi katere se vektor zgradi, je namreč sestavljena iz šestindvajsetih slik. Na vsaki sliki je na črnem ozadju znak, ki predstavlja določeno črko. Iz slik se zajamejo značilke objektov, ki se nato zapišejo v vektor, k ustreznemu elementu. Vsak element sestavlja več značilk (slika 2.6), pri čemer je njihovo število odvisno od števila objektov na sliki.



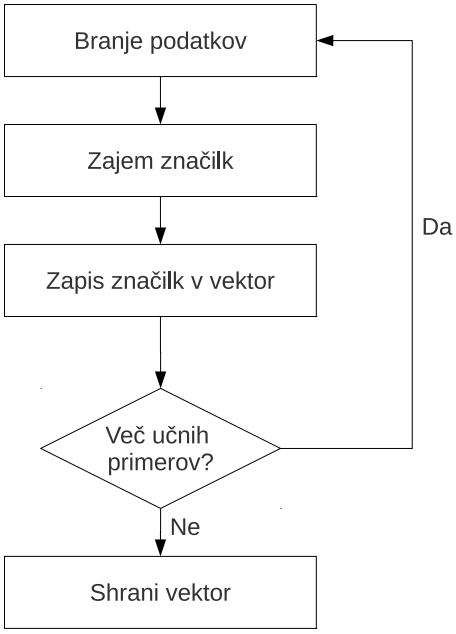
Slika 2.6 Zgradba vektorja, ki predstavlja bazo znanja.

Klasifikacija je sestavljena iz treh korakov:

- učenja oz. vzpostavitve baze znanja,
- analize vhodnega znaka, in
- iskanja najbolj podobnega znaka.

Učenje

Cilj prvega koraka je vzpostavitev baze znanja, ki jo zgradimo na podlagi množice učnih primerov. Nad vsakim primerom učne množice izvedemo zajem značilk (opisan v poglavju 2.1). Te značilke nato dodamo v bazo znanja, ki smo jo zapisali kot vektor, kjer je vsak element sestavljen iz množice numeričnih značilk. Slika 2.7 prikazuje postopek učenja.



Slika 2.7 Učenje.

V kolikor bazo znanja ustrezno shranimo, na primer v datoteko, je učenje potrebno izvesti le enkrat. Vsako naslednje klasificiranje lahko uporabi že zgrajeno bazo.

Analiza vhodnega znaka

Ker smo značilke v vektor zapisali v numerični obliki, potrebujemo še značilke vhodnega znaka v isti obliki. Spet izvedemo zajem značilk iz poglavja 2.1, tokrat nad vhodno sliko.

S tem dobimo množico značilk, ki jo lahko primerjamo z značilkami posameznega znaka iz baze znanja.

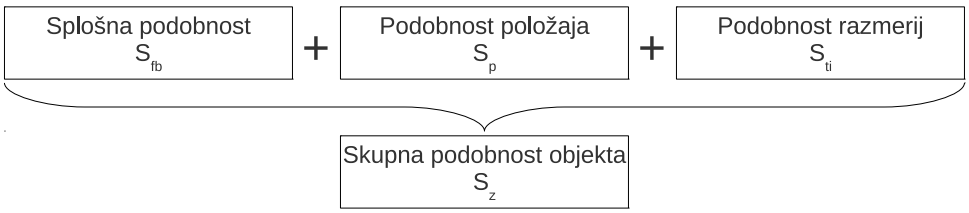
Iskanje najbolj podobnega znaka

Cilj koraka je poiskati znak iz učne množice, ki je najbolj podoben vhodnemu znaku. Znake primerjamo na podlagi objektov, ki jih sestavljajo, objekte pa na podlagi značilk, ki jih opisujejo. Ocena podobnosti je realno število med 0 in 1, kjer 1 predstavlja popolno ujemanje. Podobnost značilk izračunamo po formuli:

$$F = 1 - |F_\mathrm{t} - F_\mathrm{i}|/F_\mathrm{t}, \tag{2.1}$$

kjer je F_t značilka iz učne množice, F_i pa značilka zajeta iz vhodnega znaka.

Da lahko trdimo, da sta dva znaka enaka, ni dovolj, da se objekti ujemajo v smislu oblike, velikosti itd. ampak morajo biti ti objekti tudi na pravem mestu, gledano relativno drug na drugega. Podobnost med objekti izračunamo v štirih korakih, prikazanih na sliki 2.8. Vsak korak je utežena vsota izbranih značilk ali vmesnih ocen, pri čemer so uteži določene empirično na osnovi poizkušanja.



Slika 2.8 Izračun podobnosti enega objekta ene barvne regije.

V prvem koraku ocenimo splošno podobnost med objektom iz vhodne slike in objektom iz učne množice. Najprej izračunamo podobnost njunih objektov ozadja (2.2). Pri tem upoštevamo oceno podobnosti njunih površin (a_b) in položaja. Položaj sestavljata dve značilki. To sta oddaljenosti središča podobjekta ozadja od središča pripadajočega glavnega objekta in kot med premico (ki poteka skozi središči) ter ordinatno osjo. Ocena podobnosti med tema dvema značilkama objekta iz vhodne slike in učne množice označimo kot r_b (za oddaljenost) in φ_b (za kot). Nato izračunamo še podobnost glavnih objektov (2.3), in sicer na podlagi podobnosti višin (h_f), širin (w_f), površin (a_f) in razlike površin (d_f) med značilkami objekta iz vhodne slike in učne množice. Splošna podobnost objekta (2.4) je seštevek teh dveh skupin, primerno utežen.

$$S_{\mathrm{b}} = (a_{\mathrm{b}} \times 0.3) + (r_{\mathrm{b}} \times 0.4) + (\varphi_{\mathrm{b}} \times 0.3) \tag{2.2}$$

$$S_{\mathrm{f}} = (h_{\mathrm{f}} \times 0.25) + (w_{\mathrm{f}} \times 0.25) + (a_{\mathrm{f}} \times 0.25) + (d_{\mathrm{f}} \times 0.25) \tag{2.3}$$

$$S_{\mathrm{fb}} = (S_{\mathrm{b}} \times 0.5) + (S_{\mathrm{f}} \times 0.5) \tag{2.4}$$

V drugem koraku ocenimo še podobnost položaja (2.5) glavnega objekta glede na objekt dlani (med objektom vhodne slike in objektom iz učne množice), upoštevajoč razdaljo med njunima središčema (r_{d}) in kot φ_{d} (med premico, ki jo določata točki središča in ordinato):

$$S_{\mathrm{p}} = (r_{\mathrm{d}} \times 0.5) + (\varphi_{\mathrm{d}} \times 0.5) \tag{2.5}$$

Trenutno obravnavani glavni objekt je v nekem razmerju do objektov, ki predstavljata palec in kazalec. To razmerje določa razdalja med središčem obravnavanega glavnega objekta in središčem glavnega objekta palca oziroma kazalca in kot (med premico, ki jo določata točki njunih središč in ordinato). Zanima nas, kolikšna je podobnost med razmerji teh objektov na vhodni sliki in sliki iz učne množice. Najprej izračunamo podobnost razmerij med glavnim objektom in palcem (2.6) nato pa še podobnost med glavnim objektom in kazalcem (2.7). Pri tem upoštevamo razdaljo (r_{t}) oz. (r_{i}) in kot φ_{t} oz. φ_{i}:

$$S_{\mathrm{t}} = (r_{\mathrm{t}} \times 0.5) + (\varphi_{\mathrm{t}} \times 0.5) \tag{2.6}$$

$$S_{\mathrm{i}} = (r_{\mathrm{i}} \times 0.5) + (\varphi_{\mathrm{i}} \times 0.5) \tag{2.7}$$

$$S_{\mathrm{tk}} = (S_{\mathrm{t}} \times 0.25) + (S_{\mathrm{k}} \times 0.75) \tag{2.8}$$

