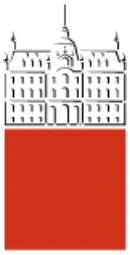
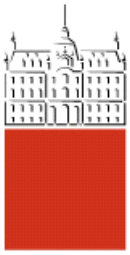
Univerza
v Ljubljani
Fakulteta
za gradbeništvo
in geodezijo



Jamova cesta 2
1000 Ljubljana, Slovenija
http://www3.fgg.uni-lj.si/

University
of Ljubljana
Faculty of
Civil and Geodetic
Engineering



Jamova cesta 2
SI – 1000 Ljubljana, Slovenia
http://www3.fgg.uni-lj.si/en/

Univerza
v Ljubljani

Fakulteta za
gradbeništvo in
geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si

UNIVERZITETNI ŠTUDIJ
PRVE STOPNJE GEODEZIJE
IN GEOINFORMATIKE

Kandidat:

JERNEJ NEJC DOUGAN

SAMODEJNO PREPOZNAVANJE PREHODOV ZA PEŠCE NA DIGITALNI SLIKI

Diplomska naloga št.: 13/GIG

AUTOMATIC PEDESTRIAN CROSSING DETECTION ON DIGITAL IMAGE

Graduation thesis No.: 13/GIG

Mentorica:
doc. dr. Mojca Kosmatin Fras

Somentor:
asist. dr. Dejan Grigillo

Član komisije:
izr. prof. dr. Krištof Oštir
doc. dr. Anka Lisec

Predsednik komisije:
izr. prof. dr. Dušan Kogoj

Ljubljana, 24. 09. 2012

STRAN ZA POPRAVKE, ERRATA

Stran z napako	Vrstica z napako	Namesto	Naj bo

IZJAVE

Podpisani **JERNEJ NEJC DOUGAN** izjavljam, da sem avtor diplomske naloge z naslovom: »**SAMODEJNO PREPOZNAVANJE PREHODOV ZA PEŠCE NA DIGITALNI SLIKI**«.

Izjavljam, da je elektronska različica v vsem enaka tiskani različici.

Izjavljam, da dovoljujem objavo elektronske različice v repozitoriju UL FGG.

Ljubljana, 18. 7. 2012 Jernej Nejc Dougan

BIBLIOGRAFSKO – DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK: **528.7:625.7(043.2)**
Avtor: **Jernej Nejc Dougan**
Mentor: **Doc. dr. Mojca Kosmatin Fras, univ. dipl. inž. geod.**
Somentor: **Asist. dr. Dejan Grigillo, univ. dipl. inž. geod.**
Naslov: **Samodejno prepoznavanje prehodov za pešce na digitalni sliki**
Tip dokumenta: **Diplomska naloga – univerzitetni študij**
Obseg in oprema: **20 str., 17 sl., 1 pregl., 11 en., 6 pril.**
Ključne besede: **prehod za pešce, prepoznavanje vzorcev, korelacija, Houghova transformacija, projektivna transformacija**

Izvleček

Količine podatkov, ki jih je potrebno obdelati, so vedno večje, zato je smiselno procese avtomatizirati. Digitalne fotografije oziroma slike so vir informacij, ki jih lahko z računalniškimi algoritmi izluščimo in interpretiramo. V diplomski nalogi smo na kratko opisali, kaj je digitalna slika in kako jo lahko obdelujemo. Poudarek naloge je bil na praktičnem delu, kjer smo razvili algoritem za samodejno prepoznavanje prehodov za pešce na digitalni sliki. Program je napisan v programskem okolju Matlab.

BIBLIOGRAPHIC – DOCUMENTATALISTIC INFORMATION AND ABSTRACT

UDC: **528.7:625.7(043.2)**
Author: **Jernej Nejc Dougan**
Supervisor: **Assist. Prof. Mojca Kosmatin Fras, Ph.D.**
Cosupervisor: **Dejan Grigillo, Ph.D.**
Title: **Automatic pedestrian crossing detection on digital image**
Document type: **Graduation Thesis – University studies**
Scope and tools: **20 p., 17 fig., 1 tab., 11 eq., 6 ann.**
Keywords: **pedestrian crossing, pattern recognition, correlation, Hough transformation, projective transformation**

Abstract

We have to process more and more data every day, so it makes sense to automate as many processes as possible. Digital images are source of information, which can be extracted and interpreted with computer algorithms. In bachelor thesis we are dealing with digital image and image processing. The emphasis of the thesis is the practical case study. We had developed algorithm for automatic pedestrian crossing detection in digital image. Program is written in Matlab.

ZAHVALA

Za pomoč in podporo pri nastajanju diplomske naloge se iskreno zahvaljujem mentorici doc. dr. Mojci Kosmatin Fras in somentorju asist. dr. Dejanu Grigillu.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK

OKRAJŠAVE

MMS	Mobilni merski sistem (ang. Mobile Mapping System)
GNSS	Globalni navigacijski satelitski sistemi
GIS	Geografski informacijski sistem

1 UVOD

V uvodnem delu predstavimo motivacijo, opredelimo cilje in probleme naloge. Na kratko opišemo strukturo naloge.

1.1 Motivacija

S sodobnimi tehnologijami postaja zajem podatkov vse bolj množičen, zato si prizadevamo za čim bolj samodejen in kakovosten način obdelave izvornih podatkov.

Samodejno iskanje vzorcev na digitalnih slikah lahko uporabimo pri najrazličnejših nalogah v industriji, računalništvu, medicini, geodeziji … Nas zanima uporaba v geodeziji, predvsem na področju fotogrametrije.

V zadnjih letih postajajo za množičen zajem prostorskih podatkov vedno bolj aktualen mobilni merski sistemi (ang. Mobile Mapping System, MMS). Mobilni merski sistem predstavlja združitev različnih tehnologij na eni platformi, ki se običajno premika. Ti sistemi so doživeli svoj razcvet z napredkom v kinematični tehnologiji določevanja položaja (GNSS), zasukov (INS) in drugih senzorskih sistemov (digitalni fotoaparati, lidarske naprave idr.) ter modernih komunikacijskih in prostorsko-informacijskih tehnologijah (GIS). Z integracijo naštetih tehnologij v enoten sistem lahko z mobilnim merskim sistemom pridobivamo podatke v realnem času, merimo objekte ter jih prostorsko upodobimo. Take sisteme uporabljamo na različnih področjih za pridobivanje merskih podatkov o prometni infrastrukturi (ceste, železnice, kolesarske poti ...), v inteligentnih transportnih sistemih in kot pomoč pri prostorskem načrtovanju (DFG Consulting d.o.o., 2012). Z MMS sistemi zajemamo ogromne količine podatkov v sorazmerno kratkem času. Zato je nadvse smotrno čim več postopkov avtomatizirati. Na digitalnih slikah, ki jih dobivamo v takih sistemih, lahko z računalniškimi algoritmi samodejno in tudi v realnem času, vektoriziramo podatke. Določamo, kje poteka os ceste, shranjujemo prostorske koordinate horizontalne in vertikalne prometne signalizacije ter drugih objektov, ki nas zanimajo.

Prav tako lahko avtomatsko prepoznavanje vzorcev uporabimo na letalskih posnetkih. Količine podatkov so ogromne in zato je smiselno uporabiti računalniške algoritme. Na letalskih posnetkih lahko iščemo horizontalno prometno signalizacijo, oslonilne oziroma vezne točke, objekte, ki jih je naredil človek itd.

1.2 Opredelitev problema, cilja in hipoteze naloge

Zaradi velike količine podatkov je smiselno nekatere procese avtomatizirati. Na digitalnih slikah lahko z različnimi programi samodejno iščemo vzorce in jih tudi prepoznamo ter lociramo. Eden od takšnih

vzorcev so prehodi za pešce. Prepoznavanje prehodov za pešce na posamezni digitalni sliki je pomembno iz več vidikov. Na tem področju je bilo že nekaj raziskav. Usmerjene so bile v odkrivanje prehodov za pešce v pomoč slabovidnim in slepim osebam (Se, 2000), v avtomobilski industriji pa za povečanje varnosti v sodobnih avtomobilih (Sichelschmidt, 2010). Predlagani algoritmi uporabljajo za odkrivanje prehodov za pešce naslednje metode: Fourierjevo transformacijo, razširjeno bipolarnost, ujemanje tarče, razmerje orientacije robov za klasifikacijo, iskanje normale z uporabo bežišča ter iskanje dveh bežišč. Glavna razlika med algoritmi za slabovidne in slepe osebe ter algoritmi, ki se uporabljajo v avtomobilski industriji, je v perspektivi digitalne slike, na kateri izvajamo iskanje prehodov za pešce. V prvi je ta iz perspektive pešca, ko se ta približuje prehodu, v drugi je iz perspektive avtomobila. Oboji algoritmi potekajo v realnem času.

