UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Zadnikar

Zaznavanje objektov na topografskih kartah

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Tomaž Dobravec

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

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

Št. naloge: 00449/2009
Datum: 05.04.2009

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

Kandidat: **ANDREJ ZADNIKAR**

Naslov: **ZAZNAVANJE OBJEKTOV NA TOPOGRAFSKIH KARTAH**
DETECTION OF STRUCTURES ON TOPOGRAPHIC MAPS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Preglejte področje računalniške grafike, ki se ukvarja z zaznavanjem objektov na sliki, ter predstavite osnovne metode za avtomatsko obdelavo slik (konvolucijske maske, Sobelov operator, Houghova linearna transformacija, odvod in podobno). Razvijte postopek, ki z uporabo lastnih metod in Houghove linearne transformacije zazna pozicije in omejujoče stranice objektov na topografski karti. Zaznani objekti naj bodo samostojni in ločeni od ostalih. Izdelajte program, s katerim boste prikazali delovanje razvitega postopka. Program, ki naj bo preprost in prijazen do uporabnika, naj rezultat obdelave zapiše v datoteko za prikaz v 3D urejevalniku.

Mentor:

doc. dr. Tomaž Dobravec

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Andrej Zadnikar,

z vpisno številko 63020316,

sem avtor/-ica diplomskega dela z naslovom:

Zaznavanje objektov na topografskih kartah

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  doc. dr. Tomaža Dobravca

- 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 _____ Podpis avtorja/-ice: _____

Zahvala

Iskreno se zahvaljujem mentorju doc. dr. Dobravcu za strokovne nasvete in pobude.

Svoje delo posvečam mami Marjetki,
ki me je potrpežljivo vzpodbujala pri študiju.

Kazalo

Povzetek

Namen diplomskega dela je razviti postopek, ki bi samodejno določil pozicije zgradb ali objektov na topografski karti ter najdenim objektom določil omejujoče stranice.

V delu so sprva na kratko predstavljene nekatere uveljavljene metode, za obdelavo in iskanje značilnosti na slikah. Predstavljene metode so:

- uporaba konvolucijske maske za obdelavo slike s filtri,
- uporaba Sobelovega operatorja za iskanje robov,
- računanje odvoda za preverjanje gradienta,
- Houghova linearna transformacija za iskanje ravnih črt na sliki.

Sledi opis postopka, ki s pomočjo zgoraj naštetih metod najprej poišče približne točke središč vseh objektov, nato pa vsakemu objektu določi omejujoče stranice.

Rezultat postopka je množica mnogokotnikov, sestavljenih iz daljic, ki s svojo obliko in pozicijo ustrezajo tlorisom zgradb na obdelovani topografski karti. Z nadaljnjo obdelavo rezultatov je mogoče le-te predstaviti v 3D okolju.

Opisan in implementiran postopek je delujoč, a prostora za izboljšave je še veliko.

Ključne besede:

zaznavanje objektov, topografska karta, Houghova linearna transformacija, konvolucijska maska, Sobelov operator, odvod, gradient.

Abstract

The purpose of this graduation thesis is to develop an algorithm to automatically acquire positions of buildings or objects on a topographic map and assign found objects their boundary lines.

The thesis first presents some established methods for image processing and analysis. The presented methods are:

- use of convolution masks for picture filtering
- use of the Sobel operator for edge detection
- derivation for gradient checking
- linear Hough transform

This is followed by the description of an algorithm that uses the aforementioned methods to find approximate positions of all objects and assign each object their boundary lines.

The output of the algorithm is a set of polygons that represent, with their shape and position, buildings on the topographic map. The results can be represented in a 3D environment with some additional processing.

The described and implemented algorithm works well but still has a lot of room for improvement.

Keywords:

object detection, topographic map, linear Hough transform, convolution mask, Sobel operator, derivative, gradient.

1 Uvodni del

Cilj diplomskega dela je sprva prepoznati objekte na topografski karti ter jim določiti pozicijo glede na višino in širino obdelovane karte. Dobljeno pozicijo je nato potrebno uporabiti kot referečno točko za iskanje omejujočih stranic vsakega objekta. Za dosego cilja je potrebno uporabiti metode za obdelovanje slik s konvolucijskimi maskami, operatorje za iskanje robov in Houghovo linearno transformacijo za iskanje ravnih črt. Vsak najdeni objekt mora biti samostojna zaokrožena celota. To bo kasneje omogočalo neodvisno manipulacijo z vsakim objektom posebej. Dobljene podatke je potrebno zapisati v čim bolj podprto obliko zapisa in tako omogočiti prikaz in urejanje v širokem naboru 3D urejevalnikov.

Definicija topografske karte, zapisana na spletnih straneh kanadskega centra za topografijo, pravi: »Topografska karta je podrobna in natančna grafična reprezentacija kulturnih in naravnih značilnosti na tleh.« [3] Topografska karta torej prikazuje vse glavne topografske elemente zemeljskega površja, kot so vode, relief, naselja, zgradbe, prometne poti itd. z dogovorjenimi znaki [1]. Topografske karte so običajno izdelane v merilih 1:25000 ter 1:50000 [2]. Narejene so po strogih geometrijskih zahtevah [2]. Če so dobro vzdrževane in posodobljene, potem so zelo dober pripomoček za natančno lociranje objektov v okolju (slika 1).

Slika 1: Primer topografske karte [9]

Če bi bilo mogoče iz topografske karte urbanega območja samodejno razbrati, kaj je stavba oziroma objekt, določiti, kje se objekt nahaja, ga ločiti od ostalih objektov ter najti njegove omejujoče stranice, potem bi lahko iz dobljenih rezultatov dokaj natančno rekonstruirali pozicije zgradb v poljubnem urbanem območju. Z določitvijo povprečne višine objektov v predelu, ki ga obdelujemo, pa bi podatke lahko predstavili v prostoru.

Osnovna ideja se mi je porodila pri delu v podjetju, ki je izdelovalo vojaške simulacije. Vsakič ko je bilo potrebno računalniško rekonstruirati neko urbano območje, je moral 3D modelar ročno poiskati vse stavbe in vsako posebej ustrezno umestiti v prostor. Velikokrat si je pomagal s topografskimi in ortografskimi posnetki območja, ki ga je bilo potrebno virtualizirati. Pomislil sem, da bi bilo delo lahko vsaj nekoliko olajšano, če bi program samodejno poiskal položaje stavb ter označil tudi stranice vsake stavbe. Tako bi se modelar lahko bolj osredotočil na obliko stavb in ne toliko na mukotrpno prilagajanje in ujemanje vsake stranice stavbe z referenčnim posnetkom. Rekonstrukcija urbanega območja bi bila hitrejša in natančnejša.

Samodejno zaznavanje objektov in njihovih omejujočih stranic bi pospešilo in olajšalo delo tudi na marsikaterem drugem področju:

- simuliranje požarov, povodni, izgredov itd. v urbanih območjih,
- vizualno podprto načrtovanje posegov v urbana območja,
- računanje najboljše možne poti med objekti (path finding),
- načrtovanje novih prometnih poti,
- virtualizacija mest in urbanih območij.

Celoten postopek zaznavanja objektov in detekcije stranic objektov sem si zamislil v šestih korakih, ki bodo podrobno opisani v nadaljevanju:

1. nalaganje poljubne topografske karte,
2. obdelava slike s filtri za izboljšavo ostrine/kontrasta,
3. računanje odvoda na sliki za preverjanje gradienta,
4. detekcija objektov na sliki,
5. iskanje omejujočih stranic posameznega objekta,
6. zapis omejujočih stranic v datoteko.