Skupna podobnost med dvema objektoma je:

$$S_{\mathrm{z}} = (S_{\mathrm{fb}} \times 0.2) + (S_{\mathrm{p}} \times 0.2) + (S_{\mathrm{tk}} \times 0.6) \tag{2.9}$$

Skupna podobnost je torej mera, ki določa podobnost dveh objektov, zasnovana pa je na podobnosti objektov istih barvnih regij. Primerjajo se torej samo objekti iste barvne regije. Težava nastane, kadar imata barvni regiji vhodnega znaka ali znaka iz vektorja učne množice več kot en element. Ker ne moremo preprosto ugotoviti kateri objekt sovpade s katerim, da bi lahko izvedli le primerjanje med njima, je takrat število ocen podobnosti enako zmnožku števila objektov. Primer: zelena barvna regija na vhodni

sliki obsega tri objekte, prav tako kot zelena barvna regija na sliki iz učne množice. Ker primerjamo vsak objekt z vsakim, dobimo devet ocen podobnosti objektov. Če vsak objekt predstavlja en prst, smo na ta način (med drugim) primerjali mezinec in prstanec, kar ni smiselno, vendar pa med objekti ne ločimo. Predpostavimo pa lahko, da so objekti z najboljšo oceno ujemanja tisti, pri katerih je prišlo do primerjanja istoležnih objektov, zato vzamemo le n-najboljših, pri čemer je n enako številu objektov te barvne regije pri vhodnem znaku.

Na voljo imamo ocene podobnosti objektov posamezne barvne regije vhodne slike z objekti iste barvne regije v učni množici, cilj pa je določiti podobnost znaka na vhodni sliki in znaka v učni množici. Ocene zato najprej združimo po posameznih barvnih regijah in izračunamo povprečno oceno posamezne barvne regije R:

$$R = \frac{1}{m} \sum_{i=1}^{m} S_{\mathrm{z}} \tag{2.10}$$

kjer je m število objektov obravnavane barne regije, S_{z} pa ocena podobnosti posameznega objekta te regije.

Potrebno je še združenje ocen posameznih barvnih regij v skupno, končno oceno podobnosti obravnavanega znaka z znakom iz učne množice. Izkaže se, da barvne regije pri končni oceni niso enako pomembne (2.11), zato smo pomembnejše regije ustrezno bolj utežili:

$$S = (R_b \times 0.15) + (R_z \times 0.1) + (R_m \times 0.15) + (R_r \times 0.3) + (R_c \times 0.3) \tag{2.11}$$

Rezultat S (kjer so R_{b} podobnost bele regije, R_{z} podobnost zelene regije, R_{m} podobnost modre regije, R_{r} podobnost rdeče regije in R_{c} podobnost celote) je realno število, ki predstavlja podobnost znaka na vhodni sliki s točno določenim znakom učne množice. Ker vektor učnih primerov obsega šestindvajset elementov je potrebno podobnosti izračunati za vsak element nato pa v primeru iskanja najbližjega soseda izbrati tistega, ki ima najvišjo oceno.

2.2.2 Iskanje k-najbližjih sosedov

Iskanje k-najbližjih sosedov je razširitev algoritma predstavljenega v poglavju 2.2.1 tako, da učna množica namesto šestindvajsetih primerov obsega oseminsedemdeset primerov, po tri za vsako črko. Postopek ugotavljanja podobnosti znakov je enak, vendar pa se

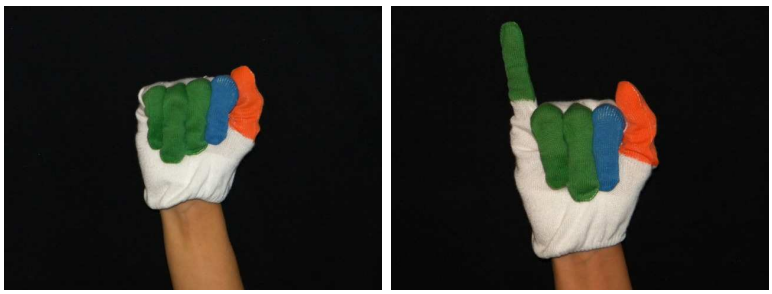
znak ne klasificira v razred z najboljšim ujemanjem ampak se upošteva najboljša tri ujemanja. Vsako ujemanje nato "glasuje" za svoj razred, teža glasu pa je odvisna od ocene podobnosti. Vhodni podatek se klasificira v razred z največ glasovi.

Kot primer vzemimo klasifikacijo, kjer so bile najboljše tri ocene: $S_1(\text{"A"}) = 0.9$, $S_2(\text{"B"}) = 0.4$, $S_3(\text{"B"}) = 0.3$. V primeru uporabe neuteženih glasov bi znak klasificirali kot črko B, saj sta sta bili dve oceni od treh za ta razred ($\frac{2}{3} = 66\%$ glasov). Ker so glasovi uteženi z oceno ujemanja, klasificiramo kot črko "A" ($\frac{0.9}{1.6} > \frac{0.7}{1.6}$).

2.2.3 Dodatna stopnja primerjave

Zaradi množice objektov, kjer ima vsak množico značilk, se zgodi, da neka ključna lastnost ne pride do izraza. Dva znaka, ki sta sicer precej podobna, imata pa eno izstopajočo lastnost, sta tako pogosto klasificirana v isti razred. Ena od rešitev tega problema je uvedba dodatne stopnje primerjave, ki vsakič, ko se vhodni podatek klasificira v katerega od problematičnih razredov, pogleda točno določene značilke in njihove vrednosti.

Primer takega problematičnega razreda sta črki "A"in "I"(na sliki 2.9). Ker je razlika samo v iztegnjenem mezincu, ta lastnost med množico ostalih, enakih lastnosti ne pride do izraza, zaradi česar je črka "I" običajno klasificirana kot "A". Z uporabo te dodatne stopnje pa se vsakič, ko je nek znak klasificiran kot "A" preveri, koliko je objektov zelene barvne regije in ali je položaj enega izmed njih bistveno nad ostalimi objekti zelene, modre in rdeče regije. V kolikor se tak objekt odkrije, se klasifikacija spremeni iz razreda "A" v razred "I". Z uporabo te dodatne stopnje se klasifikacijska natančnost občutno izboljša.



Slika 2.9 Znak za črko "A" na levi in znak za črko "I" na desni.

3 Klasifikacija na podlagi predlog

3.1 Ujemanje s predlogo

Ujemanje s predlogo (angl. template matching) je postopek, ki zna na sliki poiskati območje, ki najbolj ustreza predlogi. Za izvedbo postopka tako potrebujemo dva vhodna podatka, sliko ter predlogo (angl. template), nato na sliki poiščemo območje, ki najbolj ustreza predlogi. Rezultat je korelacijska tabela, ki vsebuje podatke ujemanja slike s predlogo.

Algoritem deluje tako, da predloga "drsi" nad glavno sliko. Pri tem se za vsako lokacijo preveri podobnost slike in predloge, podobnost pa se zapiše v korelacijsko tabelo. Predloga se nato prestavi na novo mesto, na primer eno točko bolj desno. Postopek se ponavlja dokler niso primerjane vse možne kombinacije.