V našem primeru bomo iskali prehod za pešce na sliki iz perspektive strehe vozila (mobilni merski sistemi) in iskanje ne bo potekalo v realnem času. Algoritem bo napisan splošno, tako da bi lahko deloval na različnih primerih. Parametri, ki jih je za potrebe izvedbe algoritma potrebno določiti in so tudi ključnega pomena za uspešno prepoznavanje, pa bodo primerni zgolj za izbrani primer. Želimo ugotoviti, kje na sliki se prehod za pešce nahaja, koliko belih ploskev ga sestavlja in izmeriti slikovne koordinate vseh vogalov označb. Vso programiranje bomo opravili z MatLab R2010a.

Glede na opisane cilje postavimo hipotezo: »Na digitalni sliki lahko s postopki digitalne obdelave, vektorizacije in transformacije samodejno odkrijemo prehode za pešce in izmerimo slikovne koordinate njihovih karakterističnih točk.«

1.3 Struktura naloge

V drugem poglavju podamo teoretična izhodišča za lažje razumevanje naloge. V tretjem poglavju opišemo razvoj algoritma in podrobneje razložimo posamezne operacije. Četrto in peto poglavje predstavljata analizo rezultatov in zaključek. Naloga vsebuje 6 prilog - osnovno programsko kodo in kodo petih funkcij.

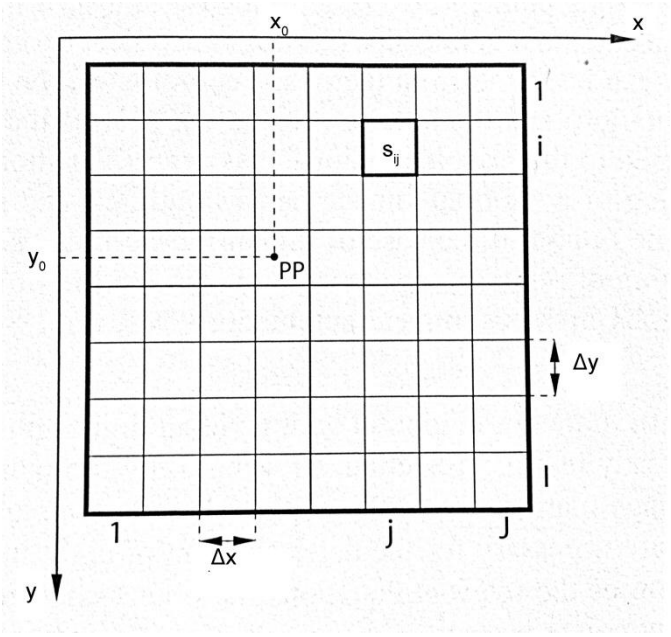
2 TEORETIČNA IZHODIŠČA

V tem poglavju opredelimo, kaj je in kako je sestavljena digitalna fotografija. Podamo osnovne principe obdelave fotografije.

2.1 Digitalna slika

Sliko lahko definiramo kot dvodimenzionalno funkcijo *f(x, y)*, kjer sta *x* in *y* prostorski koordinati in kjer je amplituda *s* intenziteta oziroma sivinska vrednost slike v določeni točki. Kadar imajo koordinate *x*, *y* in amplituda *s* končne in diskretne vrednosti, govorimo o digitalni sliki (Gonzales, Woods, Eddins, 2004).

Digitalna slika je torej matrika razsežnosti *I x J*, kjer vrednosti posameznih členov matrike predstavljajo intenziteto (s_{ij}). Ta lastnost slike nam omogoča ogromno možnosti v digitalni obdelavi slik (slika 1).



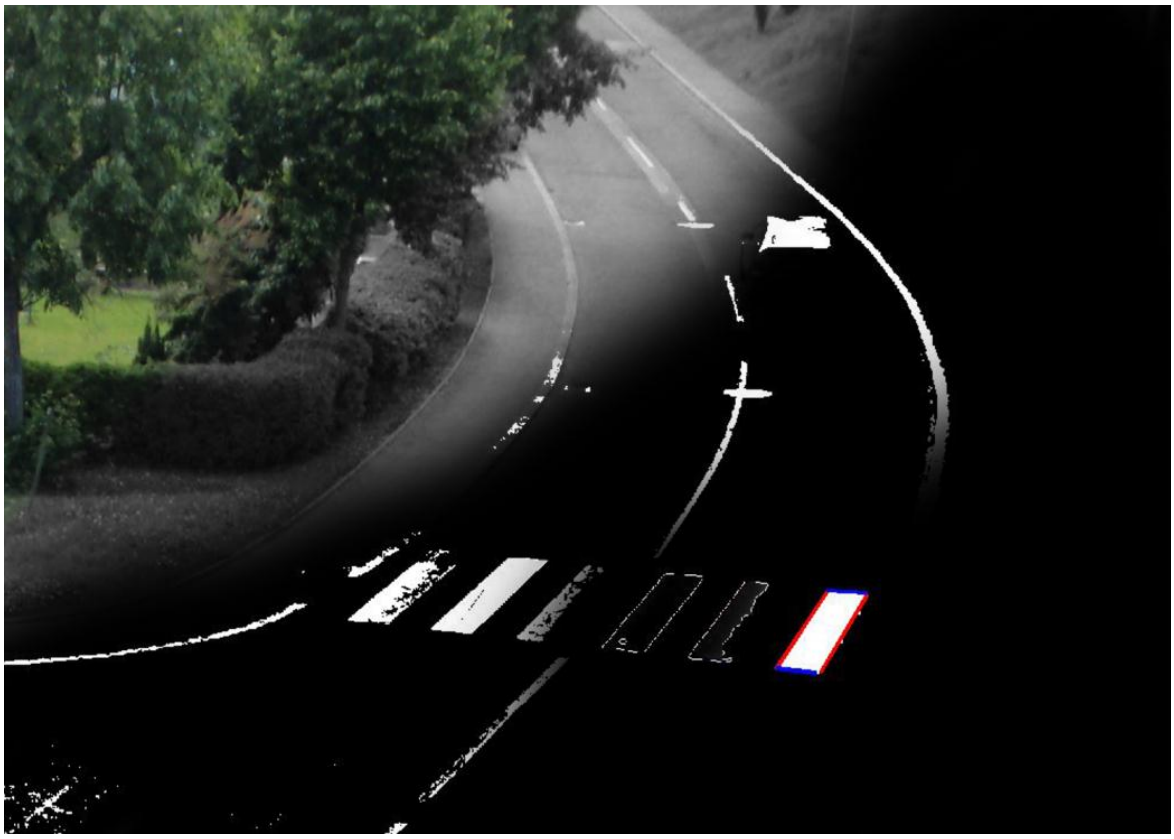
Slika 1: Digitalna slika (prirejeno po: Kraus, 2007: str 39)

2.2 Digitalna obdelava slik

Področje digitalne obdelave slik (angl. digital image processing) obsega različne metode in tehnike za izboljšanje slik (npr. odprava šuma, povečanje kontrasta), zaznavanje robov objektov na slikah, segmentacijo območij na slikah ipd.

Obdelavo lahko razdelimo na tri nivoje (slika 2) nizki, srednji in višji nivo (Gonzales, Woods, Eddins, 2004):

- Nizki nivo obdelave zajema primitivne operacije, kot je zmanjšanje šuma, poprava kontrasta in ostrenje.
- Srednji nivo obdelave zajema segmentacijo slike (na dele ali objekte), opise teh objektov, ki so primerni za računalniško obdelavo in klasifikacijo posameznih objektov. V srednjem nivoju je vhodni podatek slika, izhodni rezultati so atributi izločeni iz slike (robovi, obrisi in določitev identitete posameznih objektov).
- Višji nivo je tudi končni nivo in nam poda rezultate oziroma interpretacije. Sem sodi prepoznavanje objektov, vzorcev, analize in tudi izvajanje kognitivnih funkcij, ki jih običajno pripisujemo človeškemu vidu .