Slika 2: Virtualizacija urbanih območij, kot to lahko vidimo v Google Earth [11]

Sprva sem želel namesto topografskih kart uporabiti kar ortofotografske posnetke ali fotoplane. Ti so po vsebini enaki topografskim kartam, razlikujejo pa se po načinu prikaza. Ortografski posnetki so namreč združeni, razpačeni aeroposnetki, posneti v istem merilu. Ker so to posnetki resničnega terena iz zraka, se pri detekciji objektov pojavijo težave.

Slika 3: Primer ortofotografskega posnetka [9]

Velik problem se pojavi že pri iskanju samih objektov. Sprva sem pričakoval, da so strehe objektov na ortofotografskih posnetkih dokaj enobarvne, nerazgibane in posledično lahko razpoznavne. Žal ni tako. Veliko streh je močno razgibanih in zato barvno zelo neenotnih (barvaste strehe mansard, številna slemena, grebeni, zastekljene površine ...). Tako se velikokrat zgodi, da algoritem prepozna eno stavbo kot tri, štiri ali celo več objektov na kupu. Taki podatki so popolnoma neuporabni, saj se stranice najdenih objektov skoraj nikoli ne prilegajo.

Slika 4: Primer zelo razgibanih streh [10]

Poleg razgibanih streh so velik problem tudi sence zgradb. Sence imajo ponavadi zelo ostre robove in so lahko barvno precej bolj enotne kot streha sama. Barva sence pa se spreminja, glede na to, kam senca pada. Predstavljajo zelo velik problem, tako da ima včasih tudi človeško oko velike težave pri razlikovanju sence in strehe.

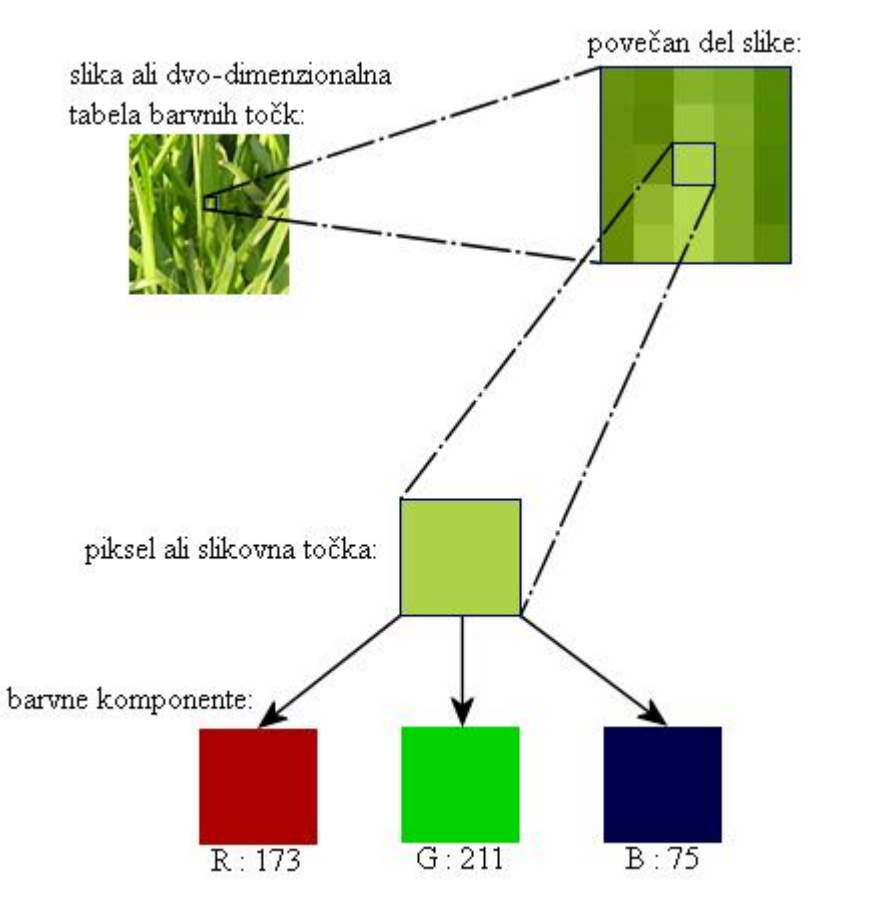
Slika 5: Primer težavnih senc [9]

Dandanes streha ni več le pokrivalo hiše. Na strehah so nameščene panele za pridobivanje elektrike s sončno svetlobo, grelci za vodo, vodni zbiralniki in celo rastlinjaki. Streha postaja vse pomembnejši del hiše in dobiva vse večjo uporabno vrednost. Zaradi tega je vedno težje klasificirati, kaj je streha na ortofotografskem posnetku.

2 Obdelava slike

2.1 Rasterska grafika

V računalniški grafiki je rasterska grafika ali dvodimenzionalna tabela slikovnih točk podatkovna struktura, ki predstavlja pravokotno mrežo barvnih pikslov ali točk na sliki. Tabela točk je določena s širino in višino slike ter številom bitov na slikovno točko. Slednje določa število barv, ki jih slika lahko prikazuje.



Slika 6: Prikaz slike, slikovne točke in komponent barve

Vsaka točka na sliki sestoji iz treh komponent: iz rdeče(R), zelene(G) in modre(B). Te komponente predstavljajo razmerja uporabljenih barv za prikaz barve v eni točki na sliki. Z različnimi razmerji teh treh komponent lahko prikažemo različne barve. Število možnih barv je odvisno od števila uporabljenih bitov za prikaz ene slikovne točke. Običajno se za prikaz slik uporablja 24 bitov na točko. To pomeni, da se 8 bitov uporablja za predstavitev rdeče, 8 bitov za predstavitev zelene in 8 bitov za predstavitev modre barve. 2^8 rezultira v 256 nivojev za vsako komponento in če združimo vse komponente, dobimo 256*256*256, kar je 16777216 različnih barv [12].

Primeri slik z različnim številom bitov na slikovno točko:

- 24 bitov na slikovno točko (24 bpp - bits per pixel) ali truecolor.

Slika 7: 16777216 barv

- 8 bitov na slikovno točko, kar pomeni, da za prikaz rdeče porabimo 3 bite, 3 bite za prikaz zelene in 2 bita za prikaz modre. Človeško oko je manj dovzetno za modro barvo, zato je modri komponenti pripisan en bit manj kot ostalima [12].

Slika 8: 256 barv

- 4 biti na slikovno točko, kjer se uporablja paleta 16 barv. Naključno razprševanje barv pripomore k boljši kvaliteti slike.

Slika 9: 16 barv

- 1 bit na slikovno točko. Edini barvi na sliki predstavljata črna in bela.

Slika 10: 2 barvi

2.2 Uporaba konvolucijske maske

Dvodimenzionalni zapis podatkov omogoča uporabo konvolucijskih mask za obdelavo podatkov na sliki. Konvolucijske maske so ena najbolj pogosto uporabljenih metod za obdelovanje slik. Prilagodimo jih lahko za različne naloge, od mehčanja in ostrenja do iskanja robov na sliki itd.