Na sliki 3.1 iščemo črko "R", ki je del logotipa Fakultete za računalništvo in informatiko. Predloga te črke je v zgornjem desnem kotu v okvirju. Predloga se postopoma premika proti desni dokler ne doseže desnega roba slike. Iskanje se nadaljuje na levi strani, eno vrstico nižje, dokler vsa slika ni preiskana.

Slika 3.1 Iskanje najboljšega ujemanja s predlogo.

Velikost korelacijske tabele je:

$$(h_\mathrm{i} - h_\mathrm{t} + 1) \times (w_\mathrm{i} - w_\mathrm{t} + 1), \tag{3.1}$$

kjer je h_i višina glavne slike, h_t višina predloge, w_i širina glavne slike in w_t širina predloge.

Tabela vsebuje vrednosti, ki predstavljajo stopnjo podobnosti, sam položaj znotraj tabele pa predstavlja položaj levega gornjega ogljišča predloge na glavni sliki. Vrednost, ki predstavlja podobnost slike s predlogo je odvisna od izbire metode za izračun. V knjižici OpenCV so na voljo tri metode, *metoda vsote kvadratov razlike* (angl. Square difference matching method), *metoda korelacijskega ujemanja* (angl. Correlation matching method) in *metoda korelacijskega koeficienta* (angl. Correlation coefficient matching method), vsaka pa ima možnost vračanja absolutnih ali normaliziranih vrednosti. V prvem primeru vrednost predstavlja razliko med sliko in predlogo, zato manjša vrednost pomeni večjo podobnost (oziroma vrednost 0 pomeni popolno ujemanje). V ostalih dveh primerih vrednost predstavlja ujemanje, kjer višja vrednost pomeni večje ujemanje [2].

Uporabili smo metodo razlike kvadratov, katere natančnost ni tako visoka, vendar je hitrost izračuna temu primerno višja. Formula za izračun podobnosti v določeni točki:

$$R_\mathrm{sqd}(x, y) = \sum_{x', y'} [T(x', y') - I(x + x', y + y')]^2 \tag{3.2}$$

Po končanem postopku iskanja podobnosti se v korelacijski tabeli poišče najvišjo vrednost (oziroma odvisno od izbire metode za izračun lahko tudi najnižjo vrednost). Na tistem mestu je bilo ujemanje slike in predloge najboljše.

Prednost uporabe ujemanja predlog je v preprosti implementaciji. Ob uporabi OpenCV knjižice uporaba algoritma zahteva le dodelitev prostora za korelacijsko tabelo in vhodne podatke. Verjetnost uspešnega zaznavanja pada obratno sorazmerno z razliko, kar smo podrobneje opisali v poglavju 3.5.

3.2 Rešitev 1

Za uporabo metode ujemanja vzorcev najprej potrebujemo predloge. Te dobimo po sledečem postopku: nad vsako sliko učne množice izvedemo barvno segmentacijo, podobno kot pri algoritmu 2.1.1. Ta nam vrne štiri binarne slike, eno za vsako barvno regijo. Te nato združimo v eno samo sliko, nad katero izvedemo analizo blobov, ki nam vrne seznam vseh objektov. Predpostavimo, da je največji objekt naš znak. Analiza blobov nam med drugim omogoča določitev najmanjšega pravokotnika, ki še v celoti zajame naš objekt. S pomočjo tega pravokotnika in lokacije objekta na sliki, lahko lokaliziramo predlogo na tisti del, kjer se nahaja znak.

Na ta način zagotovimo, da je predloga velika točno toliko, kolikor je najmanj potrebno za zajetje znaka in nič več. Tako se izognemo dodantemu procesiranju, ki je posledica predloge, ki zajema več kot je nujno potrebno.

Ko postopek ponovimo za vse znake v učni množici, imamo na voljo vse potrebne predloge. Nato izvedemo postopek iskanja najboljšega ujemanja s predlogo nad celotno vhodno sliko.

3.3 Rešitev 2

Rešitev 1, opisano v poglavju 3.2 smo v drugem koraku spremenili tako, da je velikost predloge enaka velikosti vhodne slike. Na ta način se zmanjša število primerjav in posledično pohitri algoritem, vendar pa se pojavi dodatna omejitev. Ker predloga ne "drsi" nad sliko, se mora znak nahajati na istem mestu na sliki in v predlogi.

3.4 Rešitev 3

Rešitev 1, opisano v poglavju 3.2 smo nazadnje nadgradili še tako, da se znak ne išče na celotni vhodni sliki, ampak smo ga predhodno poiskali po istem postopku, kot smo pri rešitvi 1 iskali znak na slikah učne množice, sedaj pa isti postopek uporabimo tudi za iskanje znaka na vhodni sliki. Namesto da ujemanje vzorcev izvajamo na celotni vhodni sliki, smo ga lokalizirali le na tisti del vhodne slike, kjer pričakujemo da se znak dejansko nahaja.

Tako ne iščemo mesta, kjer je ujemanje med predlogo in vhodno sliko največje, ampak nas zanima ujemanje samo na točno določenem mestu. Da to zagotovimo, morata biti

predloga in vhodna slika iste velikosti, kar dosežemo s prilagajanjem dimenzije predloge, tako da ustreza velikosti najdenega znaka na vhodni sliki.

Ta način je počasnejši od algoritma 3.2, ker vsebuje še dodatna koraka iskanja znaka na vhodni sliki in prilagajanja velikosti predloge, vendar pa je njegova natančnost zaradi prilagajanja dimenzije predloge posledično občutno višja.

3.5 Uspešnost zaznavanja pri odstopanju velikosti in rotacije

Natančnost zaznavanja pri velikih razlikah med velikostjo predloge in znaka na vhodni sliki smo testirali na sledeč način: izbrali smo si sliko črke iz učne množice, v našem primeru črko "L", zaradi njene nesimetričnosti, nato pa smo na podlagi te slike zgradili še predlogo. Predlogo smo potem postopoma zmanjševali in izračunali ujemanje. Rezultati so v tabeli 3.1.

Tabela 3.1 Uspešnost zaznavanja pri različni stopnji odstopanja velikosti.

Razlika v velikosti	Ocena podobnosti
0%	0.94
10%	0.73
25%	0.42
50%	0.17
75%	0.0

Poleg zmanjševanja smo predlogo tudi rotirali, vendar v njeni izvirni velikosti (razmerje velikosti predloge in značilke na sliki je bilo 1:1). Iz tabele 3.2 je razvidno da rotacija predloge močno vpliva na oceno podobnosti, saj je dosegla vrednost 0 že pri rotaciji za 60°.

Iz tabel 3.1 in 3.2 je tako razvidno, da že relativno majhna sprememba v rotaciji ali velikosti predloge (relativno na velikost znaka) povzroči občuten padec v uspešnosti zaznavanja.

Sam postopek iskanja najboljšega ujemanja s predlogo sicer ne ponuja možnosti za rešitev tega problema, lahko pa pred samo uporabo postopka izvedemo določene korake, s katerimi predlogo predhodno spremenimo in tako nekoliko izboljšamo natančnost.

Tak način uporabljamo pri rešitvi 3, opisani v poglavju 3.4, kjer najprej z analizo blobov poiščemo znak na sliki in njegove lastnosti, nato pa velikost predloge prilagodimo

Tabela 3.2 Uspešnost zaznavanja pri različni stopnji rotacije.

