Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Jagodnik

Detekcija potnikov v avtomobilu

diplomsko delo
visokošolski strokovni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Peter Peer
Asistent: mag. Jure Kovač

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00230/2012
Datum: 02.04.2012

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

Kandidat: **TADEJ JAGODNIK**

Naslov: **DETEKCIJA POTNIKOV V AVTOMOBILU**
DETECTION OF PASSENGERS IN A CAR

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Raziščite uspešnost, primernost in zanesljivost delovanja najpomembnejših algoritmov za detekcijo obrazov v avtomobilu. Testiranje algoritmov izvedite na video posnetkih zajetih z uporabo širokokotne digitalne kamere, nameščene v notranjosti avtomobila. Video posnetke anotirajte tako, da s pravokotniki označite obraze na slikah video posnetkov. Implementirajte sistem za ocenjevanje uspešnosti algoritmov, ki primerja rezultate algoritmov z rezultati anotacije. Ugotovite, pri katerih vrednostih parametrov dajo ti algoritmi najboljše rezultate.

Mentor:
doc. dr. Peter Peer

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
UNIVERZA V LJUBLJANI
1

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tadej Jagodnik, z vpisno številko **63070309**, sem avtor diplomskega dela z naslovom:

Detekcija potnikov v avtomobilu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Petra Peera in mag. Jureta Kovača,
- 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 5. julija 2012 Podpis avtorja:

Iskreno se zahvaljujem staršem, ker so mi omogočili študij in me skozi vsa leta podpirali.

Zahvaljujem se doc. dr. Petru Peeru in mag. Juretu Kovaču za mentorstvo, potrpežljivost, strokovno pomoč, nasvete ter vodenje pri pisanju diplomskega dela.

Zahvaljujem se tudi mag. Nataši Ujčič, univ. dipl. slov. in rus. za lektoriranje diplomskega dela.

Hvala sestri Tjaši in dekletu Ani, ki sta mi ves čas stali ob strani.

Kazalo

Seznam uporabljenih kratic

OpenCV - Odprtokodna knjižnica za računalniški vid (angl. *Open Source Computer Vision*)
SMQT - Transformacija, namenjena izboljšanju in stiskanju slike (angl. *The Successive Mean Quantization Transform*)
SNoW - Redka mreža linearnih funkcij, ki uporabljajo Window Update Rule (angl. *Sparse Network of Windows*)
SVM - Metoda podpornih vektorjev, ki se uporablja za klasifikacijo (angl. *Support Vector Machines*)
RSV - Manjša množica podpornih vektorjev (angl. *Reduce Set Vectors*)
SVM - Singularni razcep za faktorizacijo matrik (angl. *Singular Value Decomposition*)
HD - Format videa visoke ločljivosti (angl. *High Definition*)
TP - Resnično pozitiven (angl. *True Positive*)
FD - Vsota napačnih detekcij (lažno pozitivni in lažno negativni skupaj) (angl. *False Detection*)
FP - Lažno pozitiven (angl. *False Positive*)
FN - Lažno negativen (angl. *False Negative*)

Povzetek

V diplomskem delu se osredotočimo na raziskavo uspešnosti, primernosti in zanesljivosti delovanja nekaterih že razvitih algoritmov za detekcijo obrazov v okolju, kot je avtomobil. Notranjost avtomobila je izpostavljena različnim vplivom osvetlitve, ki otežujejo detekcijo. Problem predstavlja tudi utesnjen prostor, v katerem lahko pride do delnih prekrivanj obrazov. V prvem delu predstavimo tri različne pristope za detekcijo obrazov ter se sproti seznanimo z uporabljenimi orodji. Za izboljšanje delovanja in zmanjšanje števila napačnih detekcij predlagamo omejitev iskanja, tako da razdelimo sliko na dve liniji ter pet kvadratov. Testiranje algoritmov izvedemo na videoposnetkih, zajetih z uporabo širokokotne digitalne kamere, nameščene v notranjosti avtomobila. Videoposnetke anotiramo z uporabo implementiranega orodja, tako da s pravokotniki označimo obraze na slikah videoposnetkov. Implementiramo sistem za ocenjevanje, ki primerja rezultate algoritmov z rezultati anotacije. Na podlagi ocenjevanja izvedemo različne teste ter s spreminjanjem parametrov posameznih algoritmov skušamo ugotoviti, pri katerih vrednostih ti najboljše delujejo.

Ključne besede:
računalniški vid, detekcija obrazov, avtomobil

Abstract

This thesis is focused on the research of the performance, suitability and reliability of some developed algorithms for face detection in the in-car environment. The vehicle interior is exposed to various illumination conditions which can complicate detection. Cramped space can also represent a problem, since it can cause occlusion. First we present three different approaches for face detection. To improve algorithm's performance and reduce the number of false detections, we suggest splitting frames or search window to two lines and five rectangles. We test algorithms on videos, recorded with wide-angle digital camera and annotate them by bounding boxes with implemented tool. System for evaluation is implemented to compare the algorithm's results with the results of ground truth annotation. Based on the evaluation system we perform various tests and with changing algorithm's parameters we try to determine their best performance.

Key words:
computer vision, face detection, in-car environment

Poglavje 1

Uvod

1.1 Računalniški vid in detekcija obrazov

Področje računalniškega vida (angl. *computer vision*) je v zadnjih letih močno napredovalo in postalo pomemben del našega vsakdanjika. Razlog se nahaja predvsem v hitrem tehnološkem napredku in vse zmogljivejših računalnikih. Področje je najpogosteje povezano z umetno inteligenco, saj je njegov cilj reševati kompleksne naloge s pomočjo računalnikov, kot so obdelava, analiza in interpretacija vizualnih informacij. Te je sposoben reševati tudi človek sam, vendar računalnik delo opravi hitreje, natančneje in predvsem ceneje. Med pomembnejše in pogostejše naloge računalniškega vida spada detekcija obrazov (angl. *face detection*). Uvrščamo jo med naloge razpoznave objektov, ki ugotavljajo prisotnost različnih objektov na slikah ter njihovo 3D-lokacijo. Torej je detekcija obrazov naloga, ki ugotavlja prisotnost obrazov na slikah in njihovo lokacijo. Ker je obraz prirojena biometrična značilnost človeka in so si ti po strukturi precej podobni, je detekcija obrazov relativno enostavna naloga. Detekcija obrazov je prvi korak vsakega sistema, ki analizira obrazne informacije, kot so sistemi za prepoznavanje obrazov ter za nadzor in varnost [13].

1.2 Predstavitev problema

V diplomskem delu obravnavamo detekcijo ljudi oziroma potnikov v avtomobilu, ki temelji na detekciji obrazov. Notranjost avtomobila je prostor, ki je ponoči in tudi čez dan močno izpostavljen različnim vplivom osvetlitve (angl. *illumination*) od zunaj (sončna svetloba, luči javne razsvetljave, luči drugih avtomobilov itd.). To predstavlja problem, saj moteča in kompleksna ozadja otežujejo detekcijo in povečujejo število napačnih detekcij. Problem avtomobila je tudi v tem, da je njegova notranjost močno utesnjena, kar lahko povzroči delna prekrivanja obrazov, saj lahko potnikom, ki sedijo zadaj, obraze prekrivajo prednji sedeži.

Namen diplomskega dela je raziskati primernost uporabe detekcije obrazov v okolju, kot je avtomobil, hkrati pa tudi ugotoviti uspešnost in zanesljivost njenega delovanja. Osredotočimo se predvsem na raziskavo nekaterih že razvitih algoritmov in metod za detekcijo obrazov. Cilj je testirati njihovo delovanje v avtomobilu ob dnevni in nočni svetlobi, pri premikanju in mirovanju vozila ter prilagoditi njihovo delovanje. Za vsak algoritem želimo poiskati najustreznejše vrednosti parametrov, pri katerih je algoritem najbolj zmogljiv. Z omejitvijo slike oziroma iskalnega okna skušamo izboljšati delovanje algoritma in zmanjšati število napačnih detekcij.

Raziskava algoritmov za detekcijo obrazov je namenjena sistemu za zaznavo ljudi, primernemu na primer za taksi službe. Sistem služi nadzoru potnikov, ugotavljanju njihovega števila in zagotavljanju njihove varnosti. Dogajanje v avtomobilu spremlja kamera, usmerjena tako, da zajema obraze vseh potnikov. Osnova sistema je algoritem za detekcijo obrazov, z njim namreč pridobimo informacije o dogajanju v avtomobilu.

1.3 Pregled področja s sorodnimi deli

Aplikacije, ki temeljijo na umetni inteligenci, so prisotne na različnih področjih, kot so medicina, varnost ter vojaška, prehrambna, farmacevtska in avtomobilska industrija itd. Slednja je povezana z računalniškim vidom na

področju varnosti in nadzora. Napredek je spodbudil raziskavi o uporabi sistemov za različne detekcije v avtomobilih.

Prva predstavlja sistem za detekcijo potnikov, namenjen pomoči vozniku, tako da spremlja potnike v avtomobilu [15]. Uporablja se 360°-kamera, nameščena na stropu v notranjosti avtomobila. Sistem je sposoben na sliki, predstavljeni v ptičji perspektivi, zaznati potnikovo glavo (obraz, profil in zadnji del glave). Uporablja se algoritem za detekcijo objektov Viola-Jones, na podlagi katerega je zgrajen klasifikator za detekcijo glave. Sistem omogoča določanje števila potnikov v avtomobilu in njihove položaje.

Detekcija obrazov je tudi osnova varnostnega sistema, ki zaznava voznikov obraz in ga primerja s prej definiranim obrazom, ki je namenjen preprečevanju kraje avtomobila [8]. Algoritem za detekcijo obrazov je uporabljen za določitev, lokacije obraza. Tudi ta sistem uporablja kaskadni algoritem, ki sta ga predlagala Paul Viola in Michael Jones.

Sistemi za detekcijo obrazov, ki se uporabljajo v zahtevnih okoljih, v večini primerov uporabljajo algoritme, ki ne temeljijo na iskanju človeške kože (barva). Notranjost avtomobila je ena izmed kompleksnejših okolij, ki jo spremljajo številne svetlobne spremembe, zato smo se za primerjavo algoritmov Viola-Jones [14], Nilsson et al [11] in Kienzle et al [7] odločili, ker algoritmi temeljijo predvsem na klasifikaciji in ne na iskanju barve kože [5]. Dodaten razlog za primerjanje algoritmov je tudi njihova pogosta uporaba (predvsem algoritem Viola-Jones).

1.4 Pregled diplomskega dela

Za detekcijo obrazov so znane različne metode, med njimi tudi že razviti algoritmi, kot so Viola-Jones [14], Nilsson et al [11] in Kienzle et al (Fdlib) [7]. Cilj diplomskega dela je ugotoviti primernost in zanesljivost uporabe detekcije obrazov v avtomobilu ter spoznati, kateri izmed navedenih algoritmov je najustreznejši za detekcijo potnikov v avtomobilu. To skušamo ugotoviti v različnih okoliščinah, kot sta vožnja in mirovanje vozila ob različnih

osvetlitvah in pri spreminjanju števila potnikov.

V prvem poglavju opredelimo osnovne koncepte in delovanje uporabljenih algoritmov za detekcijo obrazov. Razložimo pomen parametrov, ki jih sprejmejo posamezne funkcije za detekcijo ter opišemo uporabljena orodja. Nato predstavimo implementacije, uporabljene pri našem delu in njihove razširitve. V to poglavje vključimo tudi predloge za izboljšanje delovanja algoritmov. V nadeljevanju predstavimo zajem in vsebino videoposnetkov, ki jih kasneje uporabimo za testiranje delovanja algoritmov. Opišemo namen anotacije in njeno uporabo pri ocenjevanju algoritmov. S tabelami ter grafi predstavimo rezultate testov, izvedenih na podlagi sistema za ocenjevanje, s pomočjo katerih ugotovimo, kateri algoritem je najbolj uspešen.

Poglavje 2

Metoda

V poglavju bomo opredelili delovanje treh že razvitih algoritmov za detekcijo obrazov Viola-Jones [14], Nilsson et al [11] in Kienzle et al [7], ki jih bomo uporabili pri našem delu. Predstavili bomo uporabljene implementacije teh algoritmov in njihove razširitve, kot je komponenta za zapisovanje rezultatov. Rezultate bodo predstavljale koordinate zgornje leve in spodnje desne točke pravokotnikov, ki jih bodo algoritmi izrisali za vsak najden obraz. Razložili bomo tudi pomen parametrov, ki jih bodo sprejele posamezne funkcije za detekcijo ter opisali uporabljena orodja. V nadaljevanju, bomo za izboljšanje delovanja algoritmov in zmanjšanje števila napačnih detekcij, predlagali razdelitev iskalnega okna na dve ločeni liniji, kjer bo prva namenjena obrazom potnikov spredaj, druga pa potnikom zadaj. Iskalno okno bomo razdelili tudi na pet posameznih kvadratov. Vsak od njih bo predstavljal del oziroma področje slike, kjer se bodo lahko nahajali obrazi. Določili jih bomo glede na mesta sedežev v vozilu. Pri razdelitvi slike si bomo pomagali s funkcijo za pravokotno obrezovanje (angl. *crop*) slik.

2.1 Uporabljena orodja

V diplomskem delu smo uporabljali naslednja orodja in programske jezike: odprtokodno knjižnico OpenCV [19], programski jezik Python [21] in prog-

ramsko okolje Matlab [18].

2.1.1 Knjižnica OpenCV

OpenCV (*Open Source Computer Vision*) [19] je odprtokodna knjižnica, namenjena računalniškemu vidu, ki jo je razvilo podjetje Intel. Prvotno je bila napisana v programskem jeziku C, sedaj pa je razvita tudi v C++, Pythonu in Javi. Knjižnica je neodvisna od platforme, saj jo lahko uporabljamo na različnih okoljih, kot so Linux, Windows, Android in Mac. OpenCV vsebuje več kot 2500 optimiziranih algoritmov. Uporabljena je po vsem svetu, znanih je več kot 2,5 milijonov prenosov. Področja uporabe so predvsem splošno procesiranje slik, segmentacija, transformacija, strojno učenje, geometrijski opis, kalibracija kamere itd.

2.1.2 Python

Python [21] je visokonivojski objektno orientiran tolmačeni programski jezik, ki ga je leta 1990 razvil Guido Van Rossum. Odlikuje ga predvsem dinamičnost, enostavna sintaksa in hiter razvoj. Podpira funkcionalno, proceduralno, strukturirano in objektno programiranje. Predvsem zaradi dinamičnih podatkovnih tipov močno spominja na jezike, kot so Perl, Ruby, C#, Visual Basic, Java itd. Na voljo je za vse večje operacijske sisteme, kot so Windows, Linux, Unix in Mac. Razvit je bil kot odprtokodni projekt, kar pomeni, da je njegova uporaba in distribucija brezplačna, tudi za komercialno rabo. Obstaja več implementacij jezika. CPython je prva in najbolj razširjena implementirana v C-jeziku. Jython je implementirana v Javi, IronPython pa v C#. Vedno bolj se uveljavlja tudi PyPy, implementirana v jeziku Python.

2.1.3 Matlab

Matlab [18] je programsko okolje za razvoj algoritmov in numerično računanje, uporablja se tudi pri načrtovanju in analizi podatkov. Razvilo ga je pod-

jetje MathWorks na osnovi knjižnice Lapack. Osnovna enota za delo je polje, osnovna programska struktura okolja pa je matrika. Uporaba Matlaba omogoča hitrejše reševanje različnih tehničnih problemov kot z uporabo programskih jezikov C, C++ in Fortran. Matlab omogoča izvajanje matričnih operacij, reševanje numeričnih enačb ter grafični prikaz rezultatov. Gre za programski jezik, ki je preprost za uporabnika, saj ni potrebna rezervacija spomina in spremenljivk, hkrati pa omogoča hitro in pregledno programiranje.

2.2 Opis uporabljenih algoritmov

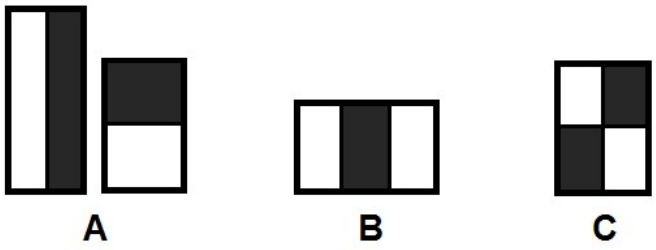
2.2.1 Algoritem Viola-Jones

Algoritem [14] za detekcijo obrazov sta predlagala Paul Viola in Michael Jones. Sposoben je hitrega procesiranja ter doseganja natančne detekcije. Algoritem temelji na naslednjih osnovnih in ključnih konceptih:

- preproste značilnice (angl. *features*) pravokotnih oblik, imenovane značilnice Haar (angl. *Haar features*) [12],
- nov način predstavitve slike, imenovan integralna slika (angl. *integral image*) [3], ki omogoča hitro računanje značilnic,
- strojno učenje AdaBoost [4], namenjeno ustvarjanju učinkovitih klasifikatorjev (angl. *classifier*) in
- združevanje klasifikatorjev v kaskado (angl. *cascade*), ki močno optimizira izvajanje.