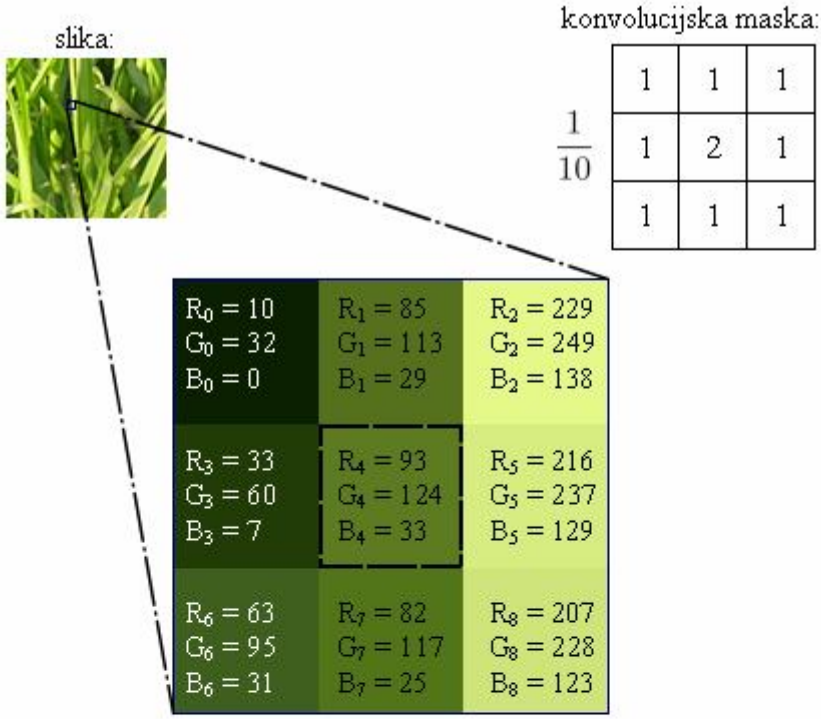
Apliciranje konvolucijske maske je časovno potratno, saj je potrebno za izračun vsake nove slikovne točke prebrati n^2 (n je red konvolucijske maske) okoliških točk. Točke je nato potrebno pomnožiti s pripadajočo vrednostjo v konvolucijski maski ter jih ustrezno sešteti. Izračunane vrednost komponent so barvne komponente nove točke na sliki. Čas izvajanja se znatno poveča, če povečamo velikost konvolucijske maske.

Psevdokoda procesiranja slike s konvolucijsko masko:

```
raz = red konvolucijske maske / 2
red = red konvolucijske maske

za x od raz do (širina slike – raz)
     za y od raz do (višina slike – raz)
          za mx od 0 do (red – 1)
               za my od 0 do (red – 1)

                    nova vrednost = nova vrednost
                    + slika[x + (mx - raz), y + (my - raz)]
                    * konvolucijska maska[mx, my]

          deli novo vrednost slikovne točke z deliteljem
          zapiši novo vrednost v rezultat[x,y]
```

Časovna zahtevnost je O((širina slike - n) * (višina slike - n) * n^2), pri čemer je n red konvolucijske maske.

Uporaba konvolucijske maske za izračun nove vrednosti v slikovni točki poteka, kot je prikazano na spodnji sliki. Pri tej metodi gre za polaganje konvolucijske maske na mrežo izbranih slikovnih točk. Izračunana vrednost pripada novi sredinski slikovni točki (slika 11).



$$R = \frac{1}{10}(R_0 + R_1 + R_2 + R_3 + 2R_4 + R_5 + R_6 + R_7 + R_8)$$

$$G = \frac{1}{10}(G_0 + G_1 + G_2 + G_3 + 2G_4 + G_5 + G_6 + G_7 + G_8)$$

$$B = \frac{1}{10}(B_0 + B_1 + B_2 + B_3 + 2B_4 + B_5 + B_6 + B_7 + B_8)$$

Nova vrednost središčne(očrtane) slikovne točke:

$$R = 111,1 \approx 111$$
$$G = 137,9 \approx 138$$
$$B = 54,8 \approx 55$$

Slika 11: Primer apliciranja konvolucijske maske

Primeri nekaterih konvolucijskih mask:

$$\frac{1}{2}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 10 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Slika 12: Uporaba konvolucijske maske za ostrenje slike

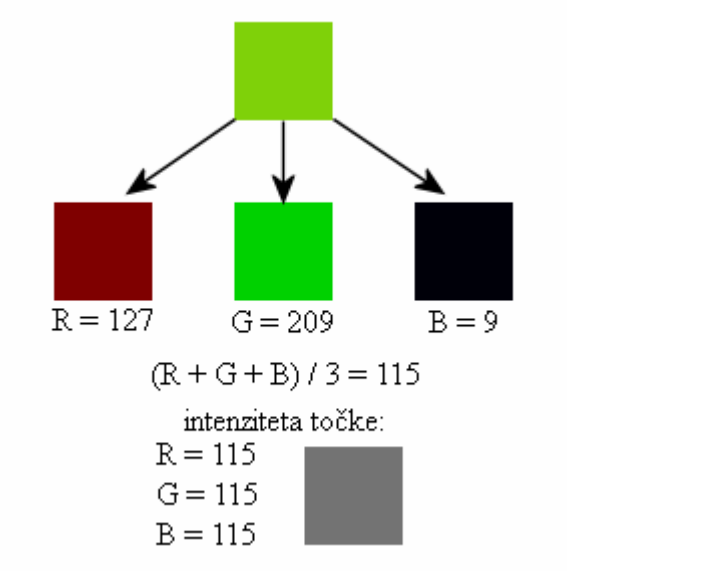
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Slika 13: Uporaba konvolucijske maske za glajenje slike (Gaussov filter)

2.3 Pretvarjanje barvne slike v sliko nivojev sivin

Vsaka slikovna točka na sliki sivin (grayscale) je predstavljena kot ena sama vrednost, in nosi le informacijo o intenziteti. Intenziteta ali odtenek sive barve je na intervalu od črne (najnižja intenziteta, vrednost 0) do bele barve (najvišja intenziteta, vrednost 255). Vseh odtenkov sivine skupaj je 256.

Večina metod za iskanje lastnosti, kot so recimo robovi na slikah, zahteva za svoje uspešno izvajanje predhodno pretvarjanje barvne slike, v sliko nivojev sivin. Najpreprostejši postopek za pretvarjanje barvne slike v sliko sivin je povprečenje vseh treh barvnih komponent v slikovni točki. Izračunano vrednost povprečja zapišemo na mesto vseh treh barvnih komponent nove točke.

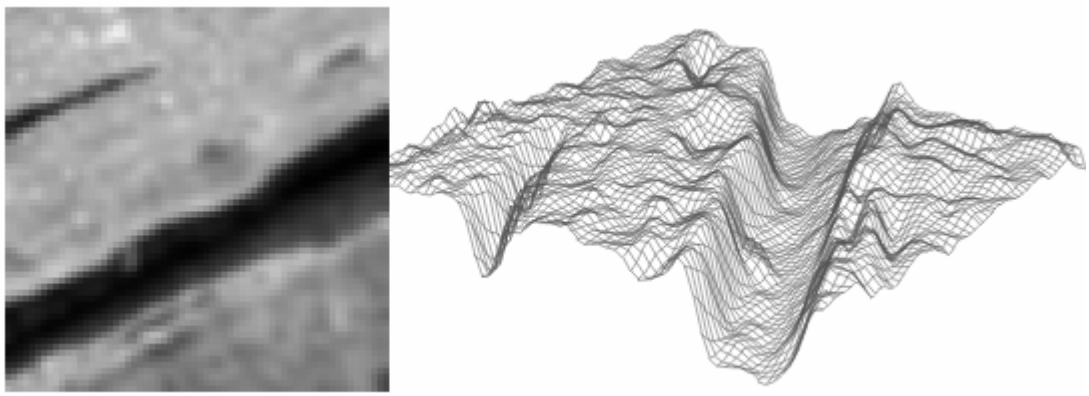


Slika 14: Računanje intenzitete v slikovni točki s povprečenjem

Slika 15: Pretvarjanje barvne slike v sliko nivojev sivin

2.4 Računanje odvoda za preverjanje gradienta na sliki

Če sliko pretvorimo v nivoje sivin, potem si jo lahko predstavljamo kot tro-dimenzionalni relief, kjer intenziteta posameznih slikovnih točk predstavlja razdaljo v navpični smeri – višino (slika 16). Z računanjem razlike intenzitete med sosednjima slikovnima točkama (odvod) lahko ugotovimo, kakšen je gradient ali naklon na reliefu. Če je absolutna vrednost razlike večja od vnaprej določenega praga, potem smatramo, da smo naleteli na rob, in obratno. Rezultat tovrstnega računanja je 1 (je rob) ali 0 (ni rob). Take rezultate lahko zapišemo v dvodimenzionalno tabelo tipa boolean ali v sliko z enim bitom na slikovno točko. Najdeni robovi so tako zapisani eksplicitno. Taka oblika je enostavna za uporabo v nadaljnjem računanju.