Razlika v rotaciji	Ocena podobnosti
0°	0.94
10°	0.71
20°	0.51
30°	0.39
45°	0.18
60°	0.0

velikosti znaka. Ta postopek lahko povzroči deformacijo predloge, kadar je sprememba velikosti različna po različnih oseh, zato ni splošno uporaben.

Ker nam analiza blobov ne razkrije dovolj podatkov o znaku, je ugotavljanje njegove rotacije težje. Ena od možnosti je, da znak poizkušamo segmentirati na manjše dele, nato pa s pomočjo njihovih središč določimo premico. Kot, ki ga oklepa ta premica in ena od koordinatnih osi nam namreč lahko namigne na rotacijo znaka na vhodni sliki. V kolikor poznamo še rotacijo znaka na predlogi, lahko sedaj predlogo po potrebi zarotiramo in tako zagotovimo višjo natančnost.

4 Testi in rezultati

4.1 Hitrost klasifikacije

Hitrost klasifikacije je bila testirana na podlagi klasifikacije šestindvajsetih znakov, po en znak za vsako črko abecede. Slika je bla zajeta v ločljivosti 10 MP (3648x2736 slikovnih elementov) nato pa pomanjšana oziroma povečana na ločljivosti zapisane v tabeli 4.1. Slike znakov so bile na voljo v trinajstih različnih ločljivostih. Predloge smo zgradili na podlagi teh istih slik.

Pri vsaki ločljivosti smo klasificirali vseh šestindvajset znakov, kot čas klasifikacije pri dani ločljivosti pa vzeli povprečje časov. Ti so zapisani v tabeli 4.1. V tabeli 4.2 pa so zapisane najhitrejše in najpočasnejše klasifikacije pri dani ločljivosti.

Rezultati kažejo na linearno zahtevnost algoritmov, ki temeljijo na iskanju s predlogo (3.2, 3.3 in 3.4). Tak rezultat je pričakovan, saj algoritmi preiščejo celotno sliko, čas potreben za to pa je odvisen od velikosti slike.

Odstopanja med njimi se pojavljajo zaradi razlik v delovanju. Vzrok v hitrosti algoritma 3.3 se skriva v dejstvu, da je tu predloga enako velika kot vhodna slika. Zato odpade potreba po premikanju predloge po sliki kar občutno zmanjša računske operacije.

Tabela 4.1 Hitrost klasifikacije, v sekundah. *Samo klasifikacija brez zajema značilk.

Ločljivost	3.2	3.3	3.4	2.2.1*	2.2.1	2.2.1+2.2.3	2.2.2	2.2.2+2.2.3
0.3 MP	0.25	0.24	0.23	0.002	0.19	0.20	0.18	0.19
0.5 MP	0.36	0.37	0.47	0.002	0.29	0.29	0.29	0.30
1 MP	0.87	0.87	1.07	0.002	0.67	0.6	0.67	0.69
2 MP	1.55	1.56	1.82	0.002	1.17	1.15	1.18	1.18
3 MP	2.45	2.45	2.95	0.003	1.86	1.85	1.87	1.88
4 MP	3.24	2.90	3.88	0.003	2.47	2.45	2.47	2.45
5 MP	4.15	3.67	4.91	0.003	3.11	3.11	3.12	3.16
6 MP	5.07	4.52	5.97	0.005	3.83	3.83	3.84	3.88
7 MP	6.19	5.37	7.06	0.005	4.52	4.56	4.54	4.57
8 MP	7.02	5.99	8.12	0.005	5.07	5.05	5.03	5.02
9 MP	7.64	6.82	8.65	0.006	5.64	5.64	5.64	5.97
10 MP	8.63	7.57	10.01	0.006	6.36	6.43	6.41	6.44
100 MP	98.83	83.07	113.12	0.009	71.70	71.42	72.87	72.75

Tabela 4.2 Odstopanje od povprečne hitrosti klasifikacije. Prva vrednost prikazuje najhitrejšo klasifikacijo, druga pa najpočasnejšo klasifikacijo.

Ločljivost	3.2	3.3	3.4	2.2.1	2.2.1+2.2.3	2.2.2	2.2.2+2.2.3
0.3 MP	0.24, 0.26	0.23, 0.25	0.22, 0.24	0.18, 0.2	0.19, 0.21	0.17, 0.19	0.18, 0.2
0.5 MP	0.35, 0.37	0.35, 0.39	0.45, 0.49	0.27, 0.31	0.28, 0.3	0.28, 0.3	0.29, 0.31
1 MP	0.84, 0.9	0.83, 0.91	1.03, 1.11	0.62, 0.72	0.57, 0.63	0.64, 0.7	0.67, 0.71
2 MP	1.5, 1.6	1.48, 1.64	1.75, 1.89	1.09, 1.25	1.09, 1.21	1.12, 1.24	1.14, 1.22
3 MP	2.38, 2.52	2.33, 2.57	2.83, 3.07	1.73, 1.99	1.76, 1.94	1.78, 1.96	1.82, 1.94
4 MP	3.14, 3.34	2.76, 3.05	3.72, 4.04	2.3, 2.64	2.33, 2.57	2.35, 2.59	2.38, 2.52
5 MP	4.03, 4.27	3.49, 3.85	4.71, 5.11	2.89, 3.33	2.95, 3.27	2.96, 3.28	3.07, 3.25
6 MP	4.92, 5.22	4.29, 4.75	5.73, 6.21	3.56, 4.1	3.64, 4.02	3.65, 4.03	3.76, 4
7 MP	6, 6.38	5.1, 5.64	6.78, 7.34	4.2, 4.84	4.33, 4.79	4.31, 4.77	4.43, 4.71
8 MP	6.81, 7.23	5.69, 6.29	7.8, 8.44	4.72, 5.42	4.8, 5.3	4.78, 5.28	4.87, 5.17
9 MP	7.41, 7.87	6.48, 7.16	8.3, 9	5.25, 6.03	5.36, 5.92	5.36, 5.92	5.79, 6.15
10 MP	8.37, 8.89	7.19, 7.95	9.61, 10.41	5.91, 6.81	6.11, 6.75	6.09, 6.73	6.25, 6.63

Drugi najboljši izmed algoritmov s predlogo (3.2) uporablja manjše predloge, ki so velikosti prostorskega obsega posameznega znaka, zato odpade dodatno procesiranje, ki je posledica prevelikih predlog. Ker obdelavo predlog izvedemo pred samo klasifikacijo, čas za njo ni zajet v tabeli 4.1. Razlika v času med algoritmoma 3.2 in 3.3 nastane zaradi potrebe po premikanju predloge nad glavno sliko, kar je posledica dejstva, da je predloga manjša kot vhodna slika.

Vzrok v najpočasnejši hitrosti algoritma s predlogo (3.4) pa je v analizi blobov. Ta se uporabi za iskanje znaka na vhodni sliki in nam omogoči primerjanje samo tistega dela vhodne slike kjer se znak nahaja. Zato tudi tu odpade potreba po premikanju predloge po sliki, vendar pa je število operacij, ki nam jih to prihrani manjše kot število operacij, ki jih potrebujemo za analizo blobov in prilagajanje velikosti predloge najdenemu znaku na vhodni sliki.