Slika 2: Prikazuje stopnje obdelave od nizkega do visokega nivoja (od leve proti desni)

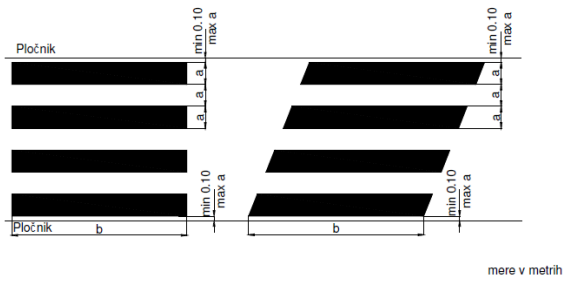
3 RAZVOJ ALGORITMA ZA AVTOMATSKO PREPOZNAVANJE PREHODA ZA PEŠCE NA DIGITALNI SLIKI

Predstavimo delovanje algoritma in podrobneje opredelimo posamezne pomembne operacije.

3.1 Idejna shema algoritma in vhodni podatki

Prehod za pešce je del vozišča, ki je namenjen prehajanju pešcev in je lahko označen (TSC, 2012):

- s pravokotniki ali paralelogrami, označenimi v celotni širini prehoda za pešce (slika 3),
- s pravokotniki ali paralelogrami, ki označujejo levi ali desni rob prehoda,
- z dodatnimi označbami z napisom ŠOLA in simbolom X, ki se označi pred in za napisom ŠOLA.



Slika 3: Prehod za pešce - pravokotniki ali paralelogrami, označeni v celotni širini (TSC, 2012: str 14)

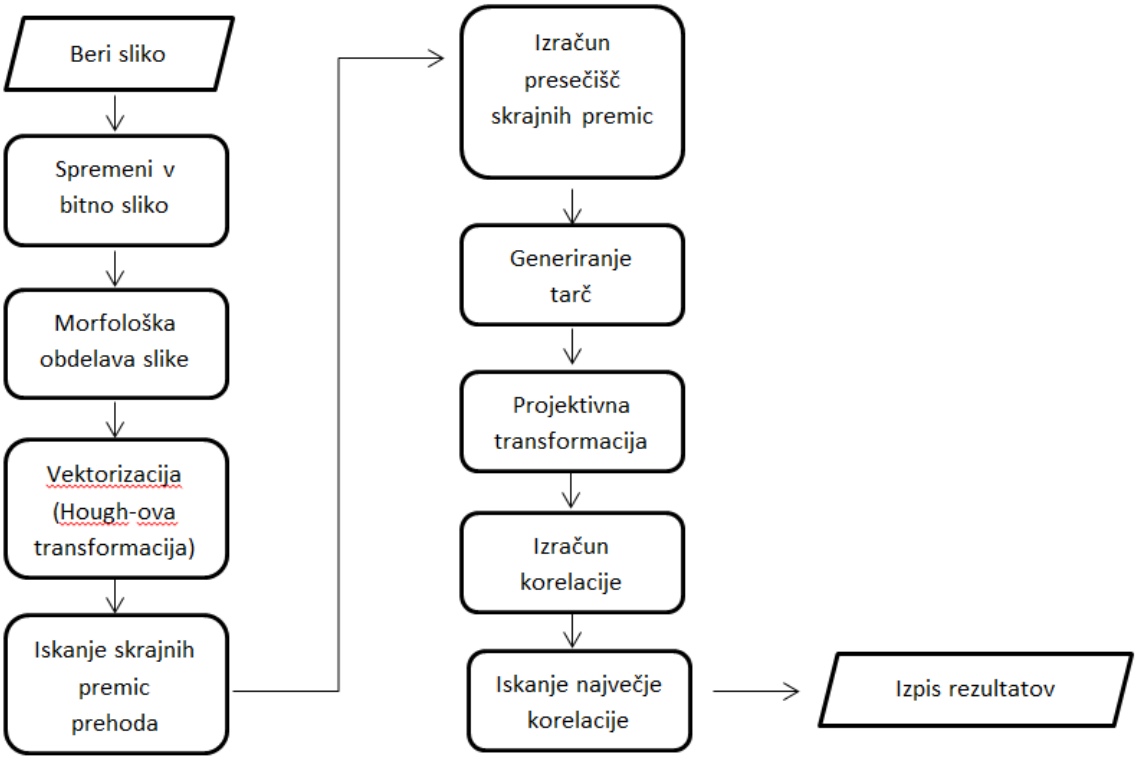
Posvetili se bomo prvemu tipu prehodov za pešce, prehodu pravokotne oblike. Razmerja med označbami niso točno določena, podane so le najmanjše dovoljene dolžini stranic. Posamezen pravokotnik prehoda za pešce imenujemo bela ploskev.

Za izvedbo iskanja prehoda za pešce na sliki smo posneli več slik. Za primer v diplomski nalogi smo izbrali samo eno (slika 4). Zato sliko smo se odločili zaradi razgibanosti. Na sliki so tanke sredinske črte, ki potekajo v različnih smereh. Prav tako imamo na vozišču vozilo, ena izmed belih ploskev prehoda za pešce je zbledela. Vse našteto nakazuje na običajen in realen primer.

Slika 4: Originalna fotografija

Naš cilj je najti prehod za pešce in ugotoviti, koliko belih ploskev ga sestavlja. Sliko je potrebno najprej pripraviti za iskanje, to pomeni, da s slike poskušamo odstraniti čim več nepotrebnih objektov, ki ne predstavljajo prehoda za pešce. To naredimo z morfološkimi operacijami. Prehod za pešce nato vektoriziramo in določimo štiri premice (štiri presečišča), ki ga omejujejo. Štiri presečišča nam predstavljajo referenčne točke za izračun transformacijskih parametrov, ki jih potrebujemo za transformacijo tarče. Tarčo transformiramo in računamo korelacijo. Tako ugotovimo, kateri tip prehoda za pešce se na sliki nahaja.

Algoritem vsebuje sintakso, ki deluje na vseh primerih. Vendar so parametri, ki vplivajo na razne operacije, nastavljeni zgolj za testni primer. Diagram poteka je prikazan na sliki 5.



Slika 5: Diagram poteka prepoznavanja vzorcev

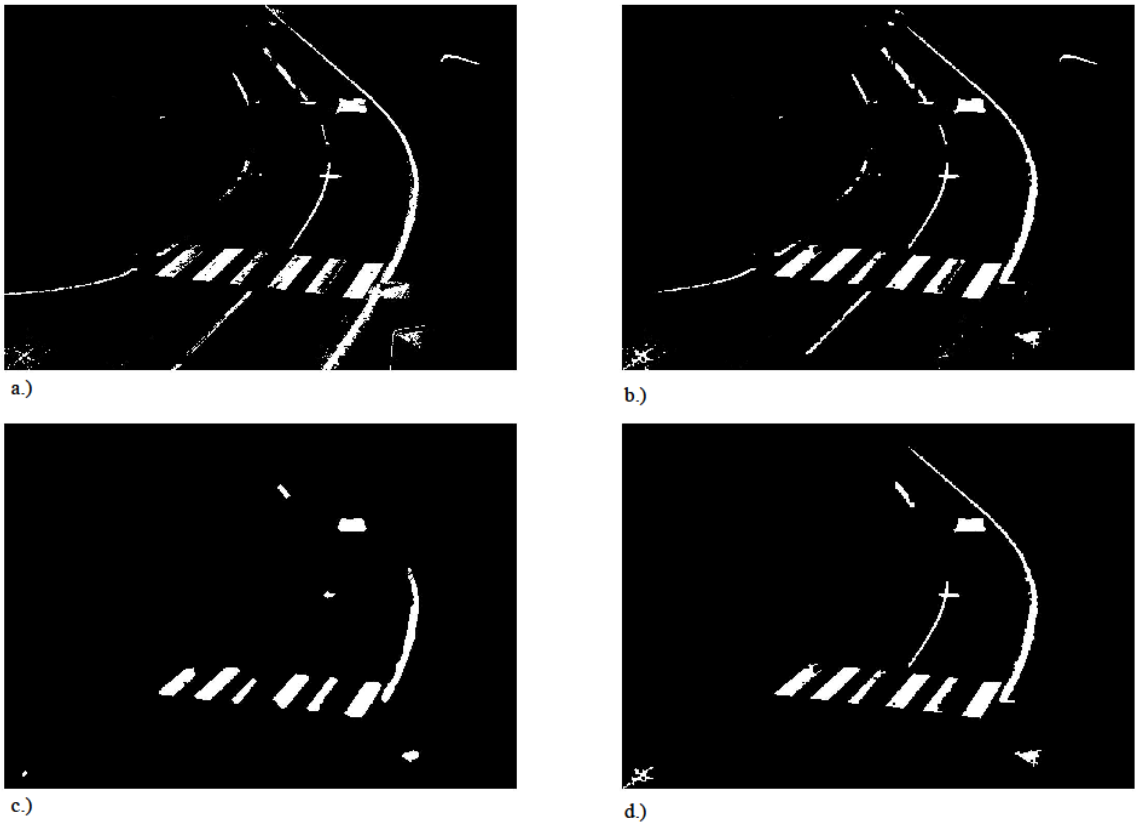
3.2 Morfološke operacije

Pri morfološki obdelavi slik govorimo o matematični morfologiji kot orodju za iskanje sestavnih delov slik, ki so pomembni za predstavitev in opis delov slik (obrisi, skeleti), in morfoloških tehnikah za pred- in poobdelavo slik, kot so morfološko filtriranje, tanjšanje ter omejevanje. Osnovni morfološki

operaciji sta dilacija in erozija, ki ju lahko združujemo tudi v kompleksnejše operacije, kot sta odpiranje in zapiranje slik (Gonzales, Woods, Eddins, 2004). Te operacije izvajamo s strukturnimi elementi in definiramo njihov vpliv na sliko (Priloga B).