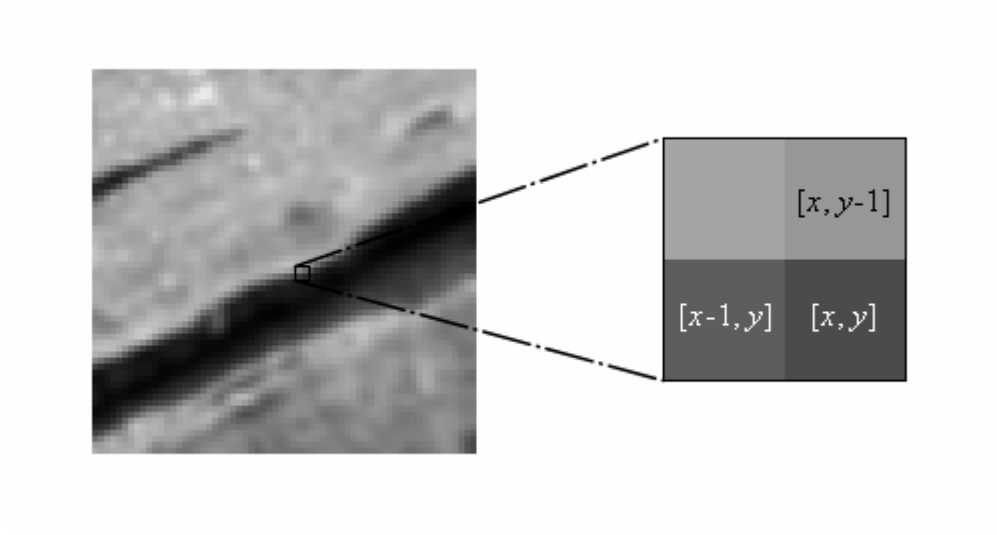


Slika 16: Intenziteta posameznih točk na sliki kot višina na reliefu

Odvod se izračuna iz intenzitete treh slikovnih točk. Prvič se izračuna razlika med intenziteto sosednjih točk v smeri *x*, nato pa še v smeri *y* (slika 17). Če je absolutna vrednost razlike sosednjih točk večja od prej določenega praga *p*,

$$(|[\, x, y\,] - [\, x,\ y\text{-}1\,]| > p) \vee (|[\, x, y\,] - [\, x\text{-}1,\ y\,]| > p)$$

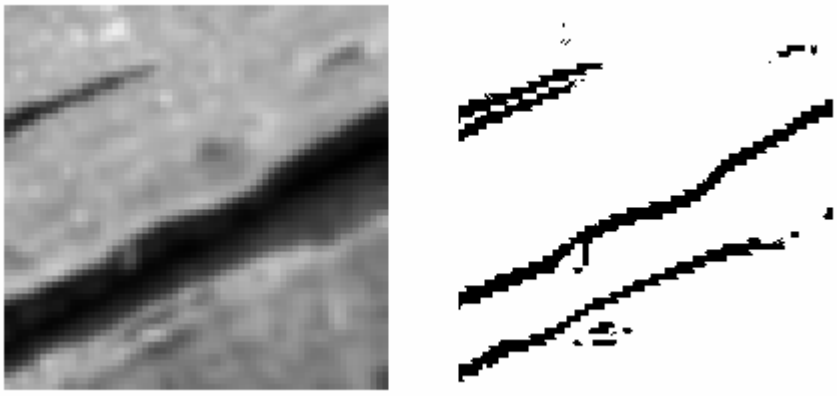
potem je bil najden rob. V rezultatno tabelo ali sliko zapišemo najdeni rob v točki [*x, y*] (slika 18).



Slika 17: Računanje odvoda iz intenzitete treh slikovnih točk

Psevdokoda procesiranja odvoda slike:

```
za y od 1 do višina slike - 1
     za x od 1 do širina slike - 1
          rezultat[x,y] = ni rob
          če (abs(slika[x,y] - slika[x-1,y]) > meja) ali
             (abs(slika[x,y] - slika[x,y-1]) > meja)
          potem
               rezultat[x,y] = je rob
```



Slika 18: Rezultat primerjanja gradienta na sliki

2.5 Sobelov operator

Sobelov operator se uporablja predvsem v algoritmih za detekcijo robov na predhodno osivljenih slikah. Operator izračunava gradient intenzitete na vsaki točki slike ter podaja smer in vrednost največje možne spremembe v intenziteti. Zaradi majhnega števila zajetih slikovnih točk operator zaznava le robove v malem merilu ter je razmeroma občutljiv na šum v sliki.

Je diskretni operator odvajanja, deluje pa na principu dveh konvolucijskih mask reda 3 (slika 19).

G_y

+1	+2	+1
0	0	0
-1	-2	-1

G_x

+1	0	-1
+2	0	-2
+1	0	-1

Slika 19: Konvolucijski maski Sobelovega operatorja

Vrednost vsake točke na sliki se izračuna dvakrat: prvič z uporabo maske G_x v smeri x in drugič z uporabo maske G_y v smeri y. Rezultat računanja z uporabo maske G_x naj bo r_x, rezultat računanja z uporabo maske G_y pa r_y. Dobljena rezultata je potrebno združiti:

$$r = \sqrt{r_x^2 + r_y^2}$$

Izračunana vrednost r predstavlja novo izračunano vrednost intenzitete [8].

Rezultat apliciranja Sobelovega operatorja so svetli najdeni robovi na temni podlagi. Intenziteta najdenega roba pove, kako nenadna ali gladka je sprememba na sliki in s tem kolikšna je verjetnost, da ta del slike predstavlja rob (slika 20).

Slika 20: Uporaba Sobelovega operatorja na sliki

Psevdokoda procesiranja slike s Sobelovim operatorjem:

```
raz = red konvolucijske maske Gx / 2
red = red konvolucijske maske Gx

za x od raz do širina slike - raz
     za y od raz do višina slike - raz
          za mx od 0 do red - 1
               za my od 0 do red - 1

                    rx = rx
                    + slika[x + (mx - raz), y + (my - raz)]
                    * Gx[mx, my]

                    ry = ry
                    + slika[x + (mx - raz), y + (my - raz)]
                    * Gy[mx, my]

          združi rx in ry
          preveri če je r na intervalu [0,255]
          zapiši r v rezultat[x,y]
```

Časovna zahtevnost je O((širina slike - n) * (višina slike - n) * 2 * n^2), pri čemer je n red konvolucijske maske Sobelovega operatorja.

2.6 Houghova linearna transformacija

Houghova linearna transformacija je tehnika, ki se uporablja pri analizi in procesiranju slik ter računalniškem vidu. S to tehniko je mogoče odkriti nepopolno pojavljanje ravnih črt z metodo glasovanja. Glasovanje se akumulira v koordinatnem sistemu, v katerem os x predstavlja kot θ premice, os y pa razdaljo r premice od izhodišča. Iz koordinatnega sistema ali Houghovega akumulacijskega prostora se na koncu glasovanja prebere parametre premic z največjim številom glasov.