Zahtevnost te je vidna pri algoritmu 2.2.1 in njegovi izpeljanki 2.2.2. Algoritma uporabljata analizo blobov za zajem značilk, nato pa na podlagi numeričnih predstavitev teh značilk izračunata najboljše ujemanje med vhodnimi znaki in znaki v učni množici. Prvi stolpec v skupini zadnjih petih prikazuje trajanje samo izračuna razlike ujemanja z znakom iz učne množice, iz katerega je razvidno, da je trajanje izračuna zanemarljivo majhno, skupni čas klasifikacije pa narašča linearno z velikostjo slike.

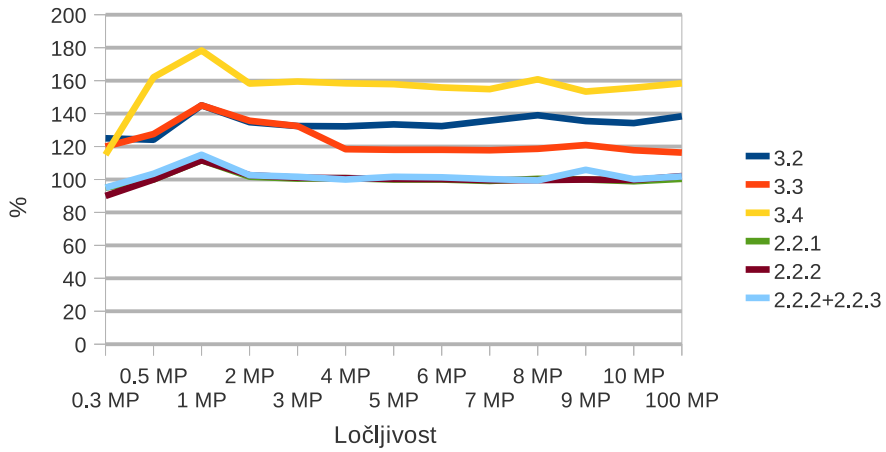
Graf 4.1 prikazuje hitrost algoritmov glede na najhitrejši algoritem, opisan v poglavju 2.2.1 z uporabo dodatne stopnje klasifikacije, opisane v poglavju 2.2.3. Odstotki predstavljajo odstotek porabljenega časa glede na najhitrejši algoritem.

4.2 Natančnost klasifikacije

Natančnost klasifikacije smo testirali na dva načina, in sicer s slikami in videom. Testiranje natančnosti na podlagi videa smo razdelili še na dva dela. V prvem smo testirali video, sestavljen iz slik, v drugem pa na živem videu, zajetem s spletno kamero.

4.2.1 Natančnost klasifikacije na podlagi slik

Natančnost klasifikacije smo najprej testirali na podlagi množice posameznih slik. Vsaka slika je na temnem ozadju vsebovala en znak, in je predstavljala eno črko ameriške prstne abecede. Sto dvainpetdeset slik je bilo razdeljenih v tri testne množice. Prva testna množica je vsebovala petnajst slik, druga oseminštirideset slik, tretja pa sto sedemintrideset slik. Druga množica je bila podmnožica tretje testne množice, vsebovala pa je tiste



Slika 4.1 Hitrost klasifikacije v odstotkih glede na najhitrejši algoritem.

primere, ki so bili bolj podobni učni množici.

Vsak klasifikator smo pognali nad vsako od treh testnih množic. Rezultati so v tabeli 4.3. Najbolj natančen je algoritem 2.2.1 v kombinaciji z dodatno plastjo klasifikacije (2.2.3). Poleg visoke klasifikacijske natančnosti pa je ta algoritem tudi najhitrejši, kar je vidno na sliki 4.2 in je zato izmed vseh opisanih najprimernejši za uporabo.

Dokaj visoko natančnost je dosegel tudi algoritem 3.4. Do razlike med njegovo natančnostjo in natančnostjo ostalih dveh algoritmov, ki temeljita na ujemanju s predlogo (3.2 in 3.3) pride zaradi dodatnega koraka analize blobov, ki na vhodni sliki lokalizira območje znaka, nato pa velikost predloge prilagodi velikosti tega dela vhodne slike. Zaradi spreminjanja velikosti predloge je ta algoritem bistveno manj občutljiv na razlike v velikosti med vhodno sliko in (izvirno, nespremenjeno) predlogo, kar je sicer ena izmed pomankljivosti algoritmov ki uporabljajo ujemanje predlog.

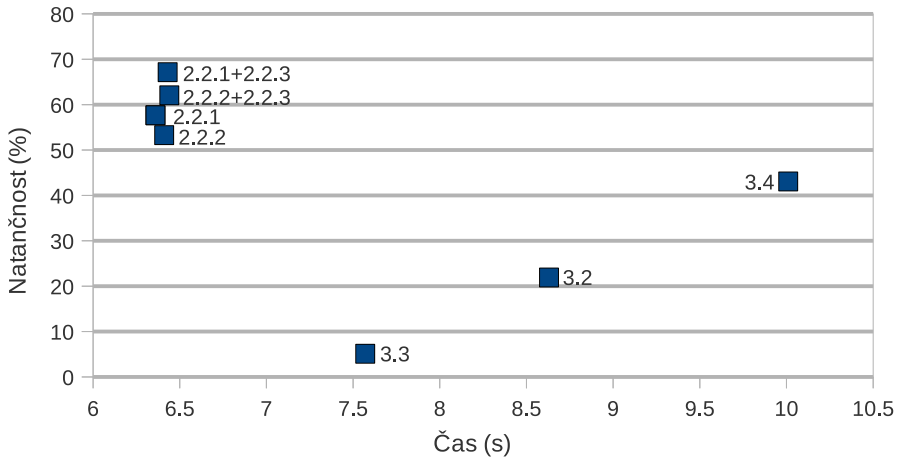
Imajo pa algoritmi ujemanja s predlogo eno bistveno prednost pred algoritmi, ki temeljijo na zajemu in primerjavi posameznih značilk (2.2.1 in 2.2.2). Slednji potrebujejo značilke posameznih barvnih regij, kar dobijo s segmentacijo na podlagi barvnih filtrov. Ker takšna barvna segmentacija ni možna na podlagi naravnih barv človeške roke, se za to uporablja rokavica. Iskanje ujemanja s predlogo segmentacije ne potrebuje[1], ker ne računa ujemanja posameznih delov roke, pač pa ujemanje roke kot celote. Zadostoval bi le barvni filter, ki bi na vhodni sliki poiskal dele, ki so iste barve kot človeška koža ter

[1] V poglavju 3.2 in 3.4 smo barvno segmentacijo na podlagi štirih barvnih regij uporabili zato, da smo testiranje lahko izvedli na isti testni množici, ki je vsebovala slike znakov z uporabo barvne rokavice.

Tabela 4.3 Natančnost pri klasificiranju slik.

	Učna množica (število primerov)		
Klasifikator	Mala (15)	Srednja (48)	Velika (137)
2.2.1	73.34 %	62.5 %	57.66 %
2.2.1 + 2.2.3	86.67 %	89.58 %	67.15 %
2.2.2	46.67 %	54.17 %	53.28 %
2.2.2 + 2.2.3	60.00 %	77.08 %	62.05 %
3.2	46.67 %	18.75 %	21.90 %
3.3	6.67 %	4.17 %	5.11 %
3.4	73.34 %	56.25 %	43.07 %

izvedel iskanje najboljšega ujemanja s predlogo.



Slika 4.2 Natančnost in hitrost pri klasificiranju slik.