Uporabljene morfološke operacije (slika 6):

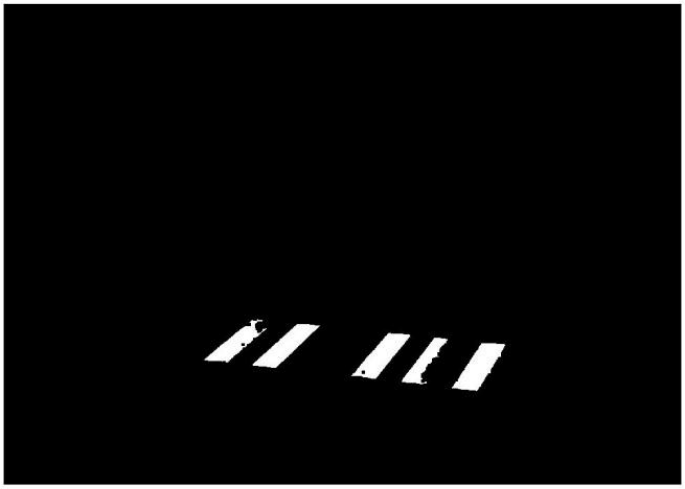
- zapiranje (dilacija, ki ji sledi erozija),
- odpiranje (erozija, ki ji sledi dilacija),
- morfološka rekonstrukcija.



Slika 6: Prikaz morfoloških operacij: a.) original (bitna slika), b.) zapiranje, c.) odpiranje, d.) morfološko odpiranje z rekonstrukcijo primera b.)

3.3 Odstranitev zelo velikih in zelo malih območij

Na sliki je ostalo še nekaj belih območij, ki niso prehodi za pešce. Te smo poskusili odstraniti z odstopanji njihove površine od povprečne površine posameznih območij na sliki 6 d.). Za vsako zaprto območje smo prešteli, koliko točk ga sestavlja. Izračunali smo povprečje vseh zaprtih območij in območja, ki niso bila na intervalu *[900<povprečje<2000]*, odstranili (Priloga A). Rezultat prikazuje slika 7.



Slika 7: Rezultat odstranitve zelo velikih in malih območij

3.4 Iskanje robov

Za potrebe vektorizacije s Houghovo transformacijo smo najprej morali na sliki poiskati obrise belih ploskev prehoda za pešce. Robove smo iskali na sliki 7. Odkrivanje deluje s pomočjo prvih in drugih odvodov. Rob oziroma linija se najverjetneje nahaja tam, kjer je prehod vrednosti intenzitet (sivinskih vrednosti) hiter. Prvi odvod sivinskih vrednosti na sliki je gradient (Gonzales, Woods, Eddins, 2004). Gradient dvodimenzionalne slike, *f(x,y)* je definiran kot vektor (1):

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{1}$$

Poiščemo magnitudo vektorja in izračun poenostavimo in aproksimiramo z uporabo absolutnih vrednosti (2):

$$\nabla f = \text{mag}(\nabla \mathbf{f}) \approx |G_x| + |G_y| \tag{2}$$

Osnovna značilnost gradienta vektorja je, da kaže v smeri največje spremembe *f* na koordinatah *(x,y).* Kot največje spremembe izračunamo (3) (Gonzales, Woods, Eddins, 2004):

$$\alpha(x,y) = tan^{-1}\left(\frac{G_x}{G_y}\right) \tag{3}$$

Za iskanje robov smo uporabili Cannyjev algoritem (Slika 8).

Slika 8: Najdeni obrisi prehoda s Cannyevim algoritmom

Cannyjev detektor je najmočnejši detektor robov v funkciji *edge*. Metodo lahko povzamemo v naslednji korakih:

- Odstranitev šuma z uporabo Gaussovega filtra s standardno deviacijo σ.
- Lokalni gradient (2) in smeri robov (3) se izračunajo v vsaki točki. Robna točka je določena kot točka, ki je lokalni maksimum v smeri gradienta.
- Robne točke ustvarijo greben na sliki magnitude gradienta. Algoritem nato zapiše vse točke, ki so na grebenu in da tistim, ki niso, vrednost nič. S pomočjo dveh pragov *T1* in *T2*, kjer je *T1<T2* določimo, katere so močne robne točke in katere točke so šibke. Tiste, ki imajo vrednost večjo od *T2*, so močne robne točke in tiste, ki imajo manjšo od *T1*, so šibke.
- Algoritem nato poveže točke in vključi tudi šibke točke, ki imajo 8 povezav do močnih točk.

3.5 Houghova transformacija

Houghovo transformacijo smo uporabili za vektorizacijo stranic prehoda za pešce. S to transformacijo obravnavamo točko (x_i, y_i) in vse premice, ki potekajo skozi njo. Skozi vsako točko poteka neskončno mnogo premic, ki jih opišemo z linearno enačbo premice (4).

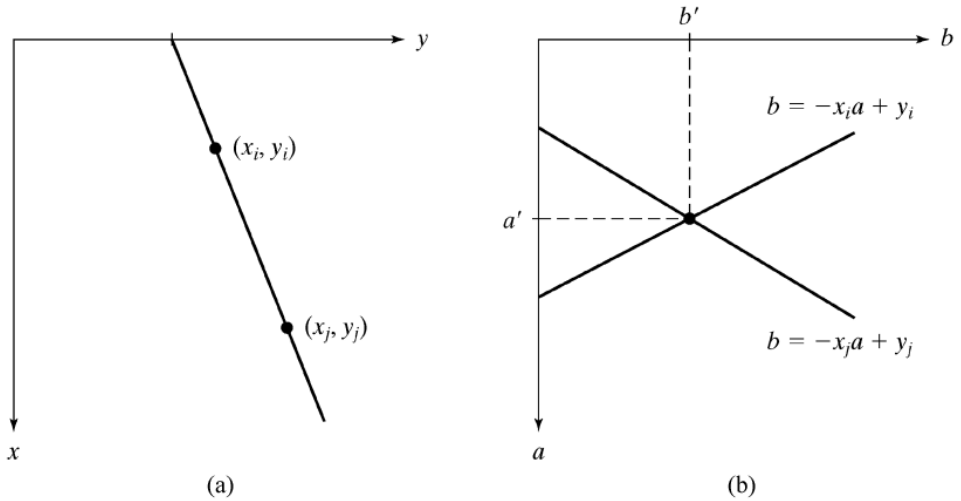
$$y_i = k\, x_i + n \tag{4}$$

Če enačbo zapišemo v obliki (5):

$$n = -k\, x_i + y_i \tag{5}$$

dobimo parametrski prostor (ravnino *kn*). Tudi druga točka (x_j, y_j) ima premico in svoj parametrični prostor. Ta premica seka premico (x_i, y_i) v k' in n'. Kjer je k' smerni količnik in n' presečišče premice,

ki povezuje (x_i, y_i) in (x_j, y_j) v xy - ravnini (Gonzales, Woods, Eddit, 2004). Slika 9 prikazuje koncept Houghove transformacije.