Uporaba značilnic Haar [12] je povezana predvsem s hitrostjo. Sistemi, ki temeljijo na značilnicah, so veliko hitrejši kot sistemi, ki temeljijo na slikovnih pikah (angl. *pixel based*). Značilnice Haar so predstavljene kot črno-beli pravokotniki, ki jih algoritem postavlja po integralni sliki in tako ugotavlja njihovo prisotnost na njej. Uporabljajo se tri vrste značilnic, dvopravokotne,

tripravokotne in štiripravokotne, ki so prikazane na sliki 2.1. Prisotnost se izračuna z odštevanjem vrednosti slikovnih pik črnih področij od vrednosti slikovnih pik belih področij. Področja imajo isto velikost in obliko ter med seboj mejijo vodoravno ali navpično (glej sliko 2.1).



Slika 2.1: Dvopravokotne značilnice so prikazane pod A. B prikazuje tripravokotne značilnice, C pa štiripravokotne značilnice.

Torej dvopravokotne značilnice izračunamo tako, da odštejemo vsoto slikovnih pik črnega področja od vsote slikovnih pik belega področja. Tripravokotne značilnice izračunamo tako, da seštejemo vsoto slikovnih pik zunanjih področij in odštejemo vsoto slikovnih pik področja na sredini. Štiripravokotne značilnice pa izračunamo med diagonalnimi pari področij.

Integralna slika [3] je učinkovit in hiter algoritem za izračun vsote vrednosti pravokotne podmnožice mrež. Algoritem je bil predstavljen že leta 1984, vendar na področju računalniškega vida ni bil vidno uveljavljen do njegove uporabe v algoritmu Viola-Jones. Ta način predstavitve slike omogoča hitro računanje pravokotnih značilnic. Integralna slika na lokaciji (x, y) vsebuje vsoto slikovnih pik levo in nad lokacijo (glej sliko 2.2). Definirana je kot:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{2.1}$$

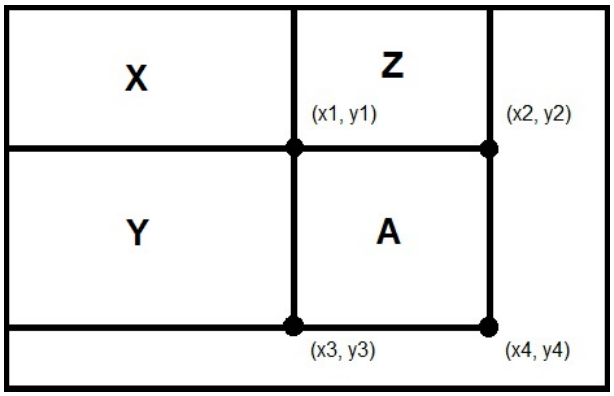
kjer $ii(x, y)$ predstavlja integralno sliko, $i(x', y')$ pa izvirno sliko. Z uporabo naslednjih ponovitev:

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{2.2}$$

in

$$ii(x, y) = ii(x - 1, y) + s(x, y), \tag{2.3}$$

kjer $s(x, y)$ predstavlja kumulativno vsoto vrstice, se integralna slika izračuna že z enim samim sprehodom po izvirni sliki.



Slika 2.2: Vsoto slikovnih točk pravokotnika A izračunamo kot $ii(x4, y4) - ii(x3, y3) - (ii(x2, y2) + ii(x1, y1))$, kjer $ii(x1, y1)$ predstavlja vsoto slikovnih pik pravokotnika X, $ii(x2, y2)$ vsoto slikovnih pik pravokotnika X in Z, $ii(x3, y3)$ vsoto slikovnih pik pravokotnika X in Y ter $ii(x4, y4)$ vsoto slikovnih pik pravokotnikov A, Y, Z in X.

AdaBoost [4] je algoritem za strojno učenje. Uporablja se za učenje klasifikatorjev s pomočjo manjšega števila značilnic. Algoritem iz šibkih (angl. *weak*) klasifikatorjev zgradi močan (angl. *strong*) klasifikator. Njegov cilj je poiskati te značilnice, saj je izgrajen tako, da je sposoben izbrati omejeno množico pravokotnih značilnic iz zelo velike množice potencialno uporabnih značilnic. Po zaključenem učenju je večina značilnic izključena, izbrana pa je zelo omejena množica značilnic, ki najbolje loči pozitivne in negativne primere. Ta množica sestavlja kaskado, ki jo imenujemo tudi močni klasifikator.

Značilnice Haar so potrebne za izgradnjo šibkih klasifikatorjev, ti pa za izgradnjo različnih močnih klasifikatorjev. Algoritem iz množice močnih klasifikatorjev poišče optimalno zaporedje in zgradi kaskado oziroma drevo

odločitev. Ta močno zmanjša čas procesiranja in poveča uspešnost detekcije, saj lahko hitro ugotovi, kateri deli slike so nepomembni.

Uporabili smo implementacijo algoritma Viola-Jones iz OpenCV [19], zasnovano v programskem jeziku Python [21]. Detekciji je namenjena funkcija HaarDetectObject() [17], ki sprejme različne parametre, kot so:

- scaleFactor (minimalna vrednost je 1) določa, za koliko se pomanjša iskalno okno med posameznimi iskanji,
- minNeighbors (minimalna vrednost je 1) določa, koliko sosednjih pravokotnikov je potrebnih, da skupaj tvorijo iskani objekt, in
- minSize, ki določa najmanjšo velikost iskanega objekta in ignorira najdene objekte, manjše od njegove vrednosti.

Ker želimo ugotoviti, pri katerih vrednostih parametrov algoritem najboljše deluje, bomo v nadaljevanju pri testiranju te spreminjali. Implementacijo smo razširili tako, da smo ji dodali komponento za zapisovanje rezultatov. Rezultate predstavljajo koordinate zgornje leve ter spodnje desne točke pravokotnika, ki ga algoritem izriše za vsak najden obraz. Slika 2.3 prikazuje obliko zapisa rezultatov v datoteko.

```
1  /
2  753 126 921 294,
3  86 177 268 360, 750 127 916 294,
4  86 177 268 360, 750 127 916 294,
5  84 176 267 360, 745 130 909 295,
6  84 176 267 360, 745 130 909 295,
7  84 176 267 360, 745 130 909 295,
8  84 176 267 360, 745 130 909 295,
9  84 176 267 360, 745 130 909 295,
10 84 175 261 352, 741 132 906 296,
11 165 91 384 309, 103 176 277 350, 740 130 908 298,
12 165 91 384 309, 103 176 277 350, 740 130 908 298,
```

Slika 2.3: Koordinate ($x1$, $y1$, $x2$, $y2$) so med seboj ločene s presledki, posamezni pravokotniki pa z vejico (,). Desna poševnica (/) predstavlja sliko videa, kjer ni nobenega zaznanega obraza.

2.2.2 Algoritem Nilsson et al

Pristop za detekcijo obrazov [11], ki uporablja kombinacijo:

- lokalnih značilnic SMQT (angl. *The Successive Mean Quantization Transform*) (angl. *local SMQT features*) [10] in
- razdeljenega klasifikatorja SNoW (angl. *Sparse Network of Windows*) (angl. *split up SNoW classifier*) [16].

Transformacija SMQT je postopek za postopno poudarjanje podrobnosti slike. Namenjen je izboljšanju kakovosti slike, tako da so na njej dobro vidne vse podrobnosti. Gre za pristop, ki izvaja samodejno strukturalno razčlenitev podatkov (angl. *automatic structural breakdown of information*). V splošnem je transformacija SMQT stopnje L definirana kot:

$$SMQT_{L} = D(x) \rightarrow M(x), \tag{2.4}$$

kjer x predstavlja podatek poljubne oblike (vektor, matrika, itd.), $D(x)$ množico teh podatkov ter $M(x)$ množico podatkov novih vrednosti (iste velikost in oblike). Pristop za detekcijo obrazov transformacijo izvaja na lokalnih področjih (angl. *local area*) slike, kjer podatek x predstavlja slikovno piko. Pridobljene nove vrednosti so neobčutljive na pridobitve (angl. *gain*) in pristranskost (angl. *bias*), ki predstavljajo predvsem vpliv senzorja oziroma kamere. Celotna intenzivnost slike $I(x)$ je definirana kot:

$$I(X) = g * E(x) * R(x) + b, \tag{2.5}$$

kjer $E(x)$ predstavlja osvetlitev, $R(x)$ odbojnost (angl. *reflectance*), g faktor pridobitve in b izraz pristranskosti. Za izgradnjo klasifikatorja je potrebno, ko ta enkrat že vsebuje strukturo objekta, izločiti odbojnost. Problem ločitve odbojnosti in osvetlitve je rešen s predpostavko, da je osvetlitev konstanta na določenem lokalnem področju. Ob veljavnosti enačbe:

$$E(x) = E, \forall x \in D, \tag{2.6}$$

so rezultat transformacije SMQT na osvetlitve ter senzor (kamera) neobčutljive značilnice.

Arhitektura za učenje SNoW je redka mreža linearnih enot, ki za učenje uporablja posodobitveno pravilo (angl. *Window Update Rule*). Prednost sistema SNoW je možnost ustvarjanja iskalnih (angl. *look-up*) tabel za klasifikacijo. Iskalni tabeli neobrazov in obrazov se lahko združijo v eno samo tabelo:

$$h_x = h_x^{noface} - h_x^{face}, \tag{2.7}$$

zato je klasifikator definiran kot:

$$\sigma = \sum_{x \in W} h_x(M(x)), \tag{2.8}$$

kjer W predstavlja področje značilnic SMQT $M(x)$. Z uporabo posodobitvenega pravila (angl .*Window Update Rule*) se izvede učenje tabel neobrazov in obrazov. Na začetku so vrednosti obeh tabel postavljene na 0, ko pa je posamezen indeks tabele naslovljen prvič, se njegova vrednost postavi na 1. Podatkovna baza za testiranje vsebuje $i = 1, 2, \ldots, N$ ter področja značilnic SMQT M_i(x) in korespondenčne razrede c_i (obraz ali neobraz). Za učenje se uporabljajo trije parametri, in sicer prag γ , parameter napredovanja α (vrednost večja kot 1) in parameter nazadovanja β (vrednost med 0 in 1). Učenje tabel obrazov se izvaja po naslednjem postopku. Če veljata pogoja:

$$\sum_{x \in W} h_x^{face}(M(x_i)) \leq \gamma \tag{2.9}$$

in

$$c_i = face, \tag{2.10}$$

potem se izvede napredovanje, ki je definirano kot:

$$h_x^{face}(M_i(x)) = \alpha * h_x^{face}(M_i(x)), \forall x \in W. \tag{2.11}$$

Ob veljavnosti obratnih pogojev se izvede nazadovanje, definirano kot:

$$h_x^{face}(M_i(x)) = \beta * h_x^{face}(M_i(x)), \forall x \in W. \tag{2.12}$$

Ta postopek se ponavlja, dokler ne pride do spremembe. Učenje tabel neobrazov se izvaja po enakem postopku.

Uporabili smo različico algoritma, implementiranega v okolju Matlab [18]. Tudi tej smo dodali komponento za zapisovanje rezultatov v obliki, kot prikazuje slika 2.3. Za detekcijo obrazov se uporablja funkcija facefind(), ki sprejme naslednje parametre:

- minf, ki določa najmanjšo velikost obraza, in
- sens (vrednost med 1 in 10), ki določa občutljivost detekcije.

Tako kot parametre pri algoritmu Viola-Jones, bomo tudi te pri testiranju spreminjali in skušali ugotoviti, pri katerih vrednostih algoritem najbolje deluje.

2.2.3 Algoritem Kienzle et al (Fdlib)

Algoritem [7] je pristop za detekcijo obrazov, ki za klasifikacijo predlaga uporabo metode podpornih vektorjev (angl. *Support Vector Machine - SVM*)[2]. Gre za metodo razvrščanja oziroma matematični postopek prepoznavanja vzorcev (angl. *pattern recognition*), ki množico iskanih objektov (ti so predstavljeni kot vektorji) razdeli v razrede. Njena naloga je poiskati hiperploskev, ki najbolje loči te razrede med seboj. Znano je, da omenjena metoda zagotavlja natančnost na področju detekcije objektov, vendar je njena uporaba omejena zaradi računsko zahtevne klasifikacije. Ker je ta odvisna od števila podpornih vektorjev, se za rešitev tega problema uporabi metodo, ki jo je predlagal Burges [1]. Metoda za dani podporni vektor ustvari manjšo množico podpornih vektorjev (angl. *Reduce Set Vectors - RSV*). Tako je z uporabo omejene množice vektorjev približno izračunana kompleksna odločitvena funkcija, kar pospeši klasifikacijo, njena natančnost pa ostane ista.

Problem predstavlja tudi število operacij, potrebnih za izračun podobnosti med podpornim vektorjem in vhodom. Predlagana je omejitev (angl.

constraint) manjših množic podpornih vektorjev, tako da so ti lahko ocenjeni preko ločitvenih filtrov (angl. *separable filter*). Filtri se delijo na širšo množico linearnih filtrov in posebno množico nelinearnih filtrov. Uporaba linearnega filtra je definirana kot:

$$J = I * H, \tag{2.13}$$

kjer I predstavlja vhodno sliko, H linearni filter oziroma masko filtra in J izhodno sliko. Tako konvolucija (angl. *convolution*) za H velikosti hxw, potrebuje $h * w$ operacij. V posebnih primerih H lahko razpade na vektorja a in b, tako da velja:

$$H = ab', \tag{2.14}$$

potem je konvolucija definirana kot:

$$J = [I * a] * b'. \tag{2.15}$$

Ker je konvolucija asociativna operacija, se izvirna konvolucija razdeli na dve ločeni operaciji z vektorji velikosti $hx1$ in $wx1$. Če je liearni filter ločljiv (glej enačbo 2.14), potem se računska kompleksnost zmanjša na $h+w$ operacij. Za ločitev linearnega filtra se uporablja singularni razcep (angl. *Singular Value Decomposition - SVD*):

$$H = USV', \tag{2.16}$$

kjer U (velikosti hxh) in V (velikosti wxw) predstavljata pravokotni matriki, S (velikosti hxw) pa diagonalo. Ker je rang S enak rangu H, potem lahko H zapišemo kot:

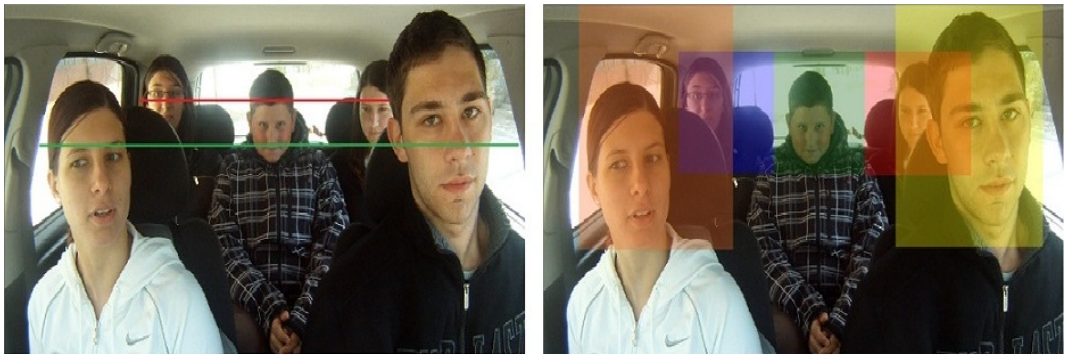
$$H = \sum_{i=1}^{r} s_i u_i v_i', \tag{2.17}$$

kjer r predstavlja rang ene matrike. Tako računska kompleksnost postane $r * (h + w)$ in je odvisna predvsem od r.

Uporabili smo Matlab [18] implementacijo algoritma in ji dodali komponento za zapisovanje rezultatov (glej sliko 2.3). Funkcija fdlibmex(), namenjena detekciji obrazov, sprejme parameter pos (vrednost med -10 in 10), ki določa občutljivost detekcije.