Naj ima točka v ravnini koordinate (x_0, y_0). Premici ki potuje skozi to točko, in ima podan parameter θ, lahko izračunamo parameter r z enačbo:

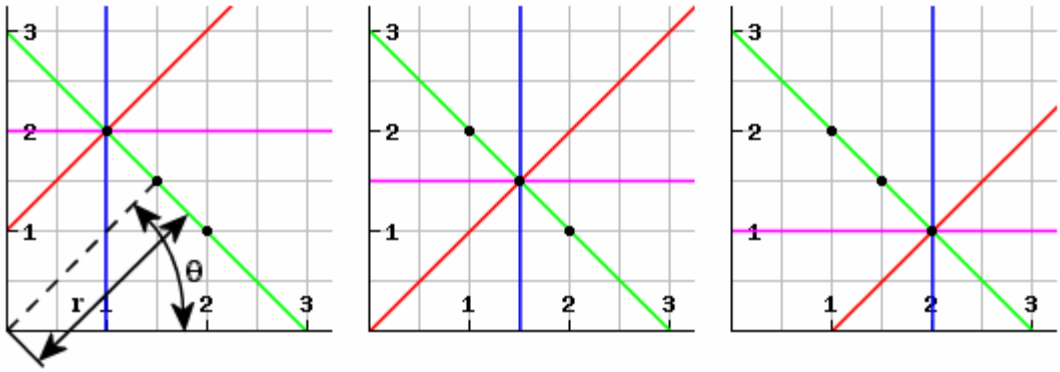
$$r(\theta) = x_0 * cos\theta + y_0 * sin\theta$$

Enačba ustreza sinusni krivulji v ravnini (r,θ). Ta krivulja pripada točki (x_0, y_0) in parametroma (r,θ) vseh možnih premic skozi to točko. Točke, ki v karteziènem koordinatnem sistemu tvorijo ravno črto, bodo v ravnini (r,θ) tvorile sinusne krivulje; te se bodo sekale v točki, ki predstavlja parametre premice – nosilke ravne črte v karteziènem koordinatnem sistemu [6,7].

Premico z dobljenima parametroma θ in r lahko zapišemo z enačbo:

$$y = (-\frac{cos\theta}{sin\theta})x + (\frac{r}{sin\theta})$$

Postopek izvajanja Houghove linearne transformacije je sledeč:



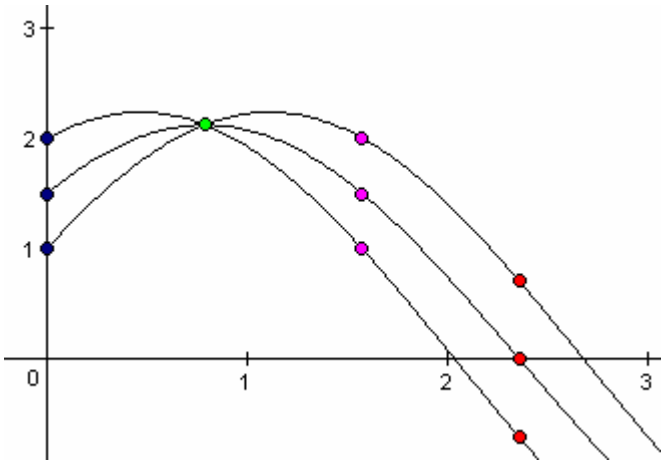
Kot(θ)	Razdalja(r)
0	1
45	2,12
90	2
135	0,71

Kot(θ)	Razdalja(r)
0	1,5
45	2,12
90	1,5
135	0

Kot(θ)	Razdalja(r)
0	2
45	2,12
90	1
135	-0.71

Slika 21: Primer računanja parametrov premice na točki

- V vsaki točki se določi parameter θ ter izračuna parameter r za končno množico premic, ki potujejo skozi to točko. Kot θ se za vsako premico poveča, običajno teče od 0 do 180 s primerno majhnim korakom.
- V Houghov akumulacjski prostor se vpiše oba parametra vsake izračunane premice. Vodoravna os predstavlja parameter θ, navpična os pa parameter r.
- Po končanem glasovanju s Houghovega akumulacijskega prostora preberemo parametre tistih točk, ki so dobile največ glasov. Ti parametri so parametri najdenih premic – nosilk ravnih črt.



Slika 22: Rezultat glasovanja v Houghovem akumulacijskem prostoru

V tem primeru se je največ glasov akumuliralo v zeleni točki (slika 22), ki s svojima koordinatama ustreza premici:

$$y = \frac{2.12 - x * cos(45)}{sin(45)}$$

Psevdokoda glasovanja za premice:

```
za i od 0 do stevila točk - 1
     za kot od 0 do 179
          razdalja = točka[i].x * cos(kot)
                   + točka[i].y * sin(kot)
          vpiši kot in razdaljo v Houghov akumulacijski
          prostor
```

3 Shranjevanje podatkov o 3D objektih (oblika .obj)

Oblika .obj je odprta oblika zapisa za shranjevanje podatkov, ki predstavljajo 3D objekte. Zapis je ponavadi sestavljen iz seznama vseh prostorskih točk (vertex), teksturnih koordinat (texture vertex) in vektorjev normal (normals), uporabljenih v vseh objektih, ki jih datoteka vsebuje. Seznamu naštetih gradnikov sledi seznam vseh lic (faces), ki jih tvorijo indeksirane prej naštete točke. Indeksi potekajo po vrsti. Prva zapisana prostorska točka (vertex) ima indeks 1, prva zapisana teksturna koordinata (texture vertex) ima indeks 1 itd., druga zapisana prostorska točka (vertex) ima indeks 2 itd.

Primer zapisa .obj:

```
mtllib mat.mtl

v  0 0 0
vt 0 0
vn 0.0 0.0 0.0
v  1 0 0
vt 1 0
vn 0.0 0.0 0.0
v  1 1 0
vt 1 1
vn 0.0 0.0 0.0
v  0 1 0
vt 0 1
vn 0.0 0.0 0.0

g grupa1
usemtl mat1
f 1/1/1 2/2/2 3/3/3 4/4/4
```

Zgornji zapis predstavlja štirikotnik (f v0/vt0/vn0 v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3) v prostoru. Ta je sestavljen iz štirih točk (v *x y z*) in leži na ravnini *z*. Vsaka točka ima tudi teksturno koordinato (vt *x y*) in vektor normale (vn *x y z*). Štirikotnik pripada grupi grupa1 (g grupa1) in za teksturo uporablja material mat1 (usemtl mat1). Material je definiran v datoteki mat.mtl (mtllib mat.mtl) [4].

Vsebina datoteke, v kateri je definiran material mat 1, pa je sledeča:

```
newmtl mtl1
map_Kd material.bmp
```

Material z imenom mtl1 (newmtl mtl1) je sestavljen iz teksture material.bmp (map_Kd material.bmp). Deklariranje materiala omogoča še veliko drugih nastavitev, kot so barva, odsev, osvetljenost, ... [5]

4 Iskanje objektov na sliki

4.1 Predstavitev postopka

Kratek pregled postopka, kako poiskati objekte na sliki in zaznati njihove omejujoče stranice

1. Nalaganje slike in priprava pomnilnika
   Nalaganje slike zapisa .bmp v glavni pomnilnik.

2. Priprava slike
   Naloženo sliko obdelamo s konvolucijskimi maskami.
   - VHOD: originalna slika,
   - IZHOD: prečiščena slika.

3. Odvajanje slike
   Izračunamo odvod na sliki za preverjanje gradienta.
   - VHOD: prečiščena slika,
   - IZHOD: bitna slika največjih gradientov.

4. Iskanje objektov na sliki
   - VHOD: prečiščena slika,
   - IZHOD: pozicije objektov na sliki.

   4.1. Izbira barve in nastavitev tolerance barve za omejitev števila objektov
   Uporabniku omogočimo, da z izbiro barv izbere kateri so objekti, ki ga zanimajo.
   - VHOD: prečiščena slika,
   - IZHOD: nabor barv željenih objektov.