4.2.2 Natančnost klasifikacije na podlagi videa

Natančnost klasifikacije v primeru videa smo testirali na dva načina. Ker je nemogoče, da bi bila večkratna črkovanja popolnoma enaka, samo testiranje izvedli na predhodno pripravljenih posnetkih. Tako je bila edina sprememba med testiranji sprememba klasifikacijskega algoritma.

Pripravili smo dva posnetka. Prvi je bil sestavljen iz slik, ki smo jih uporabili za pridobitev baze znanja in testiranje natančnosti klasifikacije opisane v poglavju 4.2.1.

Tabela 4.4 Klasifikacijska natančnost (video); video1 označuje video posnetek sestavljen iz slik, video2 pa je zajet s spletno kamero. Pravilnost je izražena v odstotkih pravilnih klasifikacij.

Klasifikator	video1	video2
2.2.1 + 2.2.3	62.5 %	50 %
2.2.2 + 2.2.3	42 %	28.5 %
3.2	25 %	15.4 %
3.3	17 %	0 %
3.4	56.25 %	34.6 %

Znaki v tem videu so bili enakomerno osvetljeni, z minimalnim šumom in z enakomernim ozadjem.

V drugem videu smo uporabili posnetek zajet s spletno kamero. Ta je bil bistveno bolj šumen, s slabšo osvetlitvijo. Posnetek je tako bližje realnim razmeram, kjer je osvetlitev pogosto slaba. Rezultati testiranja so v tabeli 4.4.

Po pričakovanjih je natančnost na prvem (umetnem) posnetku približno enaka natančnosti v poglavju 4.2.1, kar je posledica uporabe istih testnih slik, le da so tokrat v obliki video posnetka.

Rezultat klasifikacije na drugem posnetku pa je bistveno slabši. Razloge za to in možne rešitve si bomo podrobneje ogledali v nadaljevanju.

4.2.3 Prehodi med znaki

Eden od vzrokov, ki znižuje natančnost pri živem posnetku so prehodi med znaki. Ker je prvi posnetek sestavljen iz slik, med znaki ni pravega prehoda, ki se zgodi v resnici; kakor tudi ni prisotnega gibanja roke. Kot primer vzemimo znak “A”, ki se nato spremeni v znak “B”. Nekje med položajem roke, ki predstavlja oba končna znaka je položaj, ki je podoben znaku “C”. V kolikor klasificiramo v tisti razred, katerega znak je trenutno na sliki, ti dodatni “prehodni” znaki predstavljajo problem.

Rešitev je v upoštevanju zgodovine predhodnih znakov, namesto samo trenutno prikazanega. Konkretno smo uporabili pristop s tekočim povprečjem. Za vseh šestindvajset znakov hranimo zgodovino ocen zadnjih n-klasifikacij. Daljša je zgodovina ocen, manj občutljiv je klasifikator na prehode, hkrati pa mora biti znak dalj časa viden na sliki, da ga algoritem zazna kot dejanski znak in ne kot prehod.

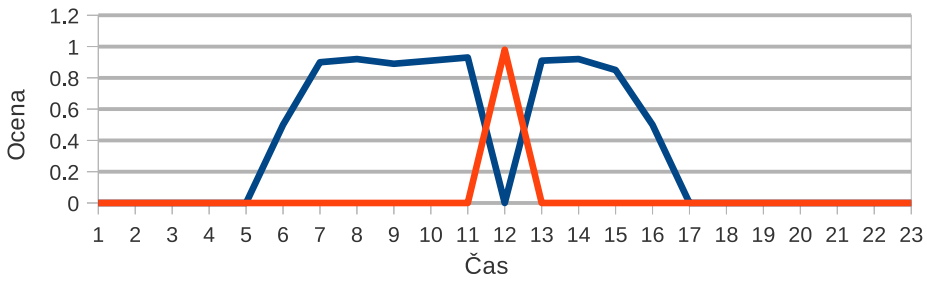
Ko se na vhodu pojavi nov znak, mu s pomočjo klasifikatorja določimo razred in oceno

ujemanja. Nato pa namesto da znak v ta razred tudi klasificiramo, v njegovo zgodovino dodamo oceno in izračunamo tekoče povprečje celotne zgodovine.

Znaku, ki ga je klasifikator prepoznal na sliki v zgodovino dodamo njegovo oceno, ostalim znakom pa lahko kot oceno vpišemo 0. Druga možnost je, da vsem znakom vpišemo oceno ujemanja. V vsakem primeru sledi še izračun tekočega povprečja za vse znake.

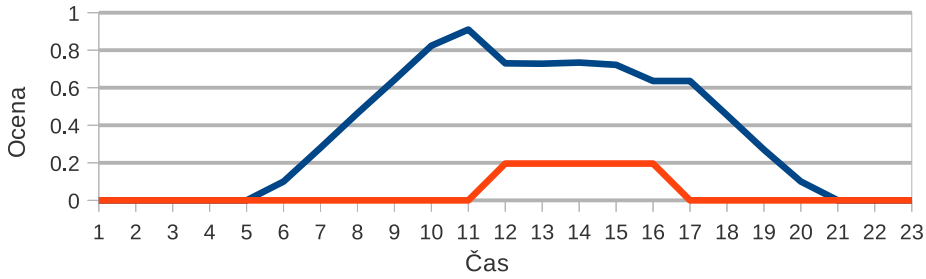
Izkaže se, da druga možnosti ni najbolj primerna. Klasifikator vrača ocene vseh črk v abecedi in najvišja ocena je pogosto le malenkost višja od druge najvišje ocene, ki je le malenkost višja od tretje najvišje, itd. Posledica je prepočasno padanje povprečja.

Graf 4.3 prikazuje oceno znaka v odvisnosti od časa. Na istem grafu sta oceni za znaka “A” in “B”. Čeprav je bil od pete do šestnajste slike viden le znak “A”, je zaradi zunanjih vplivov (šuma na sliki, sprememba osvetljave) na dvanajsti sliki prepoznan znak “B”. Prepoznano zaporedje znakov je torej “ABA”.



Slika 4.3 Graf zaznavanja. Modra krivulja označuje oceno za znak “A”, rdeča ša za znam “B”.

Na grafu 4.4 smo uporabili tekoče povprečje. Na ta način smo graf nekoliko zgladili. Posledica so manj izraziti ekstremi in nižji maksimum ter daljše trajanje znaka. Posledično ni več prisoten pulz, kjer bi bile ocene znaka “B” večje od ocene znaka “A” in se celotno zaporedje znakov razpozna kot ena sama črka “A”.



Slika 4.4 Graf zaznavanja po uporabi tekočega povprečja. Modra krivulja označuje oceno za znak "A", rdeča pa za znak "B".

Slika 4.5 prikazuje postopek črkovanja z upoštevanjem prehodov med znaki. Zgornji del prikazuje dvajset zajetih slik, srednji graf njihove ocene, spodnji graf pa tekoče povprečje zadnjih treh ocen. Iz spodnjega grafa je razvidno, da se ocena znakov pojavi z zamikom ene slike. Ker je bil znak za "D" (na dvanajsti sliki) viden samo na eni sliki, je njegovo povprečje manjše od povprečja znaka za "B", zato se "D" v končnem zaporedju znakov ne pojavi. Končno razpoznano zaporedje je torej: "ABC".