2.3 Razdelitev slike na regije

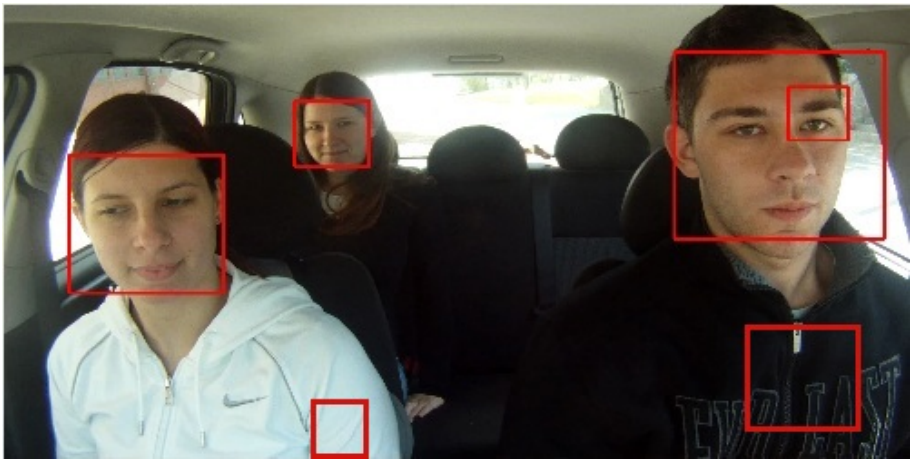
Ob spoznavanju avtomobilskega okolja se srečamo z različnim problemi, ki bi lahko otežili detekcijo. Ena izmed njih je notranjost, ki je močno utesnjena. Tako so obrazi potnikov, ki sedijo zadaj, slabše vidni, saj jih lahko prekrivajo prednji sedeži ali obrazi potnikov, ki sedijo spredaj. Problem predstavljajo tudi različna oddaljenost sedežev od kamere. Tako so obrazi potnikov zadaj slabše detektirani, saj so zaradi večje oddaljenosti manj vidni. Predlagan način za izboljšavo delovanja algoritmov temelji na omejitvi področja iskanja. Potniki v avtomobilu vedno sedijo na sedežih. Voznik in sovoznik imata vsak svoj ločen sedež v prednjem delu, ostalim potnikom pa je namenjen večji sedež v zadnjem delu avtomobila. Pri opazovanju obrazov potnikov smo ugotovili, da ti vedno ležijo na določenem delu slike. Zaradi postavitve in delitve sedežev na prednje in zadnje opazimo, da obrazi potnikov ležijo v dveh linijah, kot prikazuje slika 2.4. V eni liniji ležijo obrazi potnikov v sprednjem delu, v drugi liniji pa obrazi potnikov v zadnjem delu avtomobila. Opazimo tudi, da se položaj potnikovega obraza, glede na sedež, na katerem sedi, ne spreminja (glej sliko 2.4). To nam omogoča omejitev iskanja, saj sliko razdelimo na dve liniji ter pet kvadratov.



Slika 2.4: Levo je prikazana postavitev obrazov v dveh linijah, desno pa tipična področja slike, na kateri se nahajajo obrazi.

Za razdelitev slike smo uporabili funkcije za obrezovanje slike (angl. *crop*). Funkciji izvorno sliko obrežeta v obliki pravokotnika. Potrebno jima je podati koordinati zgornjega levega kota ter dolžino in višino nove slike.

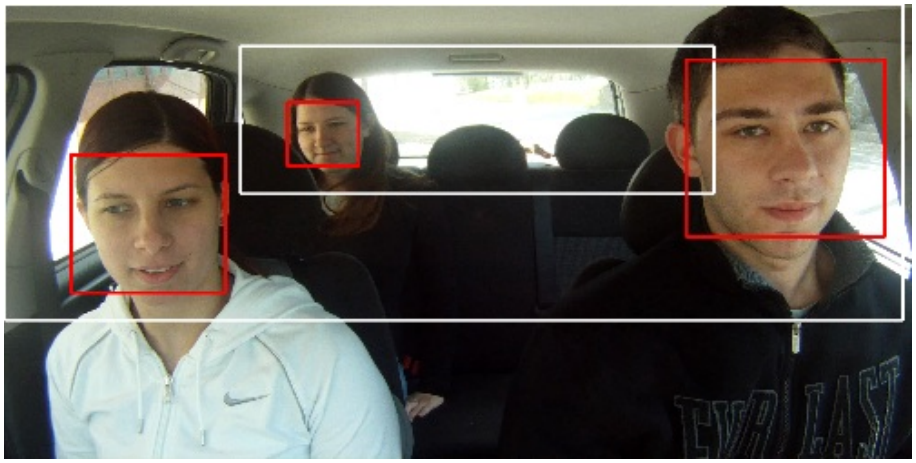
Delitev slike na dve liniji smo izvedli tako, da smo izvorno sliko (glej sliko 2.5), velikosti 1920 x 1080, razdelili na dve ločeni regiji oziroma dva ločena iskalna okna (glej sliko 2.6). Relativne enote parametrov (odstotek), ki smo jih podali funkcijama za obrezovanje slike so navedeni v tabeli 2.1. Posamezen algoritem smo izvedli na vsaki sliki (iskalnem oknu) posebej in rezultate nato združili skupaj v eno datoteko.



Slika 2.5: Primer detekcije na izvirni sliki.

Linija	**% X**	**% Y**	**% Dolžine**	**% Višine**
Spredaj	0 %	0 %	99 %	69 %
Zadaj	26 %	9 %	52 %	32 %

Tabela 2.1: Razdelitev slike na dve liniji. Relativne vrednosti parametrov, ki smo jih podali funkciji za obrezovanje slike. Za posamezeno linijo uporabimo določen odstotek dolžine in višine izvorne slike.

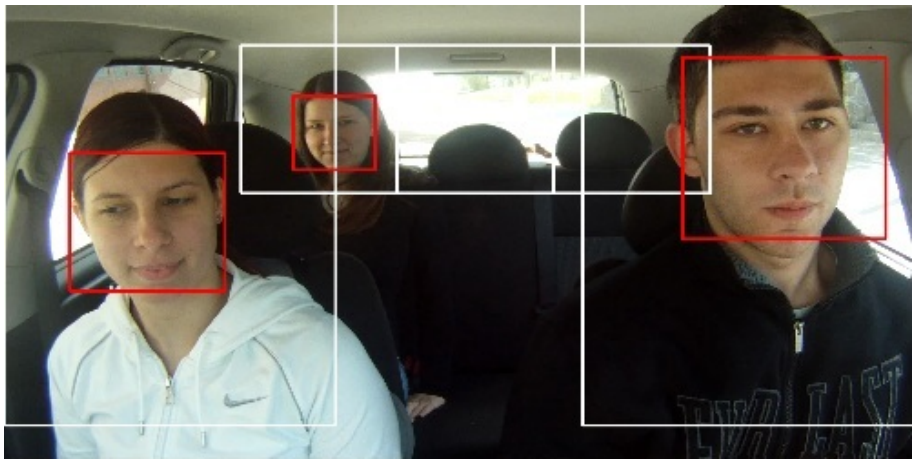


Slika 2.6: Prikaz regij izreza dveh linij.

Postopek smo ponovili pri razdelitvi slike na še manjše regije - okna, kjer se običajno nahajajo obrazi v avtomobilu, z razliko, da smo v tem primeru sliko razdelili na pet manjših kvadratov (glej sliko 2.7). Vsak kvadrat predstavlja področje slike (določimo glede na sedež), kjer v večini primerov ležijo obrazi posameznih potnikov (spredaj levo, spredaj desno, zadaj levo, zadaj na sredini in zadaj desno). Relativne vrednosti parametrov so navedene v tabeli 2.2. Tudi tukaj smo posamezen algoritem izvedli na vsakem kvadratu posebej in rezultate združili.

Kvadrat	**% X**	**% Y**	**% Dolžine**	**% Višine**
Spredaj desno	64 %	0 %	36 %	92 %
Spredaj levo	0 %	0 %	36 %	92 %
Zadaj levo	26 %	9 %	17 %	32 %
Zadaj sredina	43 %	9 %	17 %	32 %
Zadaj desno	60 %	9%	17 %	32 %

Tabela 2.2: Relativne vrednosti parametrov, uporabljenih za razdelitev slike na posamezne kvadrate. Za posamezen kvadrat uporabimo določen odstotek dolžine in višine izvorne slike.



Slika 2.7: Prikaz regij izreza petih kvadratov.

Poglavje 3

Testiranje in rezultati

V tem poglavju bomo predstavili zajem in vsebino podatkov, namenjenih testiranju algoritmov in težave, s katerimi se bomo srečali pri zajemu obrazov potnikov. Videoposnetki, ki jih bomo uporabili pri testiranju, bodo zajeti z uporabo širokokotne digitalne kamere (angl. *wide-angle digital camera*), ob različnih pogojih (dan, noč, premikanje in mirovanje avtomobila) in spreminjanju števila potnikov z njihovim vstopanjem in izstopanjem iz avtomobila. Za potrebe ocenjevanja algoritmov bomo anotirali obraze. Predstavili bomo tudi osnovno delovanje orodja za anotacijo, ki ga bomo implementirali sami. Ocenjevanje bo temeljilo na metodi natančnost-priklic (angl. *precision-recall*) [20] in izračunu ocene, ki nam bo povedala, koliko se dva pravokotnika med seboj prekrivata.

Implementirali bomo sistem za samodejno primerjavo rezultatov anotacije z rezultati algoritmov, kjer bomo za vsako sliko videoposnetka primerjali med seboj množici anotiranih in najdenih pravokotnikov. Na podlagi sistema za ocenjevanje bomo izvedli različne teste, kjer bomo vrednost ocene primerjali z različnimi vrednostmi pragu. Skupaj z njimi bomo izvedli tudi test pravilno in napačno detektiranih slik. S pomočjo tabel in grafov bomo predstavili rezultate izvedenih testov za vsak algoritem posebej in s spreminjanjem vrednosti njihovih parametrov bomo skušali ugotoviti, pri katerih vrednostih bo algoritem najbolj uspešen. Za primerjavo in določitev naj-

boljšega algoritma bomo uporabili najboljše rezultate posameznih algoritmov na vseh videoposnetkih. Na koncu bomo izboljšanje rezultatov skušali doseči s testom pravilno in napačno detektiranih kvadratov in z eksperimentom z latenco.

3.1 Zajemanje podatkov

Delovanje uporabljenih algoritmov smo testirali na videoposnetkih, ki smo jih zajeli s snemalno napravo. Za zajem smo uporabili širokokotno digitalno kamero, ki smo jo namestili v notranjost avtomobila. Usmerili smo jo tako, da je spremljala obraze potnikov ter dogajanje v avtomobilu. Naša želja je bila zajeti obraze potnikov, da bi bili kar najbolj vidni. Ugotovili smo, da so ob nižji postavitvi kamere (npr. nad armaturni plošči avtomobila) dobro vidni obrazi potnikov, ki sedijo spredaj (voznik in sovoznik). Problem predstavljajo obrazi potnikov, ki sedijo zadaj, saj so ti od kamere precej oddaljeni. Večkrat pride tudi do delnega prekrivanja njihovih obrazov z voznikovim in sovoznikovim sedežem ter njihovima obrazoma. Pri višji postavitvi kamere (npr. nad vzvratno ogledalo) so obrazi potnikov, ki sedijo zadaj, bolje vidni. Obrazi potnikov v sprednjem delu avtomobila pa so na sliki le delno vidni oziroma jih kamera ne zajame v celoti.

Izbrali smo digitalno kamero GoPro HD Hero. Gre za majhno, prenosljivo in zmogljivo širokokotno HD (High Definition) kamero. Omogoča snemanje videov v 1080p, 960p in 720p ločljivosti, pri 30 ali 60 slikah na sekundo. Razlog za izbiro kamere je bil predvsem kot snemanja, ki je lahko širok 170° ali 127°. Ugotovili smo, da lahko pri višji namestitvi kamere (v našem primeru nad vzvratno ogledalo (glej sliko 3.1)), zajamemo obraze, da so ti lahko dobro vidni spredaj in zadaj. Višja postavitev kamere omogoča boljšo vidnost obrazov potnikov, ki sedijo zadaj, širok kot kamere pa omogoča, da so obrazi potnikov, ki sedijo spredaj, na sliki v celoti vidni. Primer uporabe kamere je prikazan na sliki 3.2.

Slika 3.1: Namestitev širokokotne kamere nad vzvratno ogledalo.

Slika 3.2: Levo je prikazan zajem slike s kotom velikosti 170°, desno pa s kotom velikosti 127°.

Videoposnetke smo posneli v resoluciji 1920 x 1080 in s kotom velikosti 127°. Za manjši 127°-kot smo se odločili, ker ob uporabi kota velikosti 170° slika postane precej okrogla, kar bi lahko povzročalo težave pri detekciji (glej sliko 3.2). Zajem videoposnetkov smo izvedli tako podnevi, kot ponoči, med dvema različnima stanjema avtomobila, in sicer v mirovanju in pri premikanju. Med snemanjem se število potnikov v avtomobilu spreminja z njihovim izstopanjem in vstopanjem v avtomobil. Potniki se med seboj razlikujejo po videzu, spolu, starosti in višini. Za namen diplomskega dela smo uporabili 9 videoposnetkov (glej tabelo 3.1). Od tega jih je pet posnetih, ko avtomobil miruje, štiri pa med premikanjem avtomobila. En video je bil posnet v temi, ostali pa čez dan (pri dnevni svetlobi).

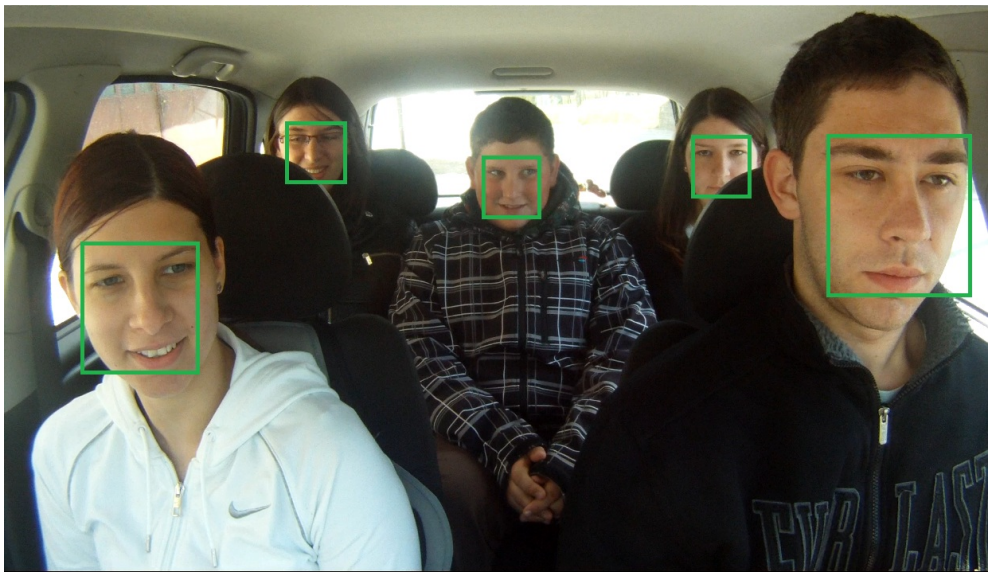
Video	**Mirovanje/Premikanje**	**Dan/Noč**	**Število slik**
1	Mirovanje	Dan	3518
2	Mirovanje	Dan	2282
3	Mirovanje	Dan	1728
4	Mirovanje	Dan	1094
5	Mirovanje	Noč	3984
6	Premikanje	Dan	9973
7	Premikanje	Dan	9972
8	Premikanje	Dan	9996
9	Premikanje	Dan	4224

Tabela 3.1: Videoposnetki, ki smo jih uporabili pri testiranju.