   4.2. Apliciranje Sobelovega operatorja
   Iskanje robov na sliki.
   - VHOD: prečiščena slika,
   - IZHOD: najdeni robovi, predstavljeni po intenziteti.

   4.3. Odvod
   Poudarjanje in digitalizacija robov na sliki.
   - VHOD: najdeni robovi, predstavljeni po intenziteti,
   - IZHOD: robovi v bitni sliki.

4.4. Upoštevanje izbora barv ali apliciranje izvzete maske

Upošteva se barve, izbrane v koraku 4.1.

- VHOD: robovi v bitni sliki,
- IZHOD: iskani objekti so bele gruče na sliki.

4.5. Močno mehčanje slike

S pomočjo večje konvolucijske maske omehčamo sliko, da postane intenziteta slikovne točke na sredini objekta največja.

- VHOD: bele gruče, ki predstavljajo objekte,
- IZHOD: močno zglajena slika, na kateri točke z največjo intenzizeto predstavljajo iskane objekte.

4.6. Detekcija vrhov na sliki

Vrhovi ali najvišje intenzitete na sliki predstavljajo iskane objekte.

- VHOD: močno zglajena slika, na kateri točke z največjo intenziteto predstavljajo iskane objekte,
- IZHOD: pozicije iskanih objektov.

5. Iskanje mej objektov

- VHOD: pozicije iskanih objektov,
- IZHOD: omejujoče stranice iskanih objektov.

5.1. Streljanje žarkov in detekcija trka

Iz najdenih točk se razpršijo žarki v vseh smereh, izračuna se točke trkov žarkov z robom na sliki (korak 3).

- VHOD: pozicije iskanih objektov, bitna slika največjih gradientov (korak 3),
- IZHOD: točke trkov žarkov z robovi.

5.2. Detekcija ravnih črt

S pomočjo točk, v katerih so žarki trčili ob robove, se vrši iskanje ravnih črt.

- VHOD: točke trkov žarkov z robovi,
- IZHOD: nosilne premice robov objekta.

5.3. Iskanje najboljših omejujočih stranic

Iz nabora vseh premic je potrebno poiskati le tiste, ki omejujejo iskani objekt.

- VHOD: nosilne premice robov objekta, točke trkov žarkov z robovi,
- IZHOD: omejujoče stranice objekta v obliki daljic.

6. Zapis rezultatov

Zapis rezultatov v obliko, ki omogoča prostorski ogled.

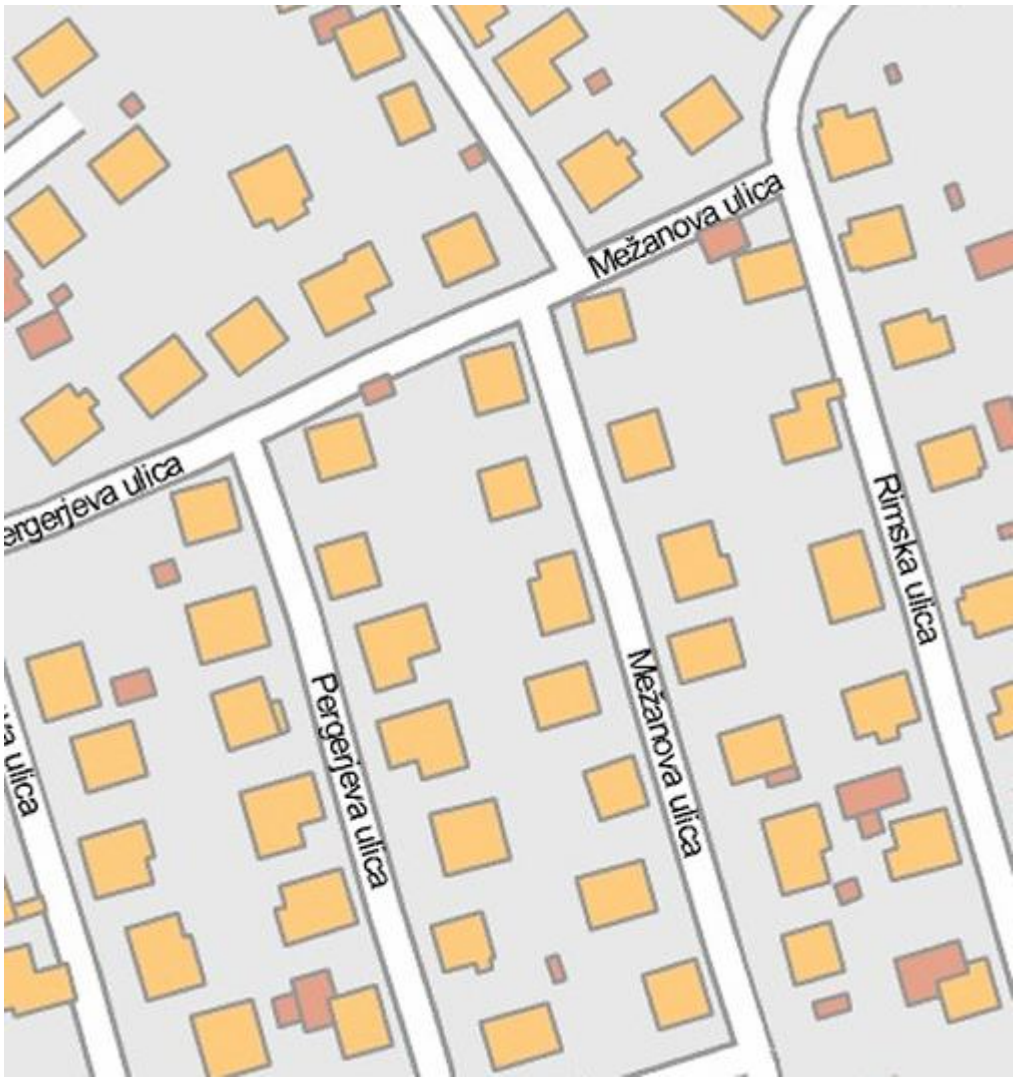
- VHOD: omejujoče stranice objekta v obliki daljic,
- IZHOD: datoteka zapisa .obj, v kateri se nahajajo vsi najdeni objekti. Objekti imajo umetno podano višino, da jih lahko prikazujemo v prostoru.

4.2 Podroben opis postopka

4.2.1 Nalaganje slike in priprava pomnilnika

Sprva se z lokalnega diska ali kateregakoli drugega podatkovnega medija naloži slika v uporabniški vmesnik (slika 23). Sprosti in pripravi se prostor v pomnilniku, ki je namenjen shranjevanju vmesnih rezultatov obdelave slike. Ko je namreč zahtevana manipulacija z vsemi slikovnimi točkami na sliki, je sliko potrebno pretvoriti iz objekta TImage v dvodimenzionalno tabelo celih števil, na katero kaže kazalec. Tovrstno pretvarjanje slike ni ravno elegantna rešitev, a omogoča veliko hitrejše dostopanje do vseh točk na sliki kot dostopanje do posameznih točk preko objekta TImage. Za tak način sem se odločil tudi zato, ker je dobro, da uporabnik s pogledom sledi vmesnim rezultatom, hkrati pa se mora manipulacija s posameznimi slikovnimi točkami izvajati hitro.

Oblika zapisa topografske karte je zaradi preprostosti branja omejena na obliko .bmp - bitmap image.