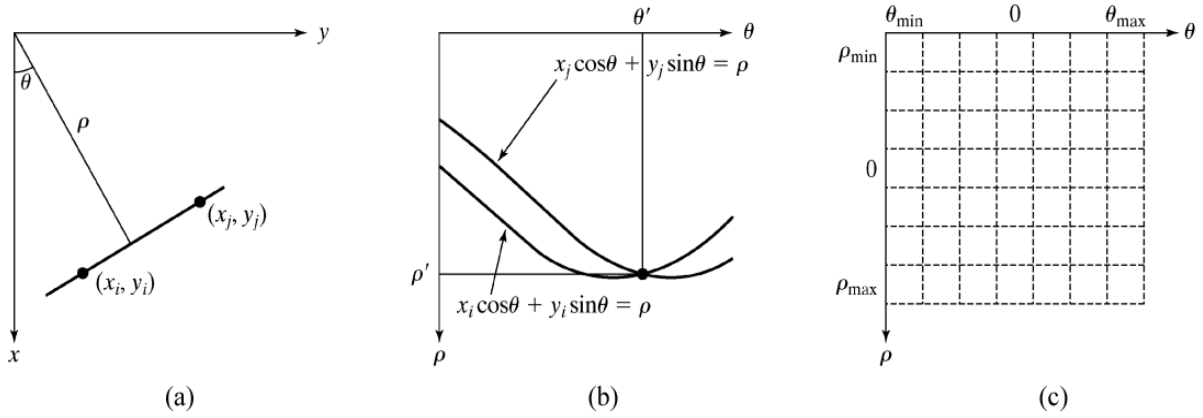


Slika 9: Ravnina xy (a) in parametrski prostor ab (b) (Gonzales, Woods, Eddit, 2004: str. 394)

Linije na sliki lahko določimo tako, da poiščemo, kje se veliko število premic parametričnega prostora seka. Vendar, ko se premica bliža vertikali, se *k* bliža neskončnosti. Problem je rešen tako, da premico zapišemo v normalni obliki (6):

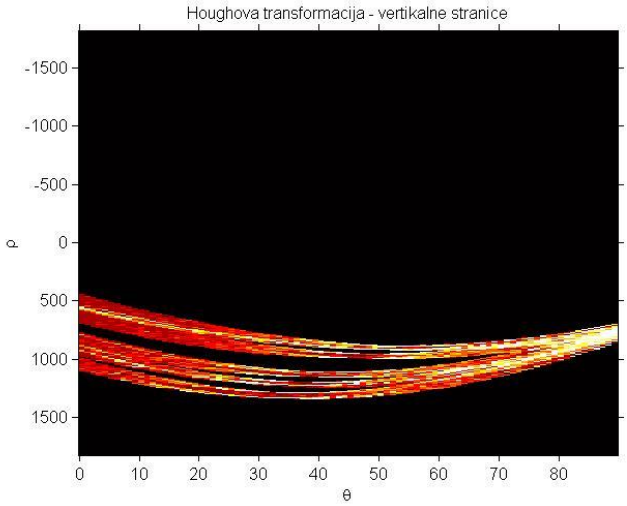
$$x\cos\theta + y\sin\theta = \rho \tag{6}$$

Slika 10 prikazuje geometrično predstavitev parametrov ρ in θ (Gonzales, Woods, Eddit, 2004).

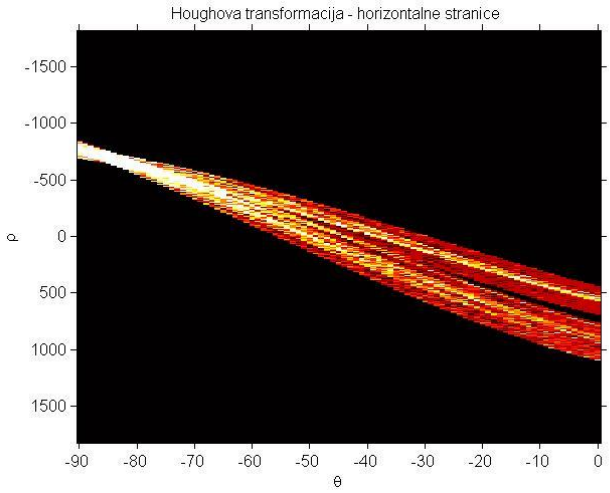


Slika 10: Parametrski prostor Houghove transformacije (Gonzales, Woods, Eddit, 2004: str. 394)

Slika 11 prikazuje geometrično predstavitev Houghove transformacije, na primeru testne slike, za iskanje vertikalnih stranic prehodov za pešce. V funkciji smo določili, na katerem intervalu se lahko nahaja θ. Slika 12 prikazuje geometrično predstavitev Houghove transformacije horizontalnih stranic prehodov za pešce.



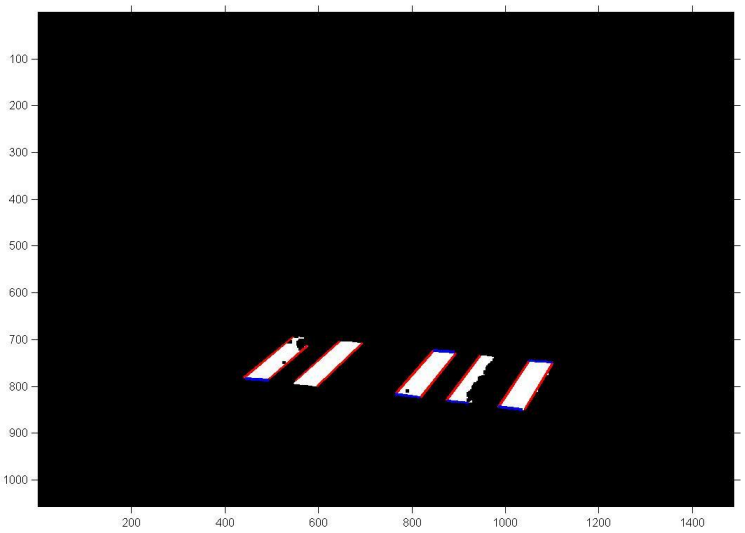
Slika 11: Houghova transformacija - vertikalne stranice



Slika 12: Houghova transformacija - horizontalne stranice

Uporabili smo še funkcijo *houghpeaks*, ki poišče lokalne maksimume v Houghovi transformaciji. V našem primeru se je zelo uporaben izkazal ukazal '*NHoodSize*', ki poskrbi, da ko funkcija *houghpeaks* najde lokalni maksimum, izbriše vse ostale lokalne maksimume na območju, ki ga mi določimo. Preden smo uporabili ta ukaz, smo dobili več vzporednih daljic, ki so predstavljale isto stranico. Funkcija *houghpeaks* vrne matriko s stolpcem in vrstico, v katerem se nahaja maksimum (Priloga C).

Funkcija *houghlines* potrebuje za vhodne podatke matriko maksimumov ter parametra ρ in θ, da tvori daljice med posameznimi točkami. Vrne slikovne koordinate začetne in končne točke vsake daljice (slika 13).



Slika 13: Rezultati Houghove transformacije

3.6 Določitev skrajnih oglišč prehoda za pešce

Določiti je bilo treba štiri skrajne točke prehoda za pešce. Za izračun teh smo izračunali enačbe premice, ki omejujejo prehod. Posebej smo določili premici, ki se nahajata skrajno levo oziroma desno ter premici, ki se nahajata skrajno spodaj oziroma zgoraj.

Skrajno levo/desno oziroma vertikalne premice smo določili s pomočjo izračuna enačbe premic iz koordinat, ki smo jih dobili iz Houghove transformacije (7), (8):

$$k = \frac{\Delta vrstica}{\Delta stolpec} \tag{7}$$

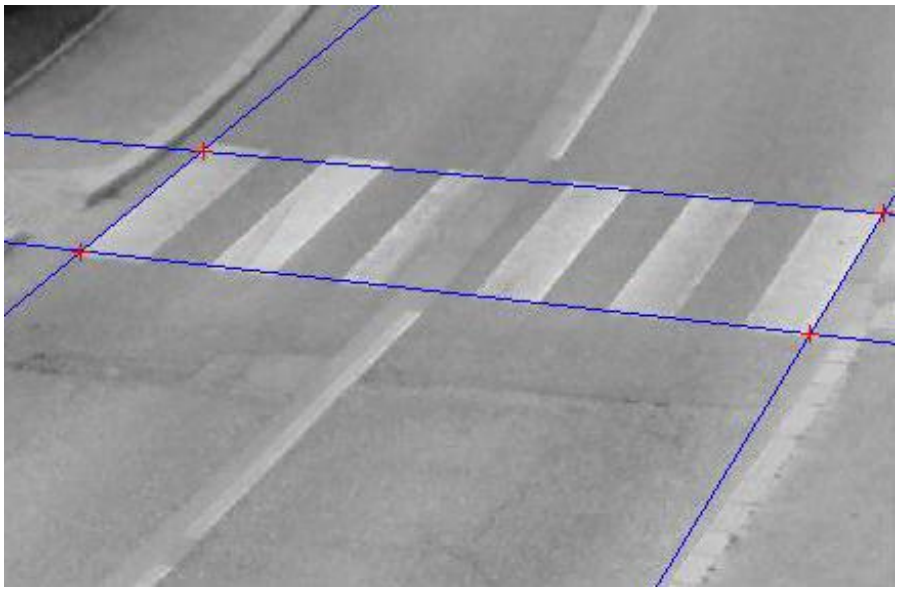
$$n = vrstica_i - k\ stolpec_i \tag{8}$$

Robni premici sta tisti dve, ki imata največji in najmanjši *n*.