3.2 Anotacija zajetih posnetkov

Namen anotacije je ugotoviti in zbrati veljavne, zanesljive in konsistentne informacije o položajih obrazov na videoposnetkih, kot prvi korak k znanstveni evalvaciji oziroma ocenjevanju. Za anotacijo obrazov smo uporabili pristop, ki temelji na označevanju objektov z geometrijskimi liki, kot so pravokotniki,

kvadrati, krogi, elipse itd [6]. Ker uporabljeni algoritmi za detekcijo obrazov na vsaki sliki videoposnetka najdene obraze označijo s pravokotniki, smo za označevanje lokacije obrazov izbrali pravokotnik. Izbran način označevanja omogoča enostavnejše primerjanje označenih pravokotnikov s pravokotniki, ki jih je našel algoritem. Kot vodilo za določanje robov pravokotnikov smo uporabili značilnosti obraza, kot so obrvi in spodnja ustnica. V primeru, da značilnosti niso bile jasno vidne, smo robove pravokotnika določili približno. Razlog za izbiro teh obraznih značilnosti je predvsem v zagotavljanju skladnosti in konsistentnosti anotacije. Obraz smo označili, če so bile vidne vse njegove glavne značilnosti, oči, večji del ust in nosu. Primer anotacije obrazov je prikazan na sliki 3.3.



Slika 3.3: Obraze označimo s pravokotniki, kot vodilo za določanje robov pravokotnikov uporabimo obrazne značilnosti (obrvi in spodnja ustnica).

Pri označevanju naših videoposnetkov smo si pomagali z orodjem, ki smo ga implementirali v programskem jeziku Python [21], ter s pomočjo knjižnice OpenCV [19]. Orodje zaženemo v ukazni vrstici in mu kot argument podamo ime videoposnetka, ki ga želimo anotirati. V naslednjem koraku ustvarimo datoteko z rezultati. Ti predstavljajo koordinate zgornje leve in spodnje desne točke pravokotnika. Oblika zapisa rezultatov je identična obliki, ki jo uporabljamo pri algoritmih za detekcijo obrazov (glej sliko 2.3). Osnovno delovanje orodja za označevanje obrazov temelji na uporabi miške, s katero označimo obraz. Na naslednjo sliko se premaknemo s tipko za potrditev (angl. *enter*). V primeru neujemanja pravokotnika z obrazom, v ukazni vrstici vnesemo znak "e", potrdimo izbiro in z uporabo funkcije mouseHandler() ponovno izrišemo pravokotnik, kot je prikazano na sliki. Če obraz na sliki ni viden, pravokotnik izpustimo. Orodje omogoča označevanje enega obraza hkrati, zato moramo postopek ponoviti tolikokrat, kot imamo obrazov. Na koncu rezultate anotacije posameznih obrazov združimo v eno samo datoteko, ki predstavlja en videoposnetek. Ker ena vrstica datoteke predstavlja eno sliko videa, združimo iste vrstice posamzenih datotek v eno samo vrstico.

Dodatno smo anotirali premike obrazov, ki bi lahko negativno vplivali na rezultate detekcije. V ta namen smo na enem izmed videoposnetkov označili obraze, medtem ko so potniki premikali glavo v stran. Dodatna anotacija premikov nam je koristila pri eksperimentu z latenco, saj smo lahko izračunali njihov delež.

3.3 Mere za ocenjevanje

Predlagan način evalvacije oziroma ocenjevanja temelji na metodi natančnost-priklic (angl. *precision-recall*) [20], ki se uporablja na področju prepoznavanja vzorcev in pridobivanja informacij (*angl. information retrieval*). Za lažje ocenjevanje smo v okolju Matlab [18] implementirali sistem, ki samodejno primerja datoteki z rezultati posameznega algoritma na videoposnetku

z rezultati anotacije posameznega videoposnetka.

Za primerjavo pravokotnikov uporabljamo oceno:

$$ma(A, B) = \frac{2(P(A \cap B))}{P(A) + P(B)}, \tag{3.1}$$

ki pove, koliko se pravokotnika med seboj prekrivata [9]. $P(A)$ predstavlja ploščino pravokotnika A, $P(B)$ ploščino pravokotnika B in $P(A \cap B)$ presek ploščin pravokotnikov A in B. V implementiranem sistemu je temu namenjena funkcija primerjajPravokotnika(). Predpostavimo, da primerjamo med seboj pravokotnika A in B. Če se pravokotnika ne prekrivata, je ocena enaka 0, kar lahko ugotovimo s preverjanjem naslednjih pogojev (glej sliko 3.4):

$$b_{x_2} < a_{x_1}, \tag{3.2}$$

$$a_{x_2} < b_{x_1}, \tag{3.3}$$

$$b_{y_2} < a_{y_1}, \tag{3.4}$$

$$a_{y_2} < b_{y_1}, \tag{3.5}$$

kjer sta a_{x_1} in a_{y_1} koordinati zgornje leve točke ter a_{x_2} in a_{y_2} koordinati spodnje desne točke pravokotnika A. Koordinati b_{x_1} in b_{y_1} predstavljata zgornjo levo točko, koordinate b_{x_2} in b_{y_2} pa spodnjo desno točko pravokotnika B.

Presek ploščin pravokotnikov A in B izračunamo kot:

$$P(A \cap B) = (\min(a_{x_2}, b_{x_2}) - \max(a_{x_1}, b_{x_1})) * (\min(a_{y_2}, b_{y_2}) - \max(a_{y_1}, b_{y_1})). \tag{3.6}$$

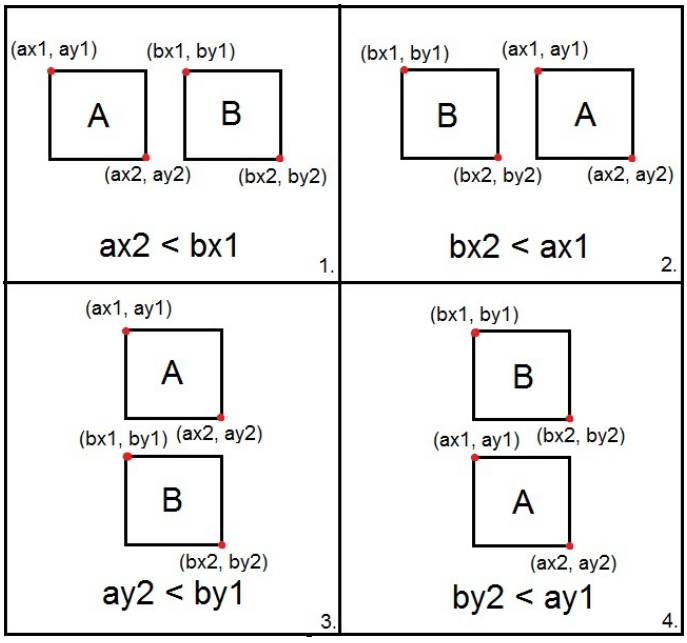
Za dano sliko videa imamo množico anotiranih pravokotnikov A in množico pravokotnikov N, ki jih je našel algoritem med testiranjem. Množici primerjamo tako, da vsak pravokotnik iz N primerjamo z vsemi pravokotniki iz N in izračunamo ocene, koliko se posamezni pravokotniki med seboj prekrivajo. Pri testu (t=0), kjer za prag (angl. *threshold*) določimo vrednost 0, maksimalno oceno posameznega najdenega pravokotnika prištejemo k TP (True Positive) in tako na koncu dobimo vsoto surovih (angl. *raw*) vrednosti

pravilnih ocen. Če je maksimalna ocena enaka 0, potem pravokotnik štejemo kot FP. Pri testih, kjer maksimalno vrednost ocene primerjamo z različnimi vrednostmi pragov (0.1, 0.5 ali 0.7), pa pravokotnik iz N štejemo kot TP, če je njegova maksimalna ocena večja kot izbrani prag. Če je maksimalna ocena manjša od vrednosti pragu, potem pravokotnik štejemo kot FP. Vsak pravokotnik iz A, ki ga algoritem ni našel, štejemo kot FN. FD (False Detection) pa predstavlja vsoto FP in FN. TP in število pravokotnikov iz množice A ter N posamezne slike sproti seštevamo. Tako dobimo število vseh pravilnih ocen, število vseh detekcij in anotacij za posamezen video posntek, ki jih potrebujemo za izračun natančnosti:

$$p = \frac{stevilo\ pravilnih\ ocen}{stevilo\ vseh\ detekcij} \tag{3.7}$$

in priklica:

$$r = \frac{stevilo\ pravilnih\ ocen}{stevilo\ vseh\ anotacij}. \tag{3.8}$$



Slika 3.4: Možni načini neprekrivanja so: A je povsem levo od B (1.), A je povsem desno od B (2.), A je povsem nad B (3.) in A je povsem pod B (4.).

Vrednosti natančnosti in priklica združimo v skupno mero:

$$f = \frac{1}{\frac{\alpha}{p} + \frac{1-\alpha}{r}}. \tag{3.9}$$

Za α določimo vrednost 0,5, kar pomeni, da imata natančnost in priklic enako težo oziroma sta enakovredna.

Na vsaki sliki videoposnetka število pravilnih ocen (TP) primerjamo s številom anotiranih obrazov. Če se vrednosti ujemata, jo štejemo kot pravilno detektirano, sicer kot napačno detektirano. V primeru, da na posamezni sliki videoposnetka (angl. *frame*) ni nobenega anotiranega obraza in ga algoritem tudi ni našel, potem to sliko štejemo kot pravilno detektirano. Na koncu se vsota pravilno in napačno detektiranih slik ujema s številom vseh slik videoposnetka. Delež pravilno detektiranih slik videoposnetka izračunamo kot:

$$\%\ pravilno\ detektiranih\ slik = \frac{stevilo\ pravilno\ detektiranih\ slik}{stevilo\ vseh\ slik}. \tag{3.10}$$

3.4 Testiranje

3.4.1 Test za različne vrednosti praga

Na podlagi sistema za ocenjevanje smo izvedli dva osnovna testa. Razlika med njima je v tem, da pri prvem testu, ki ga imenujemo t=0, med TP prištejemo vrednost ocene (med 0 in 1), ki pove, koliko se dva pravokotnika med seboj prekrivata. TP na koncu predstavlja vsoto surovih vrednosti. Pri drugem testu pa oceno najprej primerjamo z določenim pragom. Če je ta večja kot izbrani prag, jo štejemo kot TP, ki na koncu predstavlja število pravilnih ocen. Slednji test smo izvedli za naslednje vrednosti praga: 0.1, 0.5 in 0.7. Omenjeni testi so namenjeni izračunu natančnosti p, priklica r in ocene f.

3.4.2 Test pravilno in napačno detektiranih slik

Skupaj z zgornjimi testi smo izvedli tudi test pravilno in napačno detektiranih slik (angl. *frame*) videoposnetka, specifično namenjen detekciji potnikov v avtomobilu, ki nam pove, kako natančno algoritem določi število potnikov v avtomobilu.

3.4.3 Test pravilno in napačno detektiranih kvadratov

Specifični test, namenjen detekciji potnikov v avtomobilu, smo izvedli na sliki, razdeljeni na kvadrate. Kvadrat je pravilno detektiran, če algoritem na njem najde vsaj en obraz (pravokotnik), katerega ocena prekrivanja je večja od praga, sicer je kvadrat napačno detektiran. Če v kvadratu ni anotiranega nobenega obraza in ga algoritem tudi ni našel, potem ta kvadrat štejemo kot pravilno detektiran. Test je namenjen predvsem izboljšanju štetja števila ljudi v avtomobilu, saj zanemarimo napačne detekcije, ki za nas niso pomembne.

3.5 Rezultati

Rezultate smo dobili tako, da smo testirali vsak algoritem (Viola-Jones, Nilsson et al in Kienzle at al (Fdlib) algoritem) pod privzetimi parametri na različnih predstavitvah slik. Prvič smo testiranje izvedli na izvirni sliki, drugič na sliki, razdeljeni na dve liniji, in tretjič na sliki, razdeljeni na kvadrate. Izvajali smo teste, kjer za vrednost praga določimo različne vrednosti, kot so 0, 0.1, 0.5 in 0.7. Vzporedno s temi testi smo izvedli tudi test števila pravilno in napačno detektiranih slik videoposnetkov. S pomočjo navedenih testov smo želeli ugotoviti, pri katerem tipu razdelitve slike na regije in katerem pragu posamezen algoritem deluje najbolje. Pri tej določitvi smo se osredotočili na vrednost ocene f. Predstavitvi slike in testu, kjer je algoritem dosegel največjo vrednost ocene f, smo spreminjali vrednosti parametrov funkcije za detekcijo. Tako smo ugotovili, pri katerih vrednostih paramet-

rov algoritem deluje najbolje. Na koncu smo izvedli še povzetek testa na preostalih videoposnetkih s privzetimi in najboljšimi vrednostmi parametrov. Za algoritme, ki najbolje delujejo pri sliki, razdeljeni na kvadrate, smo na koncu dodatno izvedli še test števila pravilno in napačno detektiranih kvadratov, in sicer na vseh videoposnetkih ter vrednostih praga 0.1, 0.5 in 0.7.

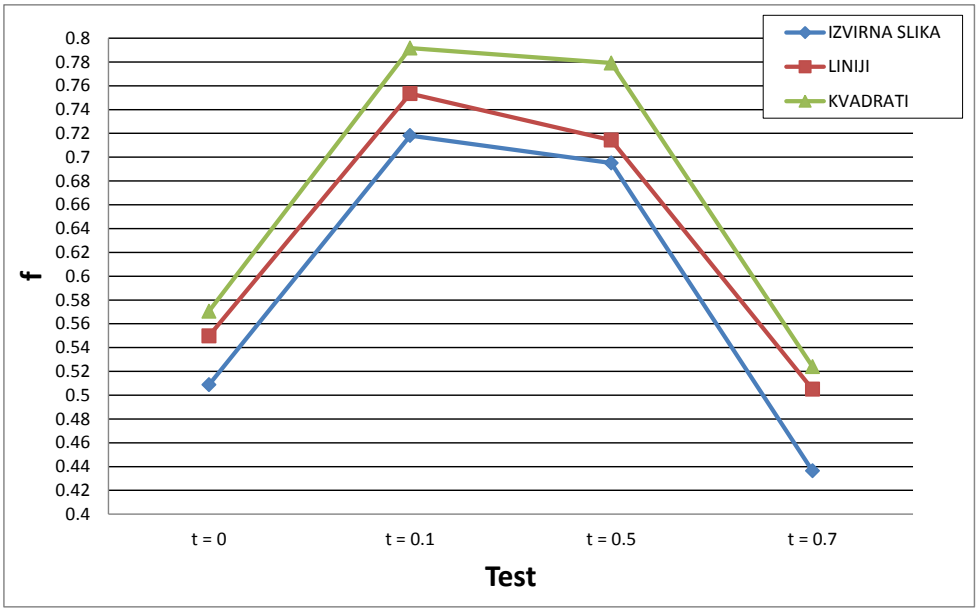
3.5.1 Algortiem Viola-Jones

V tabeli 3.2 so predstavljeni rezultati algoritma Viola-Jones, ki smo jih dosegli pri privzetih vrednostih parametrov in vseh testih, izvedenih na treh razdelitvah slike. Privzete vrednosti parametrov scaleFactor, minNeighbors in minSize so 1.1, 3 in 100. Ugotovili smo, da algoritem doseže največjo vrednost ocene f, natančnost (p) in priklica (r) ter največji odstotek pravilnih detekcij pri diskretnem testu, kjer za prag določimo vrednost 0.1. Primerjali smo vrednost ocene f na treh razdelitvah slike in ugotovili, da je vrednost te največja (0.7916) pri sliki, razdeljeni na pet kvadratov (glej sliko 3.5). Pri tej algoritem doseže tudi najvišjo vrednost natančnosti (0.9465). Na omenjeni razdelitvi slike in testu ($t = 0.1$), smo v nadaljevanju spreminjali vrednosti parametrov. Opazili smo, da se pri omejitvi iskanja, tako na liniji kot na kvadratih, zmanjša število napačnih detekcij (FP in FD). Ugotovili smo tudi, da odstotek pravilnih detekcij sovpada s številom pravilnih detekcij (TP).

	Test	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
IZVIRNA	t=0	9972	20706	19877	10323.1	11317	6073	0.4986	0.5194	0.5087	5258	4714	0.5273
	t=0,1	9972	20706	19877	14571	11441	6135	0.7037	0.7331	0.7181	5206	4766	0.5221
	t=0,5	9972	20706	19877	14104	12375	6602	0.6812	0.7096	0.6951	4925	5047	0.4939
	t=0,7	9972	20706	19877	8856	22871	11850	0.4277	0.4455	0.4364	2475	7497	0.2482
LINIJI	t=0	9972	13387	19877	9143.2	8092	801	0.683	0.46	0.5497	4046	5926	0.4057
	t=0,1	9972	13387	19877	12530	8204	857	0.936	0.6304	0.7534	4004	5968	0.4015
	t=0,5	9972	13387	19877	12330	8604	1057	0.921	0.6203	0.7143	3937	6035	0.3948
	t=0,7	9972	13387	19877	8399	16466	4988	0.6274	0.4225	0.505	2432	7540	0.2439
KVADRATI	t=0	9972	14287	19877	9745	7034	722	0.6821	0.4903	0.5705	4637	5335	0.4650
	t=0,1	9972	14287	19877	13522	7120	765	0.9465	0.6803	0.7916	4594	5378	0.4607
	t=0,5	9972	14287	19877	13308	7548	979	0.9315	0.6695	0.7791	4461	5511	0.4474
	t=0,7	9972	14287	19877	8950	16264	5337	0.6264	0.4503	0.5239	2590	7382	0.2597

Tabela 3.2: Rezultati algoritma Viola-Jones - privzeti parametri.