Slika 23: Topografska karta

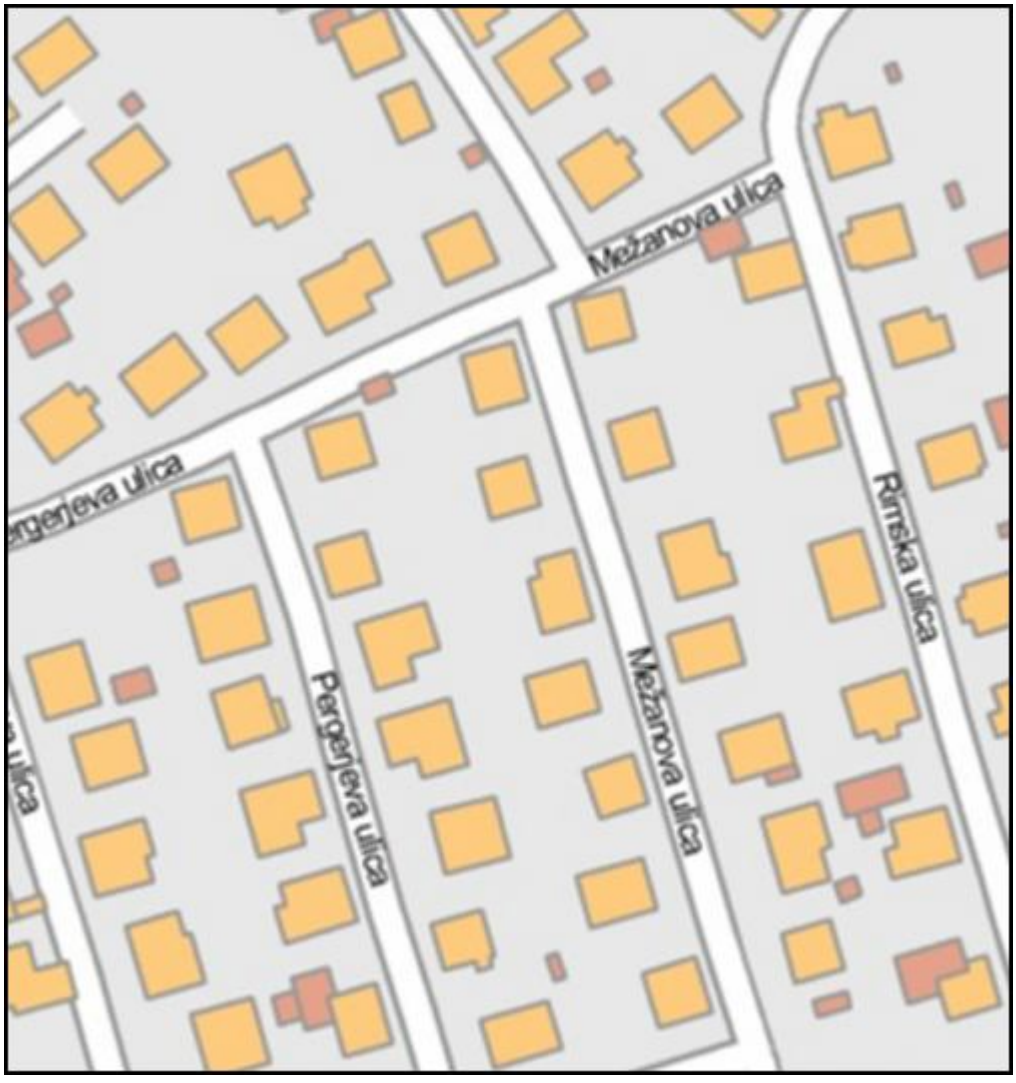
4.2.2 Priprava slike

Z uporabo poljubne konvolucijske maske je sliko mogoče predhodno obdelati in s tem izboljšati rezultate v nadaljnjih korakih obdelave. Dobro je, če so robovi na sliki čim ostrejši, ostale površine pa čim bolj enobarvne in brez šuma. S preizkušanjem sem ugotovil, da je sliko dobro nekoliko omehčati s konvolucijsko masko reda 3 (slika 24).

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Slika 24: Konvolucijska maska za povprečenje

Tako se znebimo šuma, ki je lahko prisoten na sliki. Šum se lahko pojavi kot posledica kompresiranja slike. Četudi je obdelovana slika formata .bmp, kar pomeni, da nima kompresije, je bila verjetno predhodno zapisana v formatu .jpg.



Slika 25: Zglajena slika z uporabo konvolucijske maske

4.2.3 Odvajanje slike

V drugem delu priprave slike se na že očiščeni sliki izvede odvod za preverjanje gradienta. Četudi se bo rezultat odvajanja uporabljal šele v koraku 4.2.5 - iskanje mej objektov, je vse potrebno za njegov izračun že na voljo. Z odvodom slike lahko ugotovimo, kje je razlika intenzitet sosednjih slikovnih točk večja kot vnaprej določen prag, s tem pa dobimo informacijo, kje se nahajajo robovi na sliki. Namesto 0 (ni rob) bomo v rezultatno sliko na mesto, kjer smo računali gradient, zapisali vrednost bele barve (#FFFFFF), namesto 1 (je rob) pa bomo v rezultatno sliko zapisali vrednost črne barve (#000000) (slika 26).

Slika 26: Rezultat preverjanja gradienta na sliki

Dobljena slika je enostavna za preverjanje trka žarka z robom. Robovi so namreč podani eksplicitno. Ustavitev žarkov zaradi debeline robov skoraj vedno uspe in le redko kateri »pobegne«.

4.2.4 Iskanje objektov na sliki

Da bi lahko uspešno poiskali stranice objektov, je sprva potrebno približno določiti, kje bi se lahko objekti nahajali in kaj objekt sploh je.

Do sledečega postopka sem se dokopal s preizkušanjem različnih načinov, kako zaznati ravno površino na sliki.
Postopek temelji na sledečih dveh predpostavkah:

- da je objekt ločen od okolice z zaznavnim robom,
- da je objekt, ki ga iščemo, ali ustrezne barve ali pa je gradient njegove površine zelo blizu nič ali nič (je enobarven).

Če sta ti dve predpostavki resnični, potem lahko s sledečo metodo dovolj dobro določimo pozicije objektov, da potem to informacijo uporabimo pri iskanju stranic posameznih objektov. Metoda uporablja Sobelov operator, odvajanje slike, mehčanje slike in detekcijo vrhov na sliki, več o vsakem postopku pa v nadaljevanju.