Določitev horizontalne premice je bila zahtevnejša. Poiskali smo vse točke, ki se nahajajo na zgornjem robu, in jih zapisali v vektor ter isto naredili za spodnje točke (Priloga D). Skozi točke smo nato izračunali linearno regresijo (9) :

$$\mathbf{B} = \mathbf{X}\backslash\mathbf{Y} \quad \rightarrow \quad \begin{bmatrix} k \\ n \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_n & 1 \end{bmatrix} \backslash \begin{bmatrix} y_1 \\ y_n \end{bmatrix} \tag{9}$$

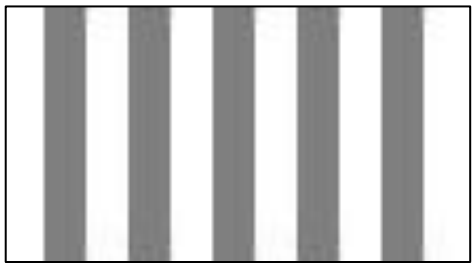
Dobili smo vse štiri premice, ki omejujejo prehod in njihova presečišča (Slika 14).



Slika 14: Najden prehod za pešce

3.7 Generiranje tarče

Tarčo generiramo sproti v programu, saj prehodi za pešce nimajo vedno enakega števila belih ploskev (slika 15). Program je napisan tako, da generira prehode za pešce od treh do dvajset belih ploskev, za vsakega izračuna transformacijske parametre, ga transformira in nato izračuna korelacijo. Tarčo generiramo iz dveh pomožnih matrik velikosti 60 x 10. Prva pomožna matrika predstavlja belo območje prehoda za pešce (vrednost intenzitete = 255). Druga pomožna matrika predstavlja območje, kjer ni bele ploskve. Tam je asfalt, odločili smo se za sivo barvo, ki je najbolj podobna asfaltu (vrednost intenzitete = 127). Pomožni matriki nato zaporedno zlagamo, da dobimo primerno tarčo (Priloga E).



Slika 15: Primer generirane tarče s 6-imi belimi ploskvami

3.8 Projektivna transformacija

Za potrebe izračuna korelacije moramo tarčo transformirati. Za izračun transformacijskih parametrov potrebujemo koordinate točk na sliki, ki se ujemajo s točkami tarče. V našem primeru so to vogalne točke prehoda za pešce, ki smo jih dobili z izračunom presečišč premic, ter vogali tarče.

Izračun transformacijskih parametrov smo opravili s funkcijo *cp2tform* (Priloga F). Izbrali smo projektivno transformacijo. Predpostavimo lahko, da prehod leži na ravnini. To nam omogoča uporabo projektivne transformacije ravnine.

Vsi objekti ležijo na isti ravnini. Iz enačbe projektivne transformacije lahko trdimo, da (10):

- ena slika zadostuje za izračun transformacijskih parametrov ravninskega objekta,
- osem neodvisnih parametrov definira centralno projekcijo ravninskega objekta.

$$X_t = \frac{a_1 X_s + a_2 Y_s + a_2}{c_1 X_s + c_2 Y_s + 1} \qquad Y_t = \frac{b_1 X_s + b_2 Y_s + b_2}{c_1 X_s + c_2 Y_s + 1} \tag{10}$$

V enačbi 10:

X_t in Y_t sta koordinate objekta, v našem primeru koordinate vogalov tarče,

X_s in Y_s sta slikovne koordinate,

$a_1, a_2, a_3, b_1, b_2, b_3, c_1$ in c_2 so transformacijski parametri projektivne transformacije.

Transformacijske parametre najdemo s pomočjo referenčnih točk. To so točke na sliki ter na objektu, katerih koordinate poznamo.

Funkcija *imtransform* transformira tarčo na podlagi parametrov (slika 16).

Slika 16: Transformirana tarča

3.9 Slikovno ujemanje

Algoritmi za slikovno ujemanje rešujejo naloge odkrivanja ustreznih območij na dveh slikah, običajno narejenih iz različnih položajev (Kraus, 2007). Ena od dveh slik je lahko umetno ustvarjena, kot je tudi v našem primeru. Ta slika je prehod za pešce, poimenujemo jo tudi tarča. Druga slika je slika prehoda za pešce in je slika, na kateri iskanje izvajamo.

Pri običajnih nalogah slikovnega ujemanja imamo določeno velikost tarče ter iskalno območje na sliki, ki je večje od tarče. Tarčo nato pomikamo po celotnem iskalnem območju na sliki in na vsakem položaju izračunamo korelacijski koeficient. Kjer je ta največji, tam se najverjetneje nahaja iskan vzorec.

V našem primeru že poznamo lokacijo vzorca, ki ga iščemo. Ugotovili smo jo z vektorizacijo. Zanima nas, katera izmed tarč se ujema s sliko. Vsaka tarča predstavlja različni tip prehoda za pešce (Priloga F).

Največje ujemanje tarče ugotovimo z računanjem korelacijskega koeficienta *r* (11). Izračun je izpeljan iz standardnega odklona σ_r in σ_s, sivinskih vrednosti s_1 in s_2 na obeh slikah in kovariance σ_{rs} med sivinskimi vrednostmi teh dveh slik (Kraus, 2007).

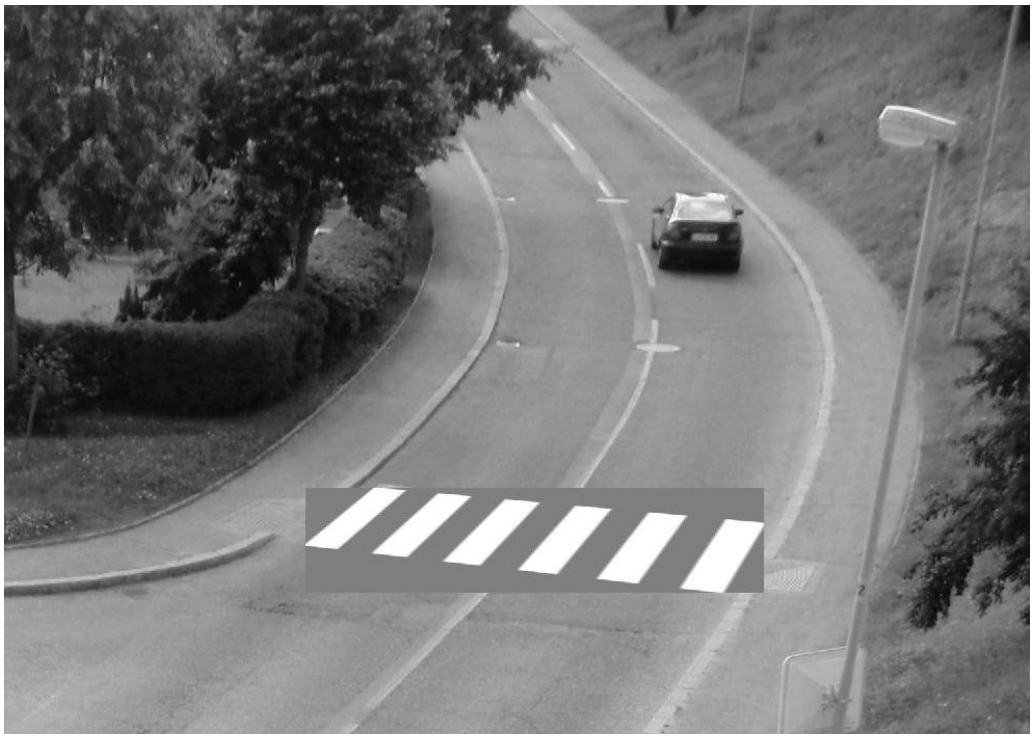
$$r = \frac{\sigma_{12}}{\sigma_1 \sigma_2} = \frac{\sum(s_1 - \bar{s}_1)(s_2 - \bar{s}_2)}{\sqrt{\sum(s_1 - \bar{s}_1)^2 \sum(s_2 - \bar{s}_2)^2}} = \frac{\sum s_1 s_2 - n\bar{s}_1\bar{s}_2}{\sqrt{(\sum s_1^2 - n\bar{s}_1^2)(\sum s_2^2 - n\bar{s}_2^2)}} \tag{11}$$

3.10 Rezultati

Transformirano tarčo smo nalepili na izvorno sliko (slika 17) ter podali slikovne koordinate vogalov prehoda za pešce (preglednica 1).