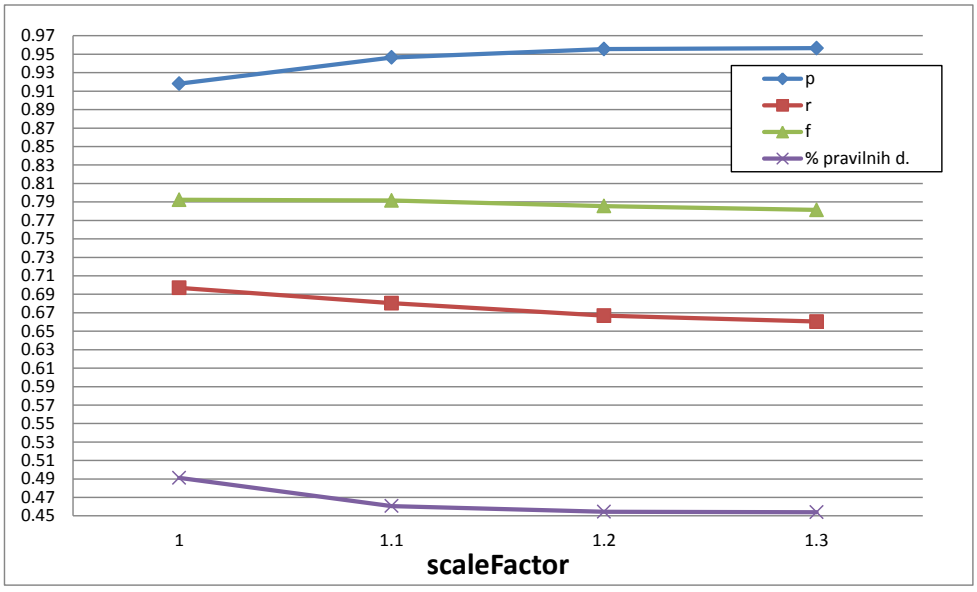
Tabela 3.3 prikazuje rezultate, ki smo jih dosegli pri spreminjanju vrednosti parametrov scaleFactor in minNeighbors. Ugotovili smo, da z večanjem vrednosti, tako parametra scaleFactor kot minNeighbors, narašča vrednost p, obenem pa se zmanjšujejo vrednosti r in ocene f ter odstotek pravilnih detekcij. Iz slik 3.6 in 3.7 je razvidno, da sta parametra dosegla najvišji odstotek pravilnih detekcij pri vrednosti 1. Spreminjanje parametra minSize smo izvedli tako, da smo posebej poiskali najboljšo vrednost za potnike, ki sedijo spredaj, in potnike, ki sedijo zadaj. Ugotovimo, da algoritem doseže največji odstotek pravilnih detekcij pri vrednostih parametra spredaj 63, 64 in 65 (glej sliko 3.8) ter zadaj 36, 37, 38 in 39 (glej sliko 3.9). Opazimo, da z višanjem vrednosti parametra minSize narašča p, z manjšanjem parametra pa r, ki sovpada z odstotkom pravilnih detekcij (glej sliki 3.8 in 3.9).



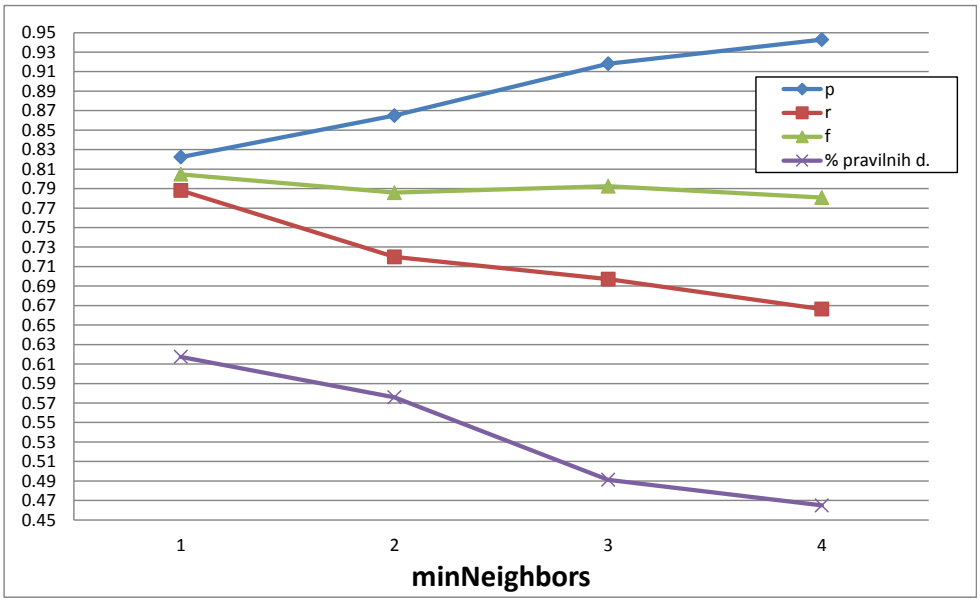
Slika 3.5: Algoritem Viola-Jones: vrednsot ocene f pri vseh izvedenih testih, pri vseh razdelitvah slike.

scaleFactor	minNeighbors	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
1	3	9972	15091	19877	13854	7260	1237	0.918	0.697	0.7924	4898	5074	0.4912
1.1	3	9972	14287	19877	13522	7120	765	0.9465	0.6803	0.7916	4594	5378	0.4607
1.2	3	9972	13869	19877	13253	7240	616	0.9556	0.6668	0.7855	4533	5439	0.4545
1.3	3	9972	13723	19877	13126	7348	597	0.9565	0.6604	0.7813	4526	5446	0.4539
1	1	9972	19051	19877	15663	7602	3388	0.8222	0.788	0.8047	6156	3816	0.6173
1	2	9972	16543	19877	14308	7804	2235	0.8649	0.7199	0.7858	5743	4229	0.5759
1	4	9972	14047	19877	13244	7436	803	0.9428	0.6663	0.7808	4636	5336	0.4649

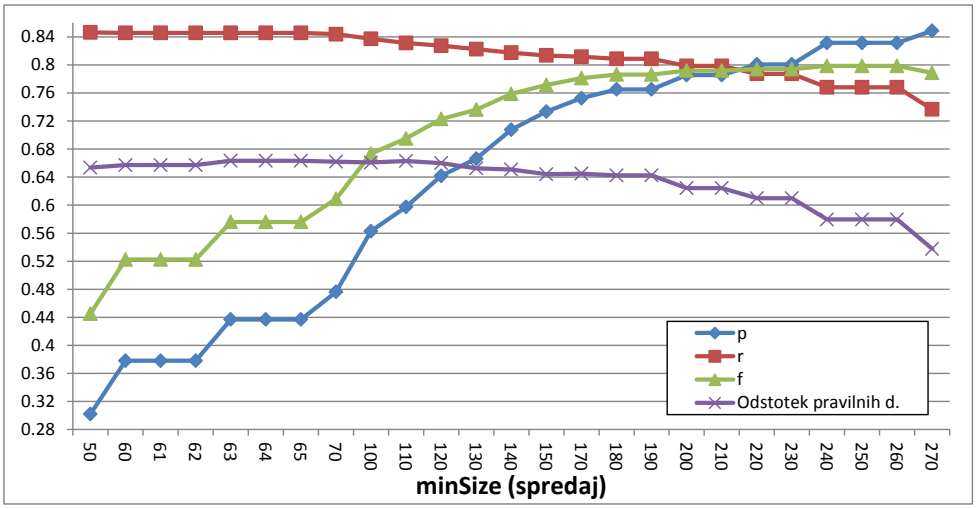
Tabela 3.3: Rezultati, ki smo jih dosegli pri spreminjanju vrednosti parametrov scaleFactor in minNeighbors.



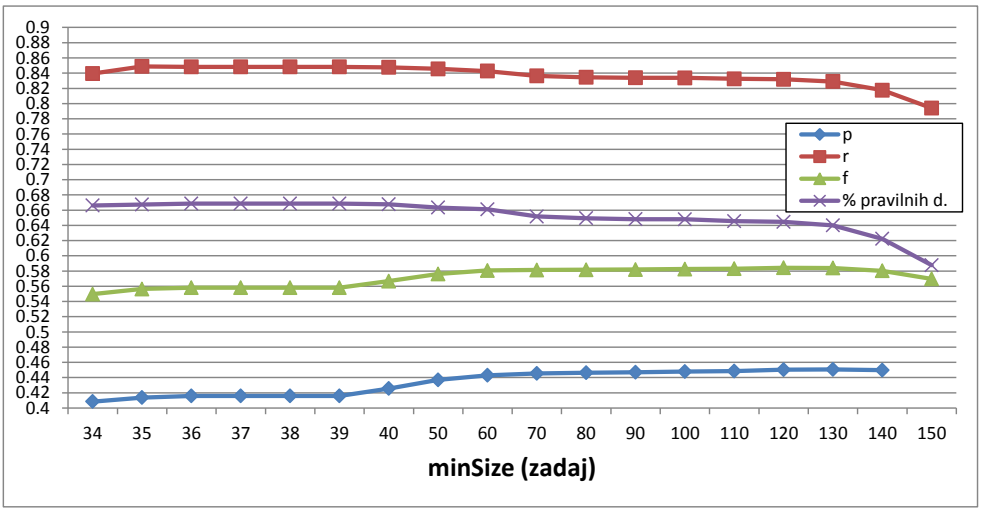
Slika 3.6: Vrednosti p, r, f in odstotek pravilnih detekcij, v odvisnosti od parametra scaleFactor.



Slika 3.7: Vrednosti p, r, f in odstotek pravilnih detekcij, v odvisnosti od parametra minNeighbors.



Slika 3.8: Vrednosti p, r, f in odstotek pravilnih detekcij, v odvisnosti od parametra minSize (spredaj).



Slika 3.9: Vrednosti p, r, f in odstotek pravilnih detekcij, v odvisnosti od parametra minSize (zadaj).

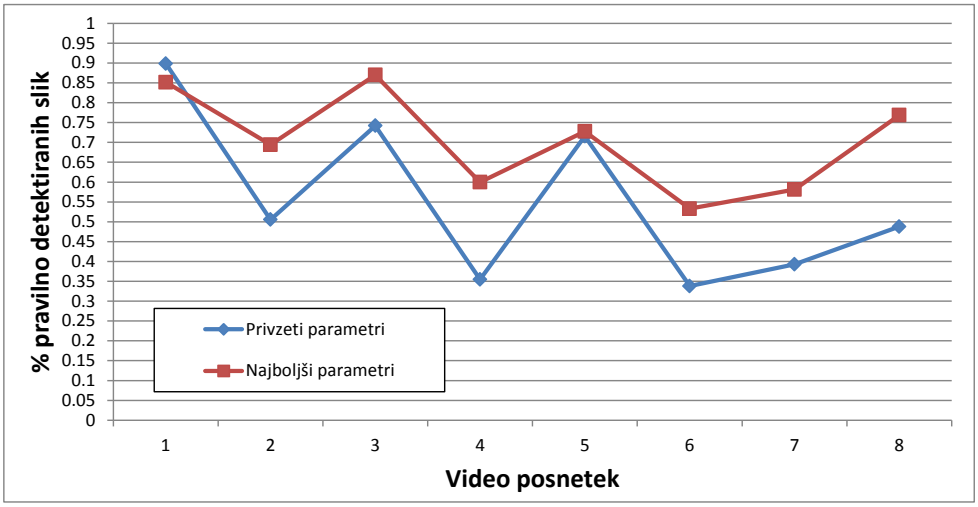
Povzetek rezultatov testiranja algoritma Viola-Jones, pri privzetih parametrih, je predstavljen v tabeli 3.4. Ugotovili smo, da na vseh preostalih videoposnetkih algoritem doseže vrednost p, večjo od 0.9. Vrednost f je sorazmerna z r, ta doseže višje vrednosti pri videoposnetkih z manjšim številom izstopanj potnikov (1, 3, 4 in 8). Opazili smo tudi, da je odstotek pravilnih detekcij najvišji (0.8985) pri videoposnetku 1, kjer se število potnikov ne spreminja. Nižji odstotek pravilnih detekcij algoritem doseže pri videoposnetkih z večjim številom potnikov (4,6,7 in 8). Tabela 3.5 predstavlja povzetek rezultatov testiranja algoritma pri najboljših vrednostih parametrov. Opazili smo, da se je število detekcij močno povečalo, v primerjavi s številom detekcij, pri privzetih parametrih. Posledično se je pri vseh videoposnetkih zmanjšala vrednost p in vrednost f, povečala pa se je vrednost r. Povečanje vrednosti r je povzročilo izboljšanje odstotka pravilno detektiranih slik (glej sliko 3.10), ki pri vseh videoposnetkih preseže vrednost 0.5. Iz tega lahko sklepamo, da je za naš problem vrednost r pomembnejša od vrednosti p. Ugotovili smo, da se števili TP in FP pri najboljših parametrih, v primerjavi s privzetimi, močno povečata.

Video	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
1	3518	2209	2478	2123	441	86	0.961	0.857	0.906	3161	357	0.8985
2	2282	2640	3899	2636	1267	4	0.999	0.676	0.806	1153	1129	0.5053
3	1728	4212	4667	4182	515	30	0.993	0.896	0.942	1282	446	0.7419
4	1094	3707	4494	3640	921	67	0.982	0.81	0.888	388	706	0.3547
5	3984	3660	5239	3580	1739	80	0.978	0.683	0.805	2850	1134	0.7154
6	9973	6299	14104	5753	8897	546	0.913	0.408	0.564	3370	6603	0.3379
7	9996	16760	23490	15675	8900	1085	0.935	0.667	0.779	3925	6071	0.3927
8	4224	15450	16744	14041	4112	1409	0.909	0.839	0.872	2061	2163	0.4879
Skupaj	18609	54937	75115	51630	26792	3307	0.94	0.687	0.794	18190	18609	0.4943

Tabela 3.4: Algoritem Viola-Jones. Povzetek rezultatov na vseh video-posnetkih, pri privzetih vrednostih parametrov.

Video	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
1	3518	5053	2478	2304	2923	2749	0.456	0.9298	0.6119	2996	522	0.8516
2	2282	4758	3899	3161	2335	1597	0.6644	0.8107	0.7303	1584	698	0.6941
3	1728	7167	4667	4441	2952	2726	0.6196	0.9516	0.7505	1503	225	0.8698
4	1094	6864	4494	4032	3294	2832	0.5874	0.8972	0.71	656	438	0.5996
5	3984	9124	5239	4108	6147	5016	0.4502	0.7841	0.572	2899	1085	0.7277
6	9973	31505	14104	9786	26037	21719	0.3106	0.6938	0.4291	5315	4658	0.5329
7	9996	40808	23490	18792	26714	22016	0.4605	0.8	0.5845	5811	4185	0.5813
8	4224	31069	16744	15648	16517	15421	0.5037	0.9345	0.6546	3246	978	0.7685
Skupaj	36799	136348	75115	62272	86919	74076	0.4567	0.829	0.5889	24010	12789	0.6525

Tabela 3.5: Algoritem Viola-Jones. Povzetek rezultatov na vseh video-posnetkih, pri najboljših vrednostih parametrov.



Slika 3.10: Primerjava odstotka pravilno detektiranih slik na vseh video-posnetkih med privzetimi in najboljšimi parametri.

Test pravilno in napačno detektiranih kvadratov

Rezultati testa pravilno in napačno detektiranih kvadratov na vseh videoposnetkih so prikazani v tabeli 3.6 in 3.7. Prva tabela predstavilja rezultate testa pri privzetih parametrih, druga tabela pa pri najboljših parametrih. Pri obeh testih smo ugotovili, da najvišji odstotek pravilno detektiranih kvadratov algoritem doseže, ko pragu določimo vrednost 0.1. Odstotek pravilno detektiranih kvadratov pri najboljših parametrih se, v primerjavi z rezultati pri privzetih parametrih, izboljša pri vseh videoposnetkih, razen pri videoposnetku 1.