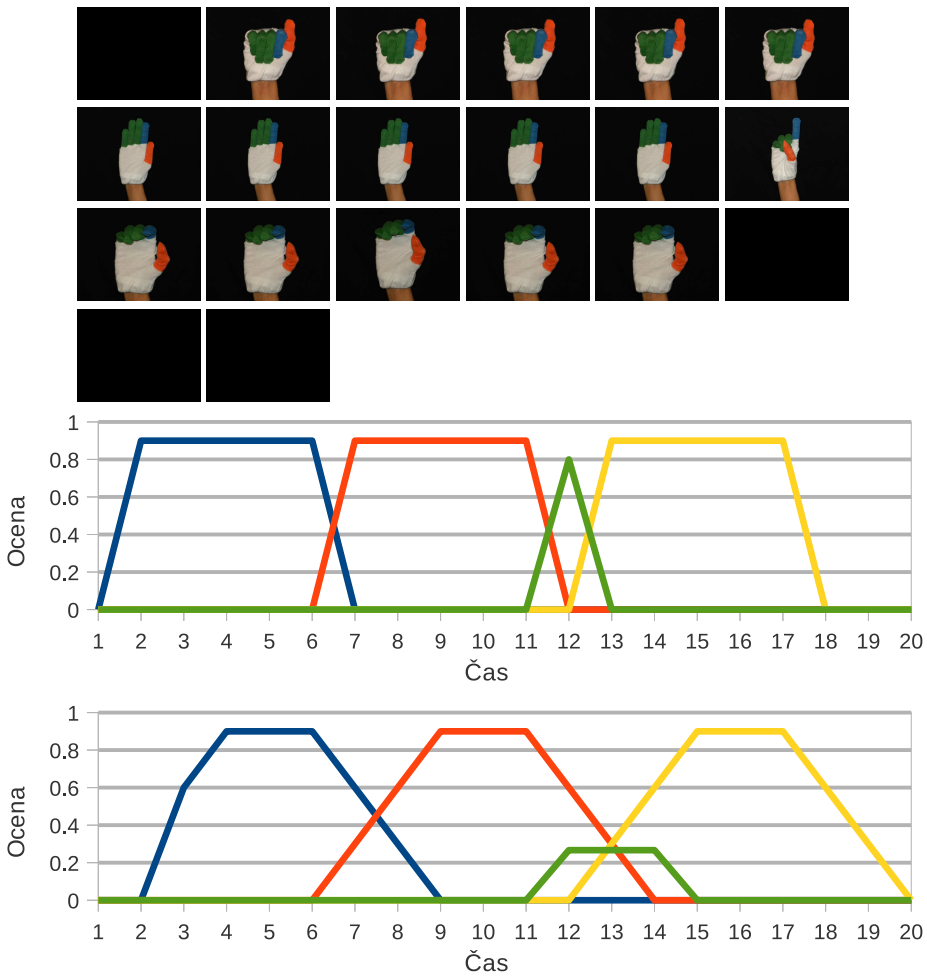
4.2.4 Šum

V videu, ki ga zajamejo spletne kamere je pogosto prisoten šum. Zgodi se, da tudi kadar je na sliki isti znak daljši čas, ga klasifikator zaradi šuma občasno prepozna kot nek drug znak. Delno se ta problem rešuje z uporabo filtra z mediano, ki skuša odpraviti šum. Res pa je tudi, da kadar je napačna klasifikacija posledica šuma, je ta navadno prisotna le krajši čas (pogosto samo eno sliko), kot pri napačnih klasifikacijah zaradi prehoda med znaki, zato ta problem odpravlja uporaba tekočega povprečja.

Natančnost klasifikacije ob prisotnosti šuma

Natančnost klasifikacije ob prisotnosti šuma smo testirali na sledeč način: izbrali smo množico slik, katero smo predhodno testirali in se prepričali, da so znaki pravilmo klasificirani, kadar šum ni prisoten. Nato smo na sliko nanesli šum v obliki naključno obarvanih pik, velikosti enega slikovnega elementa, in klasifikacijo ponovili. Rezultat prikazuje tabela 4.5.

Na šum najbolj neobčutljiv je algoritem opisan v poglavju 3.4, najbolj občutljiv pa algoritem opisan v poglavju 3.3. Algoritem najbližjih sosedov (poglavje 2.2.1), ki je v idealnih razmerah najbolj točen, je uporaben le kadar so prisotne zmerne količine šuma.



Slika 4.5 Črkovanje; modri krivulji označujeta oceno za znak "A", rdeči za znak "B", rumeni znak "C", zeleni pa znak "D".

Filter z mediano

Filter z mediano je pogosto uporabljena metoda za zmanjševanje šuma na slikah. Osnovna ideja algoritma je, da se vrednost vsakega slikovnega elementa nadomesti z mediano njegovih sosedov. Pri slikah se sosede določi s pomočjo "okna", ki je poljubne oblike in velikosti. Navadno se za okno uporabi kvadrat z liho dolžino stranice. Tako lahko preprosto določimo slikovni element, ki je točno v središču okna (recimo mu e_s). Vrednosti slikovnih elementov znotraj okna se nato uredijo po velikosti. Vrednost točno na sredini urejenega seznama (mediana) določa novo vrednost slikovnega elementa e_s. Okno se nato premakne na nov položaj in postopek se ponovi.

Na sliki 4.6 je zgoraj slika znaka "L" z dodanim šumom. Spodnja slika je rezultat

Tabela 4.5 Klasifikacijska natančnost na šumnih slikah; šum predstavlja odstotek slike, prekrite s šumnimi slikovnimi elementi.

Šum	3.2	3.3	3.4	2.2.1+2.2.3	2.2.2+2.2.3
1%	33 %	0 %	100 %	66 %	33 %
2%	33 %	0 %	100 %	0 %	33 %
3%	33 %	0 %	100 %	0 %	33 %
5%	33 %	0 %	66 %	0 %	0 %
7%	0 %	0 %	0 %	0 %	0 %

filtracije z mediano.

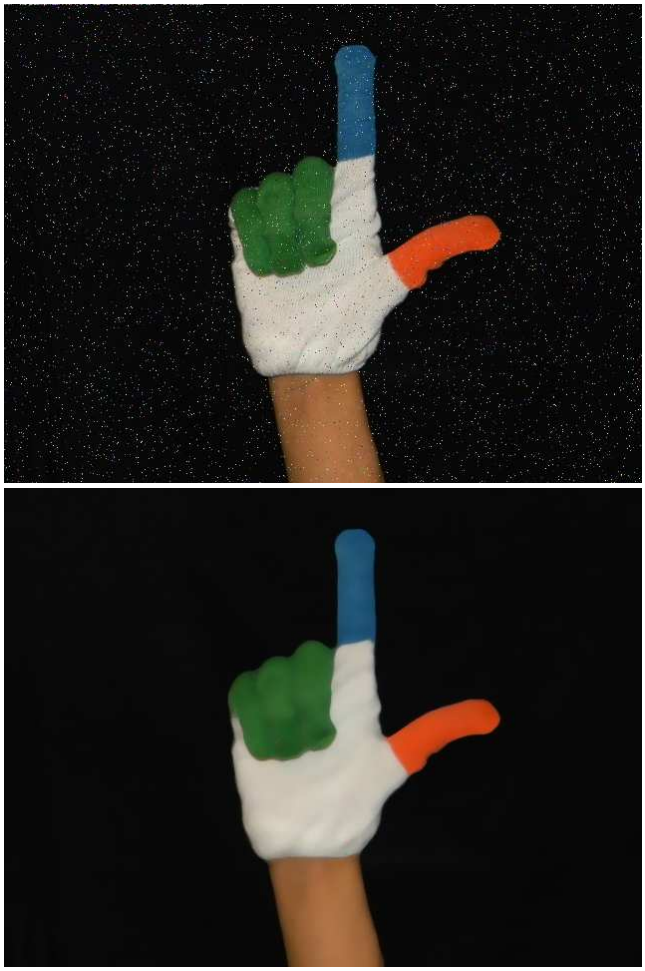
4.2.5 Osvetljava

Zajem značilk iz vhodne slike se zanaša na segmentacijo na podlagi barvnih filtrov. Čeprav se slike takoj po zajemu pretvorijo v HSV barvni prostor, ki je manj občutljiv na spremembo osvetljenosti, prevelika sprememba povzroči preveliko odstopanje vrednosti slikovnih elementov, ki nato ne ustrezajo filtrom. Filtre je zato po vsaki spremembi osvetlitve potrebno ponovno nastaviti.