Preglednica 1: Slikovne koordinate vogalov prehoda za pešce

Vogali	Stolpec [pix]	Vrstica [pix]
Levo zgoraj	541	698
Levo spodaj	441	781
Desno zgoraj	1102	749
Desno spodaj	1041	849



Slika 17: Ugotovljena tarča, prilepljena na originalno sliko

4 ANALIZA REZULTATOV

Na digitalni sliki smo uspeli določiti položaj prehoda za pešce in ugotoviti, iz koliko črt je sestavljen. Program deluje le na konkretnem primeru. Lokacijo bi lahko še nekoliko izboljšali, če bi poskusili s slikovnim ujemanjem in nato metodo najmanjših kvadratov določiti najboljši položaj ujemanja. Vendar se je smiselno vprašati, ali je boljša natančnost določitve položaja potrebna za konkreten namen uporabe (najti in prepoznati vzorce na sliki).

Problem bi nastal, če skrajni beli ploskvi prehoda za pešce na sliki ne bi bili dobro vidni (npr. bi bili delno zabrisani). V tem primeru bi prepoznali napačen vzorec prehoda za pešce.

Uspelo nam je razviti algoritem (Priloge A,B,C,D,E,F), ki omogoča iskanje prehodov za pešce na sliki, kar je tudi bil naš cilj. Hipotezo lahko potrdimo in zaključimo, da smo uspešno dosegli zastavljene cilje.

5 ZAKLJUČEK

V diplomski nalogi smo si zadali cilj, da razvijemo algoritem, ki bo na digitalni sliki samodejno poiskal prehode za pešce. Z morfološkimi operacijami, Houghovo transformacijo, projektivno transformacijo ter slikovnim ujemanjem smo uspeli izdelati program, ki uspešno najde prehode za pešce in nam tudi pove, koliko belih ploskev sestavlja prehod. Seveda program še zdaleč ni popolnoma univerzalen, torej ni uporaben za vse primere, za kar bi bilo potrebno vložiti še veliko truda. Zaradi omejenega obsega naloge ni bilo možno rešiti vseh možnih primerov. Možnosti za optimizacijo programa je tako še veliko.

Vsekakor pa izdelan algoritem predstavlja dobro izhodišče za nadaljnje delo na tem področju.

VIRI

DFG Consulting d.o.o. 2012. http://www.dfgcon.si/index.php/dejavnosti/mobilno-kartiranje (Pridobljeno 17.7.2012.)

Gonzales, R. C., Woods, R. E., Eddins, S. L. 2004. Digital image processing using Matlab. New Jersey, Pearson Prentice-Hall: 609 str.

Kraus, K. 2007. Photogrammetry: Geometry from images and laser scans. Second Edition. Berlin, Walter de Gruyter GmbH & Co: 459 str.

Se, S. 2000. Zebra-crossing Detection for the Partially Sighted. Department of Computer Science, University of British Columbia. Vancouver, B.C. Canada. http://research.stephense.com/papers/cvpr00.pdf (Pridobljeno 29.7.2012.)

Sichelschmidt, S. Haselhoff, A. Kummert, A. Roehder, M. Elias, B. Berns, K. 2010. Pedestrian Crossing Detecting as a part of an Urban Pedestrian Safety System. V: 2010 IEEE Intelligent Vehicles Symposium. University of California, San Diego, CA, USA, Junij 21-24, 2010. p. 840 – 844.

TSC 02.401 : 2012 - Označbe na vozišču oblika in mere. http://www.dc.gov.si/fileadmin/dc.gov.si/pageuploads/pdf_datoteke/TSC/TSC_02.401-2012.pdf (Pridobljeno 30.7.2012.)

Ostali viri

Grigillo, D. 2009. Samodejno odkrivanje stavb na visokoločljivih slikovnih virih za potrebe vzdrževanja topografskih podatkov. Doktorska disertacija. Ljubljana, Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo (samozaložba D. Grigillo): 156 str. http://eprints.fgg.uni-lj.si/781/1/GED_0197_Grigillo.pdf (Pridobljeno 30.7.2012.)

SEZNAM PRILOG

PRILOGA A: GLAVNA PROGRAMSKA KODA

```
%Samodejno prepoznavanje prehodov za pešce na digitalni fotografiji
%Diplomsko delo
%Matlab R2010a 7.10.0
%Jernej Nejc Dougan, julij 2012
clc;
clear all;
%Podatki
sg=rgb2gray(imread('test.tif'));
%Morfološke obdelava slike
s=morfo(sg);
%Odstranitev območij, ki odstopajo od sredine
[L, n]=bwlabel(s);
vsota=zeros(1,n);
for k=1:n
        vsota(k)=sum(sum(L==k));
end
pop=mean(vsota);
for k=1:n
        if vsota(k)<(pop-900) || vsota(k)>(pop+2000);
        L(L==k)=0;
        end
end
s=im2bw(L);
%Iskanje obrisov prehodov za pešce
s_robovi=edge(s,'canny');
%Vektorizacija
%Vertikalne daljice
theta=0.3:90; %usmerjenost premic za vektorizacijo
treshold=0.3; %vrednost, ki še ne predstavlja vrha
nhoodsize=[41 41]; %območje na katerem pobriše lokalne maksimume
peaks=9; %število vrhov
fillgap=70; %zapolni luknjo dolžine
minlength=100; %minimalna dolžine deljice
daljice_v=vektorizacija(s_robovi,theta,treshold,nhoodsize,peaks,fillgap,minlength);
%Horizontlane daljice
theta=-90:0; %usmerjenost premic za vektorizacijo
treshold=0.1; %vrednost, ki še ne predstavlja vrha
nhoodsize=[51 51]; %območje na katerem pobriše lokalne maksimume
peaks=6; %število vrhov
fillgap=25;
minlength=45;
daljice_h=vektorizacija(s_robovi,theta,treshold,nhoodsize,peaks,fillgap,minlength);
%Izračun premic, ki omejujejo prehod za pešce reft=reftocke(daljice_v, daljice_h);
%Generiranje tarče, transformacija ter izračun korelacije
%Izrežemo del slike na katerem se nahaja prehod za pešce
dS=sg(min(round(reft(:,2))):max(round(reft(:,2))),min(round(reft(:,1))):max(round(reft(:,
1
))));
st_zeber=15; %Koliko belih ploskev sestavlja prehod za pešce
R=zeros(st_zeber-2,1);
%Izračun korelacijskih koeficientov za vse prehode za pešce
for i=3:st_zeber
        R(i-2)=korelacija(i,dS,reft);
end
%Iskanje največjega ujemanja
```

```
zebra=(find(R==max(R)))+2;
maxR=max(R);
%Generiranje prave zebre
[tarca,vogali_t]=gentarce(zebra);
%Transformacija
[r,c,]=size(dS);
tform=cp2tform(vogali_t,reft,'projective');
tt=imtransform(tarca,tform,'size',[r,c],'FillValues',127);
%Prilepi transformirano tarčo na originalno sliko in jo prikaži
sg(min(round(reft(:,2))):max(round(reft(:,2))),min(round(reft(:,1))):max(round(reft(:,1))
))=
tt;
figure(1), imshow(sg,'InitialMagnification',67);
axis on;
%Izpis
fprintf('********************************************************************** \n')
fprintf('* * \n')
fprintf('* Samodejno prepoznavanje prehodov za pešce na digitalni fotografiji * \n')
fprintf('* Diplomsko delo * \n')
fprintf('* Matlab R2010a 7.10.0 * \n')
fprintf('* Jernej Nejc Dougan, julij 2012 * \n')
fprintf('* * \n')
fprintf('********************************************************************** \n\n\n')
fprintf('REZULTATI \n\n')
fprintf('Slikovne koordinate karakterističnih točk prehoda za pešce [pix]: \n')
fprintf('Stolpec: \t|\t Vrstica: \n')
for i=1:length(reft)
        fprintf('%5.1f \t\t|\t %5.1f \n', reft(i,1), reft(i,2))
end
```