Video	Prag (t)	Št. kvadratov	Št. pravilnih k.	Št. napačnih k.	% pravilnih k.
1	0.1	17590	17233	357	0.9797
	0.5	17590	17176	414	0.9765
	0.7	17590	17034	556	0.9684
2	0.1	11410	10147	1263	0.8893
	0.5	11410	10107	1303	0.8858
	0.7	11410	9965	1445	0.8734
3	0.1	8640	8155	485	0.9439
	0.5	8640	8138	502	0.9419
	0.7	8640	8004	636	0.9264
4	0.1	5470	4616	854	0.8439
	0.5	5470	4556	914	0.8329
	0.7	5470	4408	1062	0.8059
5	0.1	19920	18255	1665	0.9164
	0.5	19920	18240	1680	0.9157
	0.7	19920	17083	2837	0.8576
6	0.1	49865	41419	8446	0.8306
	0.5	49865	41252	8613	0.8273
	0.7	49865	40088	9777	0.8039
7	0.1	49980	42057	7923	0.8415
	0.5	49980	41916	8064	0.8387
	0.7	49980	38020	11960	0.7607
8	0.1	21120	18417	2703	0.872
	0.5	21120	18301	2819	0.8665
	0.7	21120	14493	6627	0.6862
Skupaj	0.1	183995	160299	23696	0.8712
	0.5	183995	159686	24309	0.8679
	0.7	183995	149095	34900	0.8103

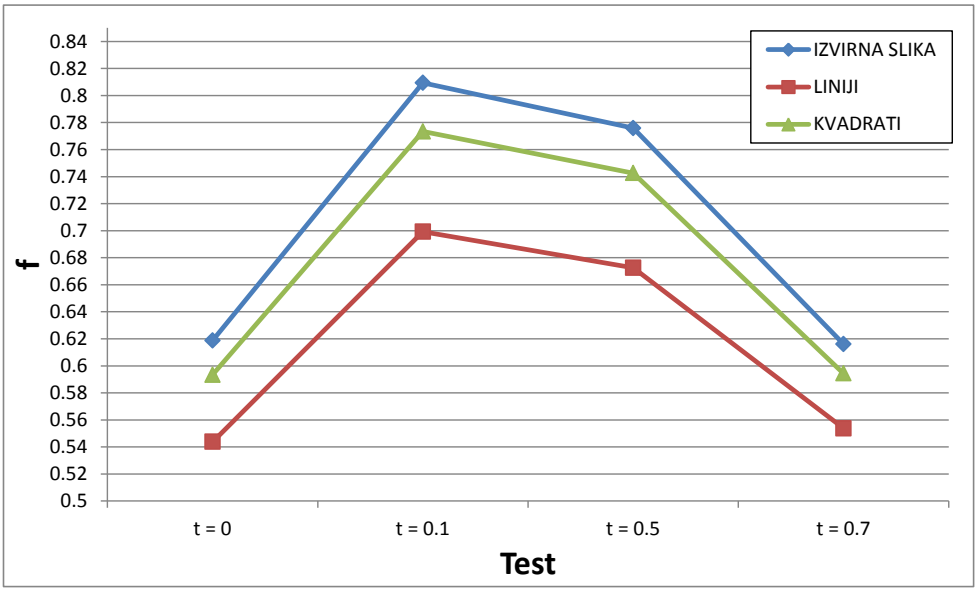
Tabela 3.6: Test pravilno in napačno detektiranih kvadratov (privzeti parametri).

Video	Prag (t)	Št. kvadratov	Št. pravilnih k.	Št. napačnih k.	% pravilnih k.
1	0.1	17590	17068	522	0.9703
	0.5	17590	16942	648	0.9632
	0.7	17590	11697	893	0.9492
2	0.1	11410	10648	762	0.9332
	0.5	11410	10339	1071	0.9061
	0.7	11410	10061	1349	0.8818
3	0.1	8640	8414	226	0.9738
	0.5	8640	8300	340	0.9606
	0.7	8640	8100	540	0.9375
4	0.1	5470	5008	462	0.9155
	0.5	5470	4853	617	0.8872
	0.7	5470	4571	899	0.8356
5	0.1	19920	18594	1326	0.9334
	0.5	19920	18376	1544	0.9225
	0.7	19920	16905	3015	0.8486
6	0.1	49865	44762	5103	0.8977
	0.5	49865	42341	7524	0.8491
	0.7	49865	39777	10088	0.7977
7	0.1	49980	45506	4974	0.9005
	0.5	49980	43321	6659	0.8668
	0.7	49980	38227	11753	0.7648
8	0.1	21120	20024	1096	0.9481
	0.5	21120	19548	1572	0.9256
	0.7	21120	14656	6464	0.939
Skupaj	0.1	183995	170024	14471	0.9241
	0.5	183995	164020	19975	0.8914
	0.7	183995	143994	35001	0.7826

Tabela 3.7: Test pravilno in napačno detektiranih kvadratov (najboljši parametri).

3.5.2 Algoritem Nilsson et al

V tabeli 3.8 so predstavljeni rezultati, ki jih je algoritem Nilsson et al dosegel pri privzetih vrednostih parametrov in vseh testih, izvedenih na vseh treh razdelitvah slike. Privzeta vrednost parametra sens je 5, parametra minf pa 100. Ugotovili smo, da je algoritem največjo vrednost ocene f, p, r in odstotek pravilnih detekcij dosegel pri testu, kjer za prag določimo vrednost 0.1. Pri vseh razdelitvah slike je algoritem dosegel najvišjo vrednost (0.8093) ocene f na izvirni sliki (glej sliko 3.11). Največjo vrednost dosežejo tudi vrednosti p (0.9145) in r (0.7258). Opazili smo tudi, da odstotek pravilnih detekcij sovpada z vrednostjo r.



Slika 3.11: Algoritem Nilsson et al: vrednsot ocene f pri vseh izvedenih testih.

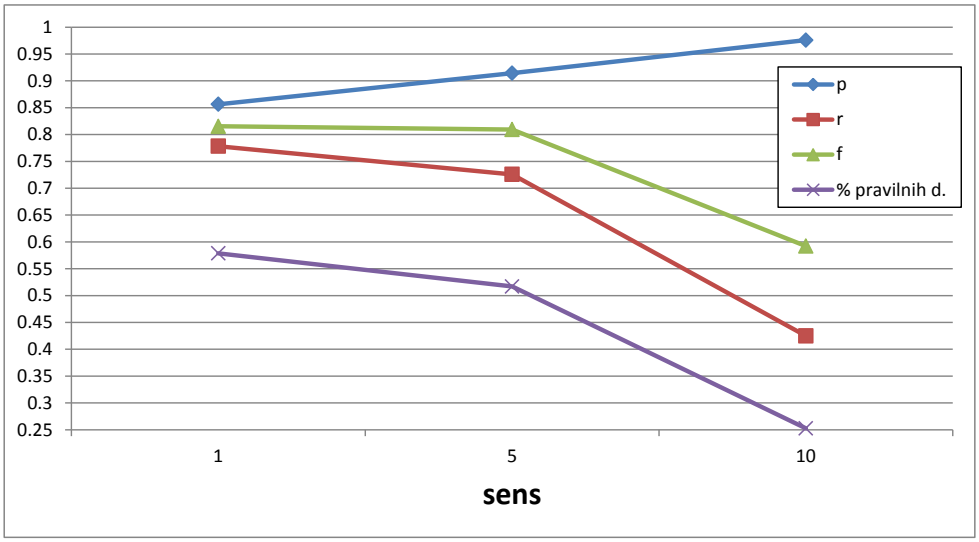
	Test	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
IZVIRNA	t=0	9972	15774	19877	11030	6731	1314	0.699	0.555	0.619	5183	4789	0.5198
	t=0,1	9972	15774	19877	14426	6799	1348	0.915	0.726	0.809	5156	4816	0.5170
	t=0,5	9972	15774	19877	13831	7989	1943	0.877	0.696	0.776	4782	5190	0.4795
	t=0,7	9972	15774	19877	10983	13685	4791	0.696	0.553	0.616	3281	6691	0.3690
LINIJI	t=0	9972	20883	19877	11083	12162	6584	0.531	0.558	0.544	5097	4875	0.5111
	t=0,1	9972	20883	19877	14252	12256	6631	0.683	0.717	0.699	5062	4910	0.5076
	t=0,5	9972	20883	19877	13705	13350	7178	0.656	0.69	0.673	4709	5263	0.4722
	t=0,7	9972	20883	19877	11286	18188	9597	0.54	0.568	0.554	3414	6558	0.3424
KVADRATI	t=0	9972	16061	19877	10663	8058	2121	0.664	0.536	0.593	4901	5071	0.4915
	t=0,1	9972	16061	19877	13898	8142	2163	0.865	0.699	0.773	4869	5103	0.4883
	t=0,5	9972	16061	19877	13344	9250	2717	0.831	0.671	0.743	4510	5462	0.4523
	t=0,7	9972	16061	19877	10680	14578	5381	0.665	0.537	0.594	3159	6813	0.3168

Tabela 3.8: Rezultati algoritma Nilsson et al - privzeti parametri.

V nadaljevanju smo spreminjali vrednosti parametrov sens in minf, na podlagi testa, kjer za prag določimo vrednost 0.1 in izvirne slike. V tabeli 3.9 so predstavljeni rezultati testov, ki smo jih izvedli za vrednosti parametra sens (1, 5 in 10). Ugotovili smo, da se z večanjem vrednosti parametra zmanjšuje odstotek pravilnih detekcij ter vrednosti ocen r in f, hkrati pa se povečuje vrednost p (glej sliko 3.13). Odstotek pravilnih detekcij je najvišji pri vrednosti 1, ki je najmanjša možna vrednost parametra, ki jo algoritem sprejme.

sens	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
1	9972	18064	19877	15468	7005	2596	0.8563	0.7782	0.8154	5770	4202	0.5786
5	9972	15774	19877	14426	6799	1348	0.9145	0.7258	0.8093	5156	4816	0.5170
10	9972	8656	19877	8448	11637	208	0.976	0.425	0.5922	2521	7451	0.2528

Tabela 3.9: Rezultati, ki smo jih dosegli pri spreminjanju vrednosti parametra sens.

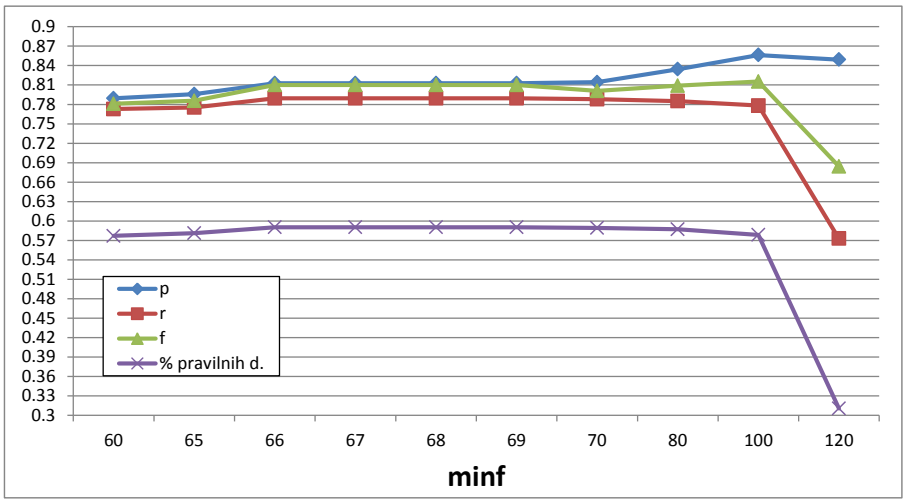


Slika 3.12: Vrednosti p, r, f in odstotek pravilnih detekcij, v odvisnosti od parametra sens.

Tabela 3.10 predstavlja rezultate, ki smo jih dosegli pri spreminjanju vrednosti parametra minf. Ugotovili smo, da algoritem doseže najvišji odstotek pravilnih detekcij pri vrednostih parametra 66, 67, 68 in 69 (glej sliko 3.13).

fsize	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
60	9972	19458	19877	15361	8613	4097	0.7894	0.7728	0.781	5762	4210	0.577
65	9972	19363	19877	15412	8416	3951	0.796	0.7754	0.7856	5795	4177	0.5811
66	9972	19307	19877	15693	7798	3614	0.8128	0.7895	0.81	5887	4085	0.5904
67	9972	19307	19877	15693	7798	3614	0.8128	0.7895	0.81	5887	4085	0.5904
68	9972	19307	19877	15693	7798	3614	0.8128	0.7895	0.81	5887	4085	0.5904
69	9972	19307	19877	15693	7798	3614	0.8128	0.7895	0.81	5887	4085	0.5904
70	9972	19239	19877	15666	7784	3573	0.8143	0.7881	0.801	5877	4095	0.5894
80	9972	18705	19877	15607	7368	3098	0.8344	0.7852	0.809	5857	4115	0.5873
100	9972	18064	19877	15468	7005	2596	0.8563	0.7782	0.8154	5770	4202	0.5786
120	9972	13410	19877	11388	10511	2022	0.8492	0.5729	0.6842	3095	6877	0.3104

Tabela 3.10: Rezultati, ki smo jih dosegli pri spreminjanju vrednosti parametra minf.



Slika 3.13: Vrednosti p, r, f in odstotek pravilnih detekcij, v odvisnosti od parametra minf.

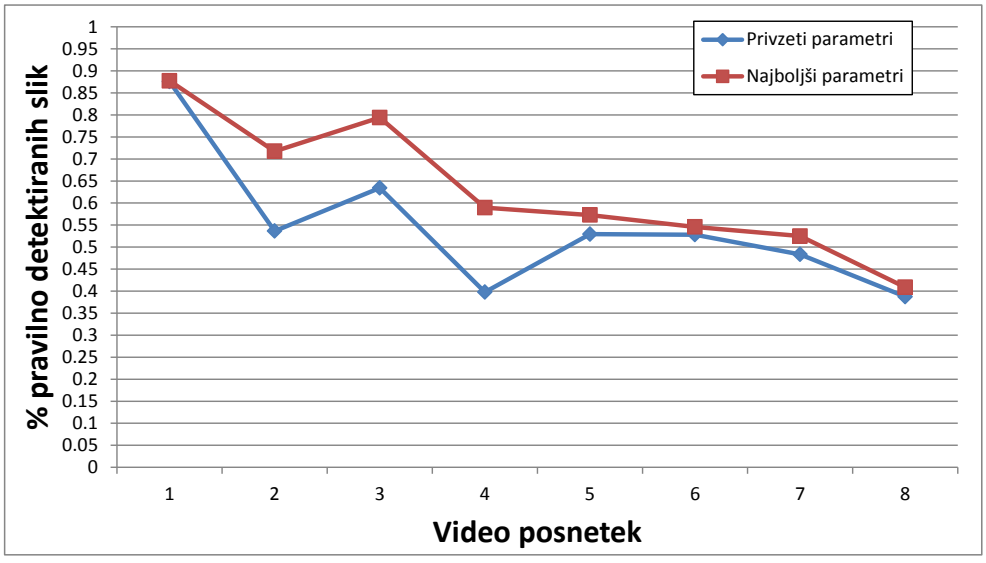
Povzetek rezultatov testiranja algoritma Nilsson et al, pri privzetih parametrih, je predstaviljen v tabeli 3.11, pri najboljših parametrih pa v tabeli 3.12. Ugotovili smo, da pri privzetih vrednostih parametrov vrednost p pri vseh videoposnetkih preseže 0.9. Vrednost r (0.3495), in posledično tudi vrednost ocene f (0.5159), je najmanjša pri petem videoposnetku, zajetem v temi. Odstotek pravilno detektiranih slik (glej sliko 3.14) je najvišji (0.8758) pri prvem videoposnetku, zajetem ob mirovanju avtomobila, kjer se število potnikov ne spreminja. Najnižji odstotek pravilnih detekcij algoritem doseže pri videoposnetkih 4 (0.3976) in 8 (0.3875), kjer je naekrat v avtomobilu veliko število ljudi. Tabela 3.12 predstavlja povzetek rezultatov testiranja algoritma pri najboljših vrednostih parametrov. Povečalo se je tako število detekcij kot število pravilnih detekcij (TP). Pri vseh videoposnetkih se je zvišal priklic in s tem tudi odstotek pravilnih detekcij. V primerjavi s privzetimi vrednostmi parametrov se je število TP in FP, pri najboljših vrednostih parametrov, povečalo.