Težava lahko nastane tudi zaradi avtomatskih prilagoditev strojne opreme. Nekatere spletne kamere same prilagajajo osvetljenost, barvni kontrast, itd. Vse te spremembe vplivajo na vrednosti slikovnih elementov (barve se navidezno spremenijo) in posledično na njihovo ustreznost barvnim filtrom.

Med testiranjem natančnosti na živem videu (zajetem s spletno kamero) smo naleteli na sledečo težavo: aplikacija je ustrezno klasificirala znak za črko “A” z oceno 0.9, natančnost naslednjega znaka (črka “B”) pa je bila 0.0. Težavo smo odkrili v avtomatskem prilagajanju nastavitev spletne kamere. Znak za črko “B” ima večjo površino bele barvne regije kot znak za “A”, zaradi česar je kamera avtomatsko prilagodila osvetljenost. Ker barvne regije niso več ustrezale barvnim filtrom, so bile rezultat segmentacije (poglavje 2.1.1) štiri slike, na katerih je bilo samo ozadje.

Težavo smo odpravili z izključitvijo avtomatskega prilagajanja strojne opreme. Za prilagajanje skrbijo gonilniki naprave in vsi ne omogočajo izklopa. Na voljo smo imeli dve spletni kameri, Philips PCVC675K in Logitech QuickCam Communicate STX. Samo Logitech kamera je podpirala izklop avtomatskega prilagajanja.



Slika 4.6 Filtriranje z mediano.

4.2.6 Črkovanje besed

Do tega trenutka je bil naš cilj prepoznavanje posameznih znakov na sliki, končni cilj pa je črkovanje celotnih besed. Naivna rešitev je dodajanje vsakega novega znaka na konec spremenljivke, kjer hranimo celotno besedo.

Klasifikator vrne prepoznano črko za vsako zajeto sliko. Kot je razvidno iz tabele 4.1, zajamemo novo sliko pribljižno vsake 0.2 sekunde. Taka hitrost zajema utegne biti prehitra za povprečnega črkovalca, kar pomeni, da bo en znak viden na več zaporedno zajetih slikah. Težavo rešujemo tako, da upoštevamo predhodni znak. Le kadar sta trenutni in predhodni znak različna, trenutni znak dodamo na konec spremenljivke z besedo.

Pojavi se nova težava. Nekatere besede imajo podvojene črke (na primer beseda “oddaja”). Takšne besede so v slovenščini sicer redke, so pa precej pogoste v angleščini. Upoštevajoč četrto pravilo poglavja 1.3 take črke ločimo s poskokom roke. V poglavju 2.1.3 smo opisali značilke objektov, ki jih beležimo. Ena takih je tudi središče znaka. Središče predhodnega znaka zatorej shranjujemo. V kolikor oddaljenost središča trenutnega znaka od predhodnega znaka preseže nek prag, ignoriramo pravilo v prejšnjem odstavku in črko dodamo na konec spremenljivke z besedo.

Besedo zaključimo tako, da zadnji znak ne spremenimo določen čas t_z. Ta čas določimo tako, da beležimo trajanje posameznega znaka v besedi ter izračunamo povprečno trajanje znakov v besedi, pri čemer ne upoštevamo zadnjega znaka. Povprečno trajanje pomnožimo z dve in dobimo t_z.

5 Zaključek

5.1 Sklepi in ugotovitve

V diplomski nalogi smo analizirali metode za prepoznavanje znakov Ameriške prstne abecede na posameznih slikah in v videu. V prvem delu smo opisali algoritme, ki klasificirajo na podlagi značilk, v drugem delu pa algoritme, ki klasificirajo na podlagi predlog. Pri testiranju smo ugotovili, da je najbolj primeren algoritem, ki z uporabo metode najbližjih sosedov znak klasificira v isti razred, kateremu pripada najbližji sosed. Najprimernejši je po obeh testiranih kriterijih, to sta hitrost in natančnost.

Pri testiranju je izstopala še tretja implementacija algoritma, ki klasificira na podlagi ujemanja s predlogo. Kljub slabši natančnosti je algoritem potencialno uporabnejši, saj ne zahteva uporabe rokavice.

Opisali smo tudi različne težave, ki otežujejo postopek prepoznavanja znakov. Te so lahko posledica slabe kvalitete slike, zajete s spletno kamero, na primer slaba osvetlitev in šum. Lahko pa so posledica same narave črkovanja s prstno abecedo. Primer tega so prehodi med znaki, ki so podobni drugim znakom.

Problem prehoda med znaki smo zmanjšali do sprejemljive ravni z uporabo tekočega

povprečja, šum pa z uporabo filtriranja z mediano. Problemov, povezanih s slabo osvetlitvijo nam ni uspelo odpraviti.

5.2 Nadaljnje delo

V poglavju 2.2.1 smo znake klasificirali z uporabo metode najbližjega soseda oziroma n-najbližjih sosedov. Izboljšave bi bilo morda moč doseči z uporabo kakšnega drugega klasifikatorja, na primer odločitvenih dreves, Bayesovega klasifikatorja, umetnih nevronskih mrež, mehke logike ali pa z uporabo hibridnega algoritma, kjer bi kombinirali več klasifikatorjev in se tako izognili slabostim posameznih algoritmov [3].

Sprememba osvetljenosti kadra predstavlja problem. Za pravilno segmentacijo slike potrebujemo barvne filtre, pri spremembi osvetljenosti pa se spremenijo vrednosti posameznih slikovnih elementov. Kadar ta sprememba preseže nek določen prag, slike ne moremo več pravilno segmentirati. Implementacija avtomatske ali pa ročne kalibracije bi to težavo odpravila. Pri ročni kalibraciji bi uporabnik s klikom na posamezne barvne regije nastavljal filtre. Na ta način bi odpravili slabe segmentacije, kadar bi svetloba dalj časa ostala konstantna.

Pri pogostem spreminjanju svetlobe pa bi bila potrebna avtomatska kalibracija. To bi bilo moč doseči z uporabo analize histogramov. Za uporabo tega pristopa pa bi potrebovali enakomerno ozadje.

LITERATURA

[1] F. Chang, C-J. Chen, and C-J. Lu, "A Linear-Time Component-Labeling Algorithm Using Contour Tracing Technique", Computer Vision and Image Understanding 93 (2) (2004) 206–220.

[2] G. Bradski, A. Kaehler, "Learning OpenCV, Computer Vision with the OpenCV Library", O'Reilly Media, Inc. (2008) 214–218.

[3] I. Kononenko, "Strojno učenje", ZALOŽBA FE in FRI (2005) 2–5.