PRILOGA B: FUNKCIJA ZA MORFOLOŠKE OPERACIJE

```
function [s]=morfo(sivinska_slika)
%Funkcija z uporabo morfoloških operaciji (zapiranje, odpiranje, dilacija,
%erozija ter morfolška rekonstrukcija) s slike odstrani motnje na sliki.
s=sivinska_slika;
s=im2bw(s, 0.7);
s=imclearborder(s); %počisti robove slike
s=imclose(s,strel('square',7));
s=bwareaopen(s, 30); %zapolni bele pike
s=~bwareaopen(~s, 30); %zapolni črne pike
%odstranitev tankih objektov
dim_o=10;
m=s;
sep=eye(dim_o);
m=imopen(m,sep);
sep2=flipud(eye(dim_o));
m=imopen(m,sep2);
s=imreconstruct(m,s,4);
end
```

PRILOGA C: FUNKCIJA ZA VEKTORIZACIJO

```
function[daljice]=vektorizacija(s_robovi,theta,treshold,NhoodSize,peaks,fil
lgap,minlength)
%Funkcija vektorizira sliko s Houghovo transformacijo
[H,T,R] = hough(s_robovi,'RhoResolution',1,'Theta',theta);
P = houghpeaks(H,peaks,'threshold',ceil(treshold*max(H(:))),
'NHoodSize',NhoodSize);
daljice =
houghlines(s_robovi,T,R,P,'FillGap',fillgap,'MinLength',minlength);
end
```

PRILOGA D: FUNKCIJA ZA IZRAČUN REFERENČNIH TOČK

```
function [reft]=reftocke(daljice_v, daljice_h)
%Določi skrajne premice, ki omejujejo prehod za pešce in izračuna
%referenčne točke za transformacijo
%Določitev zgornjih-spodnjih ogljišč vertikalnih daljic
for i=1:length(daljice_v)
        xy1 = [daljice_v(i).point1; daljice_v(i).point2];
        if xy1(1,2)>xy1(2,2)
                xy_v_sp(i,1)=xy1(1,1);
                xy_v_sp(i,2)=xy1(1,2);
                xy_v_zg(i,1)=xy1(2,1);
                xy_v_zg(i,2)=xy1(2,2);
        else
                xy_v_sp(i,1)=xy1(2,1);
                xy_v_sp(i,2)=xy1(2,2);
                xy_v_zg(i,1)=xy1(1,1);
                xy_v_zg(i,2)=xy1(1,2);
        end
        % plot(xy_v_zg(i,1),xy_v_zg(i,2),'y+');
        % plot(xy_v_sp(i,1),xy_v_sp(i,2),'g+');
end
%Izračun smernega količnika k in n vertikalnih daljic.
premica_v=zeros(length(daljice_v),2);
for j=1:length(daljice_v)
        xy1 = [daljice_v(j).point1; daljice_v(j).point2];
        k=(xy1(2,2)-xy1(1,2))/(xy1(2,1)-xy1(1,1));
        n=xy1(1,2)-k*xy1(1,1);
        premica_v(j,1)=k;
        premica_v(j,2)=n;
        %refline(premica_v(j,1),premica_v(j,2));
end
%Izračun smernega količnika k in n horizontalnih daljic.
premica_h=zeros(length(daljice_h),2);
for j=1:length(daljice_h)
        xy1 = [daljice_h(j).point1; daljice_h(j).point2];
        k=(xy1(2,2)-xy1(1,2))/(xy1(2,1)-xy1(1,1));
        n=xy1(1,2)-k*xy1(1,1);
        premica_h(j,1)=k;
        premica_h(j,2)=n;
        %refline(premica_h(j,1),premica_h(j,2));
end
%Izračunaj povprečni k in n vseh horizontalnih premic za ugotovitev katere so zgoraj katere
pa spodaj
%PRIMER DA SO SAME SPODNJE/ZGORNJE ERROR!
pm=[mean(premica_h(:,1)) mean(premica_h(:,2))];
%refline(pm(1),pm(2));
spodaj=1;
zgoraj=1;
for i=1:length(premica_h)
        xy1 = [daljice_h(i).point1; daljice_h(i).point2];
        k1=(xy1(1,2)-xy1(2,2))/(xy1(1,1)-xy1(2,1));
        n1=xy1(1,2)-k*xy1(1,1);
        ypm=pm(1)*xy1(1,1)+pm(2);
        yh=k1*xy1(1,1)+n1;
        if yh>ypm %je spodaj
                xy_h_sp(spodaj, 1)=xy1(1,1);
```

```
                xy_h_sp(spodaj, 2)=xy1(1,2);
                xy_h_sp(spodaj+1, 1)=xy1(2,1);
                xy_h_sp(spodaj+1, 2)=xy1(2,2);
                spodaj=spodaj+2;
        else %je zgoraj
                xy_h_zg(zgoraj, 1)=xy1(1,1);
                xy_h_zg(zgoraj, 2)=xy1(1,2);
                xy_h_zg(zgoraj+1, 1)=xy1(2,1);
                xy_h_zg(zgoraj+1, 2)=xy1(2,2);
                zgoraj=zgoraj+2;
        end
end
%združitev koordinat horizontalnih in poševnih premic
[rv,~]=size(xy_v_zg);
[rh,~]=size(xy_h_zg);
xy_zg(1:rv,:)=xy_v_zg;
xy_zg(rv+1:rv+rh,:)=xy_h_zg;
[rv,~]=size(xy_v_sp);
[rh,~]=size(xy_h_sp);
xy_sp(1:rv,:)=xy_v_sp;
xy_sp(rv+1:rv+rh,:)=xy_h_sp;
%Izračun linearne regresije
%Zgornja premica
X=[xy_zg(:,1),ones(length(xy_zg),1)];
Y=xy_zg(:,2);
B(1,:)=X\Y;
%Spodnja premica
X=[xy_sp(:,1),ones(length(xy_sp),1)];
Y=xy_sp(:,2);
B(2,:)=X\Y;
%skrajne vertikalne premice
A(1,:)=premica_v(premica_v(:,2)==max(premica_v(:,2)),1:2);
A(2,:)=premica_v(premica_v(:,2)==min(premica_v(:,2)),1:2);
%izračun presečišča
st=1;
tts=zeros(4,2);
for i=1:length(A)
        for j=1:length(B)
                tts(st,1)=(B(j,2)-A(i,2))/(A(i,1)-B(j,1));
                tts(st,2)=B(j,1)*tts(st)+B(j,2);
                st=st+1;
        end
end
reft(1,:)=tts(tts(:,2)==min(tts(:,2)),:);
reft(2,:)=tts(tts(:,1)==min(tts(:,1)),:);
reft(3,:)=tts(tts(:,1)==max(tts(:,1)),:);
reft(4,:)=tts(tts(:,2)==max(tts(:,2)),:);
end
```

PRILOGA E: FUNKCIJA ZA GENERIRANJE TARČE

```
function [T, vogali_t]=gentarce(stevilo_belih_ploskev)
%Funkcija generira tarčo prehoda za pešce
%GENERIRANJE TARČE
z1=ones(60,10)*255;
z0=ones(60,10)*127;
tarca=z1;
for i=1:(stevilo_belih_ploskev-1)
        [~,c]=size(tarca);
        tarca(:,c+1:c+10)=z0;
        tarca(:,c+11:c+20)=z1;
end
T=uint8(tarca);
%figure(2), imshow(tarca)
%vogalne točke
[r,c]=size(T);
vogali_t=[1, 1;
1, r;
c, 1;
c, r];
end
```

PRILOGA F: FUNKCIJA ZA IZRAČUN SLIKOVNEGA UJEMANJA

```
function R=korelacija(stevilo_belih_ploskev,dS,referencne_tocke)
%Funkcija generira število željenih belih ploskev in izračuna korelacijo s sliko
reft=referencne_tocke;
%GENERIRANJE TARČE
[T,vogali_t]=gentarce(stevilo_belih_ploskev);
%TRANSFORMACIJA
dS=double(dS);
[r,c,]=size(dS);
tform=cp2tform(vogali_t,reft,'projective');
tt=imtransform(T,tform,'size',[r,c],'FillValues',127);
tt=double(tt);
%KORELACIJA
[vt,st]=size(tt);
n=vt*st;
%izračun korelacijskega faktorja
R=(sum(sum(tt.*dS))-n*mean2(tt)*mean2(dS))/sqrt((sum(sum(tt.*tt))-
n*mean2(tt)^2)*(sum(sum(dS
.*dS))-n*mean2(dS)^2));
end
```