Video	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pravilnih d.	Št. napačnih d.	% pravilnih d.
1	3518	2196	2478	2091	488	101	0.9539	0.8438	0.8955	3081	437	0.8758
2	2282	2728	3899	2641	1345	87	0.9681	0.6774	0.7970	1224	1058	0.5364
3	1728	3972	4667	3898	844	75	0.9811	0.8352	0.9023	1096	632	0.6343
4	1094	3657	4494	3598	955	59	0.9839	0.8006	0.8828	435	659	0.3976
5	3984	1859	5239	1831	3436	28	0.9849	0.3495	0.5159	2109	1875	0.5294
6	9973	9177	14104	8682	5917	495	0.9461	0.6156	0.7458	5265	4708	0.5279
7	9996	18180	23490	17216	7238	964	0.9470	0.7329	0.8263	4830	5166	0.4832
8	4224	14625	16744	13377	4615	1248	0.9147	0.7989	0.8529	1637	2587	0.3875
Skupaj	36799	56394	75115	53334	24838	3057	0.9457	0.71	0.8111	19677	17122	0.5347

Tabela 3.11: Algoritem Nilsson et al. Povzetek rezultatov na vseh videoposnetkih, pri privzetih vrednostih parametrov.

Video	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pravilnih d.	Št. napačnih d.	% pravilnih d.
1	3518	2824	2478	2205	892	619	0.7808	0.8898	0.8318	3087	431	0.8775
2	2282	3637	3899	3190	1156	447	0.8771	0.8182	0.8466	1637	645	0.7174
3	1728	4672	4667	4264	811	408	0.9127	0.9136	0.9132	1372	356	0.794
4	1094	4176	4494	3995	608	181	0.9567	0.889	0.9216	465	449	0.5896
5	3984	2268	5239	1982	3463	206	0.8739	0.3783	0.528	2282	1702	0.5728
6	9973	11196	14104	8973	5131	2223	0.8014	0.6362	0.7093	5441	4532	0.5456
7	9996	22107	23490	17493	10611	4614	0.6556	0.7447	0.6973	5249	4747	0.5251
8	4224	19305	16744	14117	7815	5188	0.7313	0.8431	0.7832	1727	2497	0.4089
Skupaj	36799	70185	75115	56219	30487	13886	0.801	0.7484	0.7738	21260	15359	0.5777

Tabela 3.12: Algoritem Nilsson et al. Povzetek rezultatov na vseh videoposnetkih, pri najboljših vrednostih parametrov.



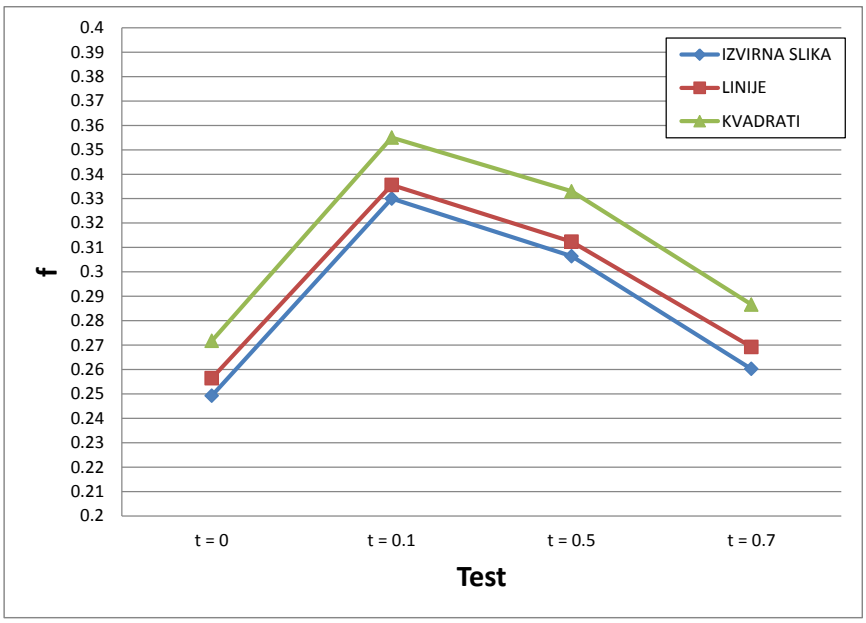
Slika 3.14: Primerjava odstotka pravilno detektiranih slik na vseh videoposnetkih med privzetimi in najboljšimi parametri.

3.5.3 Algoritem Kienzle et al (Fdlib)

V tabeli 3.13 so predstavljeni rezultati algoritma Kienzle et al (Fdlib), doseženi pri privzeti vrednosti parametra in vseh testih, izvedenih na vseh treh predstavitvah slike. Privzeta vrednost parametra pos je 0. Ugotovili smo, da algoritem doseže največjo vrednost ocene f (0.355) ter p (0.3782) pri sliki, razdeljeni na pet kvadratov, in testu, pri pragu 0.1 (glej sliko 3.15). Opazili smo, da algoritem doseže že veliko število napačnih detekcij (FD in FP) pri privzetih parametrih, pri vseh razdelitvah slike. Delovanje algoritma se izboljša pri sliki, razdeljeni na kvadrate, saj se število napačnih detekcij (FD in FP) zmanjša, medtem pa na sliki, razdeljeni na liniji, ostane približno enako v primerjavi z izvirno sliko. Tudi pri tem algoritmu odstotek pravilnih detekcij sovpada s številom pravilnih detekcij (TP). V nadaljevanju smo na podlagi testa, v katerem za prag določimo vrednost 0.1 in slike, razdeljene na kvadrate, spreminjali vrednosti parametra pos.

	Test	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
IZVIRNA	t=0	9972	31052	19877	6347	33281	22228	0.2044	0.3193	0.2492	2241	7731	0.2247
	t=0,1	9972	31052	19877	8403	34123	22649	0.2706	0.4227	0.33	2058	7914	0.2064
	t=0,5	9972	31052	19877	7803	35323	23249	0.2513	0.3926	0.3064	1838	8134	0.1843
	t=0,7	9972	31052	19877	6625	37679	24427	0.2134	0.3333	0.2602	1468	8504	0.1472
LINIJI	t=0	9972	31218	19877	6549.8	33001	22171	0.2098	0.3295	0.2564	2360	7612	0.2367
	t=0,1	9972	31218	19877	8574	33947	22644	0.2746	0.4314	0.3356	2146	7826	0.2152
	t=0,5	9972	31218	19877	7978	35139	23240	0.2556	0.4014	0.3123	1917	8055	0.1922
	t=0,7	9972	31218	19877	6877	37341	24341	0.2203	0.346	0.2692	1560	8412	0.1564
KVADRATI	t=0	9972	17585	19877	5090.1	23272	10409	0.2895	0.2561	0.2717	1541	8431	0.1545
	t=0,1	9972	17585	19877	6650	24162	10935	0.3782	0.3346	0.355	1400	8572	0.1404
	t=0,5	9972	17585	19877	6237	24988	11348	0.3547	0.3138	0.333	1305	8667	0.1309
	t=0,7	9972	17585	19877	5368	26726	12217	0.3053	0.2701	0.2866	1095	8877	0.1098

Tabela 3.13: Rezultati algoritma Kienzle et al - privzeti parametri.

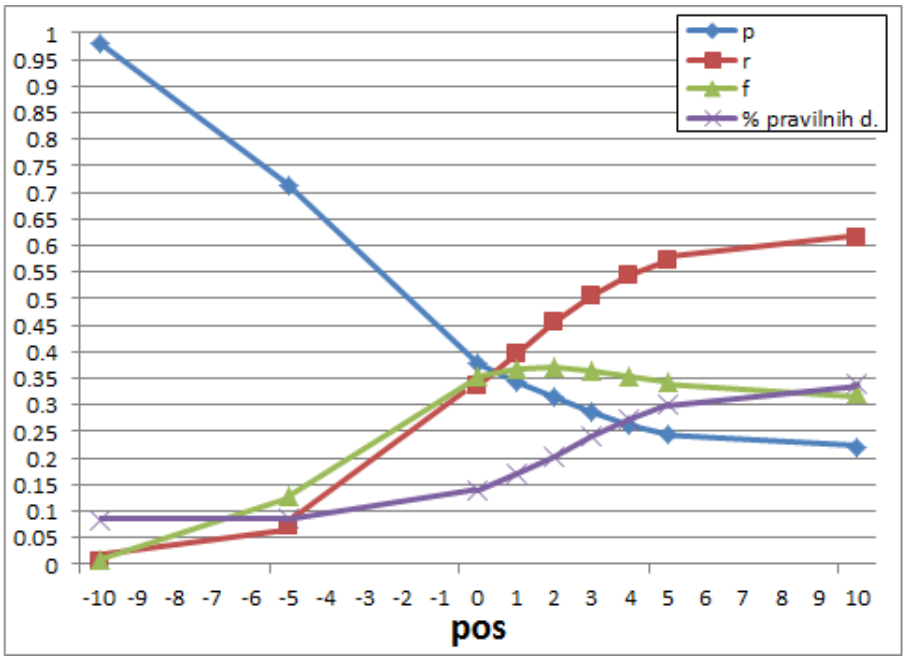


Slika 3.15: Algoritem Kienzle et al. Vrednost ocene f pri vseh izvedenih testih.

V tabeli 3.14 so prikazani rezultati, ki jih je algoritem dosegel pri spreminjanju vrednosti parametra pos. Ugotovimo, da se z večanjem njegove vrednosti zmanjšuje vrednost p, zvišuje pa vrednost r in ocene f (glej sliko 3.16). Odstotek pravilnih detekcij sovpada z r in doseže največjo vrednost pri vrednosti parametra 10, ki je tudi najvišja možna vrednost, ki jo sprejme algoritem.

pos	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
-10	9972	95	19877	93	19786	2	0.9789	0.0047	0.0093	818	9154	0.082
-5	9972	1980	19877	1412	19033	568	0.7131	0.071	0.1292	871	9101	0.0873
0	9972	17585	19877	6650	24162	10935	0.3782	0.3346	0.355	1400	8572	0.1404
1	9972	22901	19877	7852	27074	15049	0.3429	0.395	0.3671	1703	8269	0.1708
2	9972	28884	19877	9043	30675	19841	0.3131	0.4549	0.3709	2031	7941	0.2037
3	9972	34912	19877	10004	34781	24908	0.2865	0.5033	0.3652	2396	7576	0.2403
4	9972	40941	19877	10781	39256	30160	0.2633	0.5424	0.3544	2712	7260	0.2720
5	9972	46344	19877	11346	43529	34998	0.2448	0.5708	0.3427	2986	6986	0.2994
10	9972	55549	19877	12184	51058	43365	0.2193	0.613	0.3231	3369	6603	0.3378

Tabela 3.14: Rezultati, ki jih algoritem doseže pri spremembi parametra pos.



Slika 3.16: Vrednosti p, r, f in odstotek pravilnih detekcij, v odvisnosti od parametra pos.

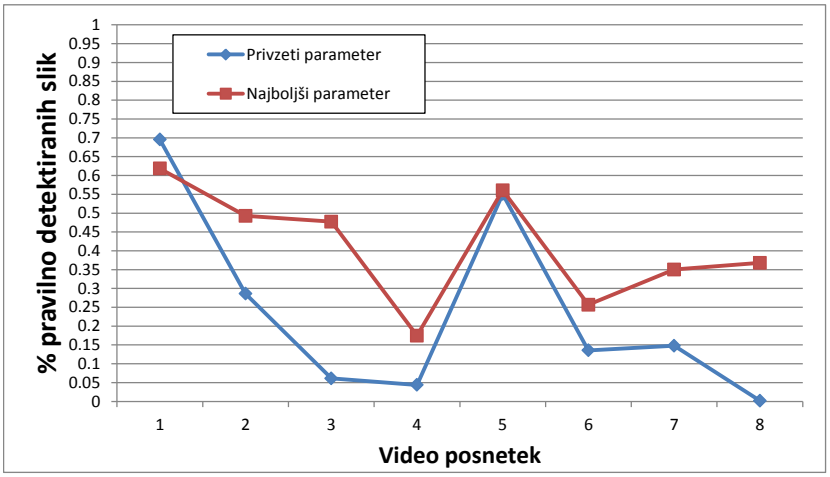
V tabelah 3.15 in 3.16 sta predstavljena povzetka rezultatov testiranja algoritma Kinezle et al (Fdlib). Prva tabela (glej tabelo 3.15) prikazuje rezultate pri privzetem parametru. Vrednost p je višja pri videoposnetkih, posnetih ob mirovanju avtomobila. Najnižjo vrednost p algoritem doseže pri videoposnetku 6, kjer se število potnikov precej spreminja. Odstotek pravilnih detekcij je odvisen od vrednosti r. Najvišji odstotek algoritem doseže pri videoposnetku 1 (0.6953), kjer je tudi vrednost r največja (0.6885), najnižjega pa pri videoposnetku 8, kjer je število potnikov ves čas veliko. V drugi tabeli 3.16 je predstaviljen povzetek rezultatov pri najboljšem parametru pos. Odstotek pravilno detektiranih slik (glej slko 3.17) se je izboljšal pri vseh videoposnetkih, razen pri prvem. Pri vseh videoposnetkih se je zvišala tudi vrednost r, posledično pa zmanjšala vrednost p. Povečalo se je tudi število detekcij, hkrati pa tudi število TP in število napačnih detekcij (FD in FP).

Video	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
1	3518	3336	2478	1706	2402	1630	0.5114	0.6885	0.5869	2446	1072	0.6953
2	2282	2476	3899	1761	2853	715	0.7112	0.4517	0.5525	653	1629	0.2862
3	1728	2029	4667	1552	3592	477	0.7649	0.3325	0.4636	106	1622	0.0613
4	1094	2242	4494	1370	3996	872	0.6111	0.3049	0.4068	48	1046	0.0439
5	3984	2387	5239	2299	3028	88	0.9631	0.4388	0.6029	2194	1790	0.5507
6	9973	15918	14104	3024	23974	12894	0.19	0.2144	0.2015	1353	8620	0.1357
7	9996	16729	23490	9302	21615	7427	0.556	0.396	0.4626	1476	8520	0.1477
8	4224	10928	16744	6017	15638	4911	0.5506	0.3594	0.4349	8	4216	0.0019
Skupaj	36799	56045	75115	27031	77098	29014	0.4823	0.3599	0.4122	8284	28515	0.2251

Tabela 3.15: Algoritem Kienzle et al. Povzetek rezultatov na vseh videoposnetkih, pri privzeti vrednosti parametra.

Video	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
1	3518	13409	2478	2094	11699	11315	0.1562	0.845	0.2636	2176	1342	0.6185
2	2282	7318	3899	2676	5865	4642	0.3657	0.6863	0.4771	1124	1158	0.4926
3	1728	7445	4667	3420	5272	4025	0.4594	0.7328	0.5647	825	903	0.4774
4	1094	6345	4494	2787	5265	3558	0.4392	0.6202	0.5143	191	903	0.1746
5	3984	3270	5239	2356	3797	914	0.7205	0.4497	0.5538	2233	1751	0.5605
6	9973	33351	14104	5704	36047	27647	0.171	0.4044	0.2404	2561	7412	0.2568
7	9996	54446	23490	15223	47490	39223	0.2796	0.6481	0.3907	3500	6496	0.3501
8	4224	41396	16744	12878	32384	28518	0.3111	0.7691	0.443	1554	2670	0.3679
Skupaj	36799	166980	75115	47138	147819	119842	0.2823	0.6275	0.3894	14164	22635	0.3849

Tabela 3.16: Algoritem Kienzle et al. Povzetek rezultatov na vseh videoposnetkih, pri najboljši vrednosti parametra.



Slika 3.17: Algoritem Kienzle et al (Fdlib). Primerjava odstotka pravilno detektiranih slik na vseh videoposnetkih med privzetimi in najboljšimi parametri.

Test pravilno in napačno detektiranih kvadratov

V tabelah 3.17 in 3.18 so predstavljeni rezultati testa pravilno in napačno detektiranih kvadratov na vseh videoposnetkih. Rezultate testa pri privzetih parametrih predstavlja tabela 3.17, rezultate testa pri najboljših parametrih pa tabela 3.18. Ugotovili smo, da je odstotek pravilno detektiranih kvadratov najvišji pri pragu 0.1, pri obeh testiranjih. Pri testiranju z najboljšimi parametri se odstotek pravilno detektiranih kvadratov viša pri vseh videoposnetkih v primerjavi s testom pri privzetih parametrih, razen pri prvem.