4.2.4.1 Izbira barve in nastavitev tolerance barve za iskanje objektov

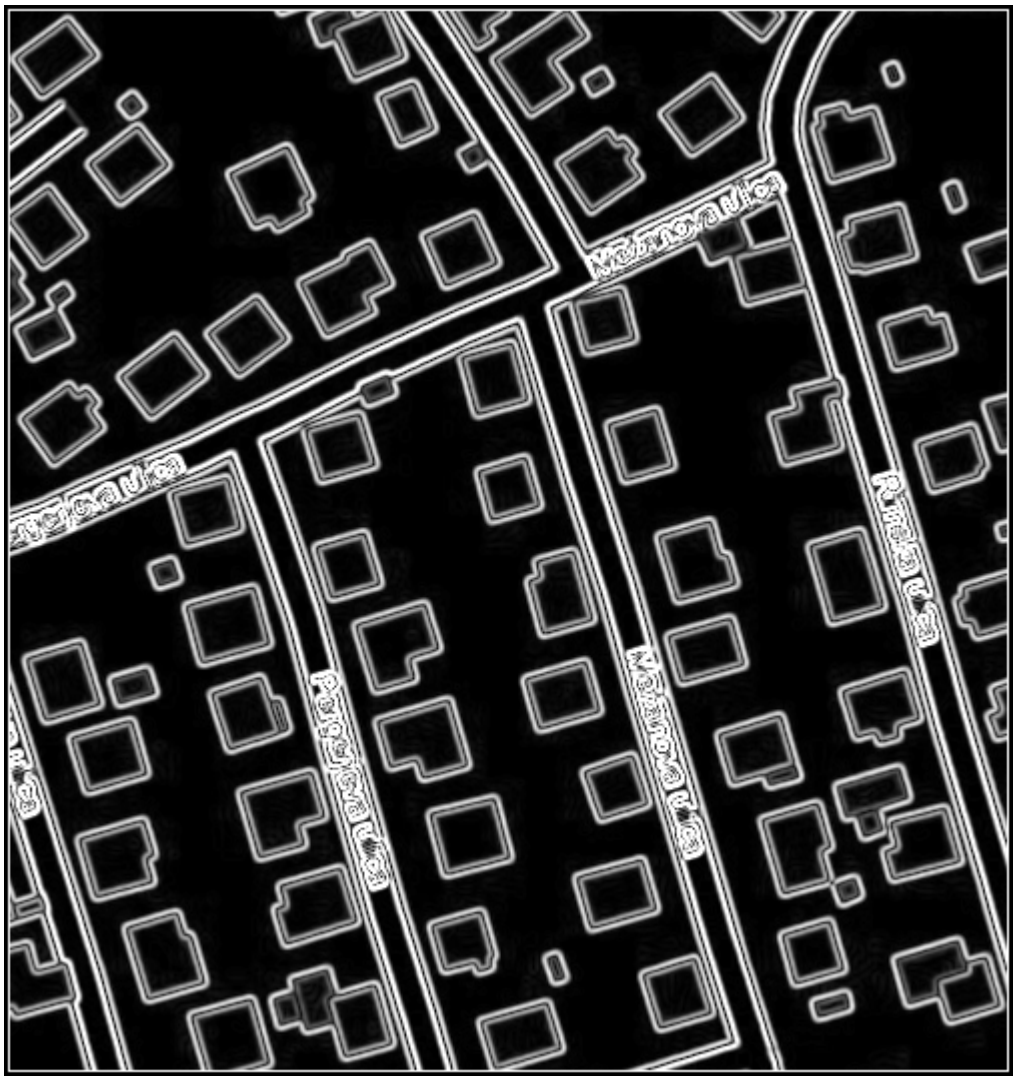
Včasih je na sliki polno enobarvnih površin. Tako je tudi na topografskin kartah. Potrebno je bilo narediti orodje, s katerim lahko uporabnik sam izbere, kakšne barve so tiste površine, ki jih predstavljajo iskani objekti. Izbor se lahko obrne in tako uporabnik lahko določi barve objektov, katerih naj postopek ne obdela.

Slika 27: Izbiranje barv objektov

Na zgornji sliki uporabnika denimo zanimajo vsi objekti oranžne barve. Z določeno toleranco bodo vse ostale barve odstranjene iz nadaljnjega postopka v koraku 4.2.4.4.

4.2.4.2 Apliciranje Sobelovega operatorja

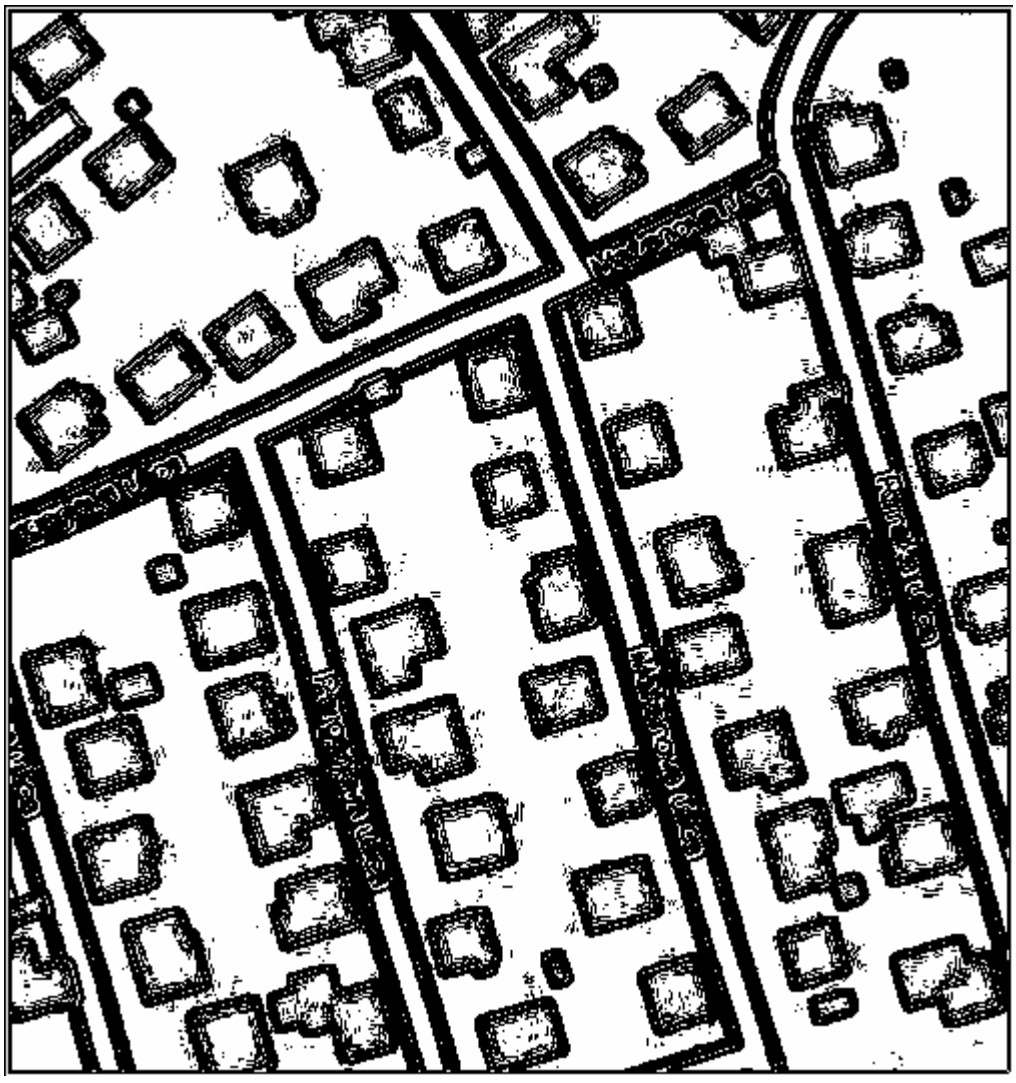
Čeprav je Sobelov operator zelo občutljiv na šum in dobro zaznava le robove z razponom manj kot 3 slikovne točke, sem ga izbral zaradi hitrosti in preprostosti implementacije. Občutljivost na šum v tem primeru ni pomembna, saj v nadaljevanju sliko tako ali tako močno zgladimo. Rezultat apliciranja Sobelovega operatorja so na temni podlagi izstopajoči svetli robovi; njihova intenziteta pove, kako grob oziroma gladek je najdeni rob (slika 28).



Slika 28: Rezultat Sobelovega iskanja robov

4.2.4.3 Odvod

Sledi odvajanje rezultata Sobelovega operatorja. Za vnovično odvajanje že odvedene slike s Sobelovim operatorjem sem se odločil zato, ker je bilo potrebno še bolj poudariti enobarvne površine. Ker so robovi, ki jih producira Sobelov operator, gladki (njihov gradient ni skokovit in ga je lahko zaznati), lahko s ponovnim navadnim odvajanjem ta gradient zopet zaznamo kot rob. Hkrati pa se z navadnim odvajanjem, kot je opisano v koraku 4.2.3, ravne površine obarvajo belo in tako po sami vrednosti postanejo vrhovi na sliki. Te vrhove je potem lahko zaznati. Sicer je res, da je zaznati vrhove enako težko/lahko kot zaznati najnižje vrednosti na sliki, ker pa sem to metodo uporabil tudi pri iskanju vrhov na sliki Houghovega akumulacijskega prostora, sem jo uporabil tudi tu.