Video	Prag (t)	Št. kvadratov	Št. pravilnih k.	Št. napačnih k.	% pravilnih k.
1	0.1	17590	16518	1072	0.9391
	0.5	17590	16463	1127	0.9359
	0.7	17590	16394	1196	0.932
2	0.1	11410	9270	2140	0.8124
	0.5	11410	9249	2161	0.8106
	0.7	11410	9217	2193	0.8078
3	0.1	8640	5525	3115	0.6395
	0.5	8640	5405	3235	0.6256
	0.7	8640	5335	3305	0.6175
4	0.1	5470	2346	3124	0.4289
	0.5	5470	2282	3188	0.4172
	0.7	5470	2242	3228	0.4099
5	0.1	19920	16980	2940	0.8524
	0.5	19920	16960	2960	0.8514
	0.7	19920	16804	3116	0.8436
6	0.1	49865	38207	11658	0.7662
	0.5	49865	37122	12743	0.7445
	0.7	49865	36878	12987	0.7396
7	0.1	49980	35579	14401	0.7119
	0.5	49980	35119	14861	0.7027
	0.7	49980	33878	16102	0.6778
8	0.1	21120	10393	10727	0.4921
	0.5	21120	9978	11142	0.4724
	0.7	21120	9611	11509	0.4551
Skupaj	0.1	183995	134818	49177	0.7327
	0.5	183995	132578	51417	0.7206
	0.7	183995	130359	53636	0.7085

Tabela 3.17: Algoritem Kienzle et al. Test pravilno in napačno detektiranih kvadratov (privzeti parameter).

Video	Prag (t)	Št. kvadratov	Št. pravilnih k.	Št. napačnih k.	% pravilnih k.
1	0.1	17590	16248	1342	0.9237
	0.5	17590	16078	1512	0.914
	0.7	17590	15941	1649	0.9063
2	0.1	11410	10166	1244	0.891
	0.5	11410	9985	1425	0.8751
	0.7	11410	9923	1487	0.8697
3	0.1	8640	7393	1247	0.8557
	0.5	8640	6883	1757	0.7966
	0.7	8640	6750	1890	0.7813
4	0.1	5470	3763	1707	0.6879
	0.5	5470	3374	2096	0.6168
	0.7	5470	3242	2228	0.5927
5	0.1	19920	17036	2884	0.8552
	0.5	19920	16985	2935	0.8527
	0.7	19920	16806	3114	0.8437
6	0.1	49865	40635	9230	0.8149
	0.5	49865	38546	11319	0.773
	0.7	49865	37915	11950	0.7604
7	0.1	49980	41413	8567	0.8286
	0.5	49980	39475	10505	0.7898
	0.7	49980	37406	12574	0.7484
8	0.1	21120	17254	3866	0.817
	0.5	21120	15669	5451	0.7419
	0.7	21120	14735	6385	0.6977
Skupaj	0.1	183995	153908	30087	0.8365
	0.5	183995	146995	37000	0.7989
	0.7	183995	142718	41277	0.7757

Tabela 3.18: Algoritem Kienzle et al. Test pravilno in napačno detektiranih kvadratov (najboljši parameter).

3.5.4 Povzetek rezultatov

Tabela 3.19 prikazuje najboljše rezultate algoritmov za vse videoposnetke skupaj. Razvidno je, da algoritem Viola-Jones, v primerjavi z ostalima dvema, deluje najboljše. To lahko sklepamo iz doseženih vrednosti p (0.4567), r (0.829), f (0.5889) in odstotka pravilno detektiranih slik (0.6525), ki so najvišje (glej tabelo 3.19). Za naš namen je torej najboljši algoritem Viola-Jones, ki je pri vseh videoposnetkih presegel vrednost odstotka pravilno detektiranih slik 0.5.

Algoritem	Št. slik	Št. detekcij	Št. anotacij	TP	FD	FP	p	r	f	Št. pra. d. s.	Št. nap. d. s.	% pra. d. s.
Viola-Jones	36799	136348	75115	62272	86919	74076	0.4567	0.829	0.5889	24010	12789	0.6525
Nilsson et al	36799	70185	75115	56219	30487	13886	0.4344	0.4059	0.4195	21260	15359	0.5777
Kienzle et al	36799	166980	75115	47138	14789	119842	0.2823	0.6275	0.3894	14164	22635	0.3849

Tabela 3.19: Najboljši rezultati vsakega izmed algoritmov za vse videoposnetke skupaj.

Za algoritma Viola-Jones in Kienzle et al smo izvedli tudi test števila pravilno in napačno detektiranih kvadratov. Ugotovili smo, da je tudi pri tem testu algoritem Viola-Jones dosegel, tako pri privzetih vrednostih kot pri najboljših vrednostih parametrov, boljše rezultate. Pri privzetih vrednostih parametrov je algoritem dosegel odstotek pravilnih kvadratov v povprečju 0.89, pri najboljših pa v povprečju 0.93.

3.5.5 Eksperiment z latenco

Eksperiment je namenjen izboljšanju rezultatov testa pravilno in napačno detektiranih kvadratov. Detekcijo lahko otežujejo tudi nenadni premiki potnikove glave v stran, česar vpliv smo želeli zmanjšati z uporabo latence. Eksperiment so izvedli na videoposnetku z 9976 slikami, na katerem smo prej testirali delovanje algoritmov in spreminjali vrednosti parametrov. Za eksperiment smo izbrali algoritem Viola-Jones pri najboljših vrednostih parametrov, kjer je ta dosegel najbolše rezultate. Za posamezen kvadrat smo spremljali detekcijo obrazov. Če je algoritem detektiral obraz na določenem

zaporednem številu slik, nato pa na nekaj zaporednih slikah obraza ni detektiral, te kvadrate označimo, kot v primeru, ko algoritem dejansko detektira obraz. To izvedemo tudi v nasprotnem primeru. Zaporedje detektiranih ali nedetektiranih vmesnih kvadratov določimo s parametrom latence.

Tabela 3.20 prikazuje rezultate, ki mso jih dosegli pri eksperimentu z latenco. Za parameter latence smo določili vrednosti med 0 in 50, s korakom 5. Ugotovili smo, da je algoritem dosegel navečji odstotek pravilno detektiranih kvadratov pri vrednosti parametra latence 30, najmanjši pa pri vrednosti 10. Opazimo tudi, da se je odstotek pravilno detektiranih kvadratov povečal pri vseh podanih vrednostih parametra latence, razen pri vrednosti 10.

Latenca	Št. kvadratov	Št. pravilno d. k.	Št. napačno d. k.	% pravilno d. k.
0	49860	19917	29943	0.3995
5	49860	20001	29859	0.4011
10	49860	19817	30043	0.3975
15	49860	20226	29634	0.4057
20	49860	20211	29649	0.4054
25	49860	20816	29044	0.4175
30	49860	20965	28895	0.4205
35	49860	20709	29151	0.4153
40	49860	20829	29031	0.4177
45	49860	20862	28998	0.4184
50	49860	20346	29514	0.4081

Tabela 3.20: Rezultati eksperimenta z latenco.

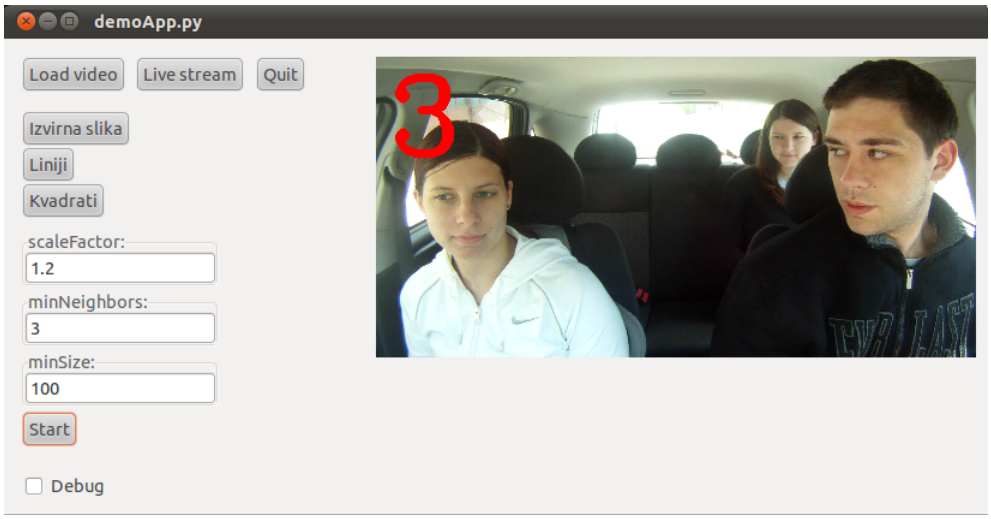
Poglavje 4

Zaključek

Napredek računalniškega vida je pripomogel k hitremu razvoju aplikacij na tem področju. Med njimi so tudi aplikacije, ki so namenjene sistemom za nadzor in varnost in temeljijo na detekciji različnih objektov. Tudi v avtomobilski industriji se uveljavljajo različni sistemi, ki so osnovani na detekciji obrazov. Omogočili naj bi nadzor dogajanja v avtomobilu in hkrati povečali varnost potnikov.

Pri našem delu smo se seznanili z že obstoječimi algoritmi (Viola-Jones, Nilsson et al in Kienzle et al) za detekcijo obrazov, ki smo jih uporabili za namen sistema za detekcijo potnikov v vozilu. Predlagali smo razdelitev na dve liniji in pet kvadratov, kar je izboljšalo delovanje algoritmov. Testiranje algoritmov smo izvedli na podlagi implementiranega sistema za ocenjevanje in različnih testov. V praktičnem delu smo z uporabo digitalne kamere zajeli videoposnetke, namenjene testiranju algoritmov. Implementirali smo orodje za anotacijo videoposnetkov. S spreminjanjem vrednosti parametrov posameznih algoritmov smo poiskali najboljše parametre delovanja za detekcijo v vozilu.

Na koncu smo razvili demo aplikacijo (glej sliko 4.1), ki omogoča nalaganje videoposnetka, izbiro razdelitve slike, spreminjanje parametrov algoritma in izpisovanje števila potnikov.



Slika 4.1: Primer delovanja demo aplikacije.

Pri testiranju smo ugotovili, da se algoritmi med seboj razlikujejo v delovanju, saj sta Viola-Jones in Kienzle et al bila najbolj uspešna pri sliki, razdeljeni na kvadrate, medtem ko je algoritem Nilsson et al najbolje deloval pri izvirni sliki. Pri primerjanju uspešnosti delovanja algoritmov smo se osredotočili na odstotek pravilno detektiranih slik. Opazili smo, da je algoritem Viola-Jones dosegel najboljše rezultate pri šestih videoposnetkih, vključno z videoposnetkom, na katerem smo spreminjali vrednosti parametrov. Na preostalih treh videoposnetkih je bil uspešnejši algoritem Nilsson et al, medtem ko se je algoritem Kienzle et al odrezal najslabše, saj je na vseh videoposnetkih dosegel najnižji odstotek pravilnih detekcij ter občutno višje število napak. Razlog za slabše rezultate algoritma Kienzle et al vidimo v tem, da ta ne sprejme parametra, s katerim spreminjamo minimalno velikost iskanih obrazov. Pri ostalih dveh smo s pomočjo parametrov bolj učinkovito povečali število TP, kar je za nas tudi najpomembneje.

Problema, ki sta se izpostavila pri praktičnem delu, sta: utesnjenost prostora, ki povzroča prekrivanje obrazov, in kompleksno okolje, ki povečuje število napačnih detekcij. Dodaten problem je povzročala osvetlitev, ki ponoči otežuje detekcijo. Algoritmi so obraze detektirali le ob dodatni svet-

lobi v avtomobilu, kot je notranja osvetlitev vozila, ki se vključi ob vstopu potnika v avtomobil. Ker je ta nameščena v sprednjem delu, so bili obrazi potnikov v zadnjem delu slabše vidni in posledično slabše detektirani. Ta problem lahko rešimo z vgradnjo dodatne osvetlitve avtomobila, predvsem v zadnjem delu, tako da ta ne bi motila voznika.

Metode, ki smo jih uporabili v diplomskem delu, so primerne za naš namen, saj smo s pomočjo teh ugotovili primernost uporabe detekcije in prepoznali morebitne probleme, ki se pri detekciji obrazov v avtomobilu lahko pojavijo. Prav tako smo s pomočjo navedenih metod odkrili, kateri algoritem je najboljši in najbolj primeren za uporabo. To smo dosegli z izvajanjem testov za vse tri algoritme in s spreminjanjem parametrov. Naš cilj smo dosegli, saj smo ugotovili, da je algoritem Viola-Jones najbolj uspešen. Na koncu smo boljše delovanje algoritmov dosegli z ustrezno razdelitvijo slike oziroma iskalnega okna.

Slike

Tabele

Literatura

[1] C. C. J. Burges, "Simplified Support Vector Decision Rules", International Conference on Machine Learning, str. 71–77, 1996.

[2] C. Cortes, V. Vapnik, "Support-vector networks", Machine Learning, letnik 20, št. 3, str. 273–297, 1995.

[3] F. Crow, "Summed-area tables for texture mappings", Proceedings of SIGGRAPH, zbornik 18, Str. 207–212, 1984.

[4] Y. Freund, R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Computational Learning Theory: Eurocold '95, str. 23–37, Springer-Verlag, 1995.

[5] M. Hoon Yap, H. Ugail, R. Zwiggelaar, B. Rajoub, V. Doherty, S. Appleyard, G. Hurdy, "A Short Review of Methods for Face Detection and Multifractal Analysis", 2009 International Conference on CyberWorlds, str. 231–236, UK, 2009.

[6] R. Kasturi, D. Goldgof, P. Soundararjan, V. Manohar, J. Garofolo, R. Bowers, M. Boomstra, V. Korzhova, J. Zhang, "Framework of Performance Evaluation of Face, Text and Vehicle Detection and Tracking in Video: Data, Metric and Protocol", IEEE Transactions on Pattern Analysis and Machine Intelligence, zbornik 31, str. 319–336, 2009.

[7] W. Kienzle, G. Bakir, M. Franz, B. Scholkopf, "Face Detection - Efficient and Rank Deficient", Advances in Neural Information Processing Systems, zbornik 17, str. 673–680, 2005.

[8] V. Kulkarni, V. Babu, "Embedded Smart Car System on Face Detection", International Journal of Computer & Communication Technology IJCCT, letnik 3, št. 1, str. 112–116, 2012.

[9] S. M. Lucas et al, "ICDAR 2003 robust reading competitions: entries, results and future directions", Springer-Verlag, str. 105–122, 2005.

[10] M. Nilsson, M. Dahl, I. Claesson, "The Successive Mean Quantization Transform", Acoustic Speech and Signal Processing (ICASSP '05), IEEE International Conference on Image Processing (ICIP), zbornik 2, str. 429–432, 2005.

[11] M. Nilsson, J. Nordberg, I. Cleasson, "Face Detection Using Local SMQT Features and Split Up SNoW Classifier", Acoustics, Speech and Signal Processing (ICASSP), zbornik 2, str. 589–592, 2007.

[12] C. Papageorgiou, M. Oren, T. Poggio, "A general framework for object detection", Sixth International Conference On Computer Vision, str. 555–562, 1998.

[13] F. Solina, "Računalniški vid nekdaj in danes", Računalniška obdelava slik in njena uporaba v Sloveniji ROSUS, Maribor, 2006.

[14] P. Viola, M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE Computer Vision and Pattern Recognition, zbornik 1, str. 511–518, Kauai, 2001.

[15] S. Wender, O. Loehleim, H. M. Gross, "Multiple Classifier Cascade for Vehicle Occupant Monitoring Using an Omnidirectional Camera", IEEE Intelligent Vehicles Symposium, str. 345–350, 2004.

[16] M. Yang, D. Roth, N. Ahuja, "A SNoW-Based Face Detector", Advances In Neural Informations Processing Systems 12 (NIPS 12), str. 855–861, MIT Press, 2000.

[17] (2012) Cascade Classification. Dostopno na:
http://opencv.itseez.com/modules/objdetect/doc/cascade_classification.html

[18] (2012) MATLAB – The Language of Technical Computing. Dostopno na:
http://www.mathworks.com/products/matlab/

[19] (2012) OpenCV. Dostopno na:
http://opencv.willowgarage.com/wiki/

[20] (2012) Precision and recall. Dostopno na:
http://en.wikipedia.org/wiki/Precision_and_recall

[21] (2011) Python. Dostopno na:
http://python.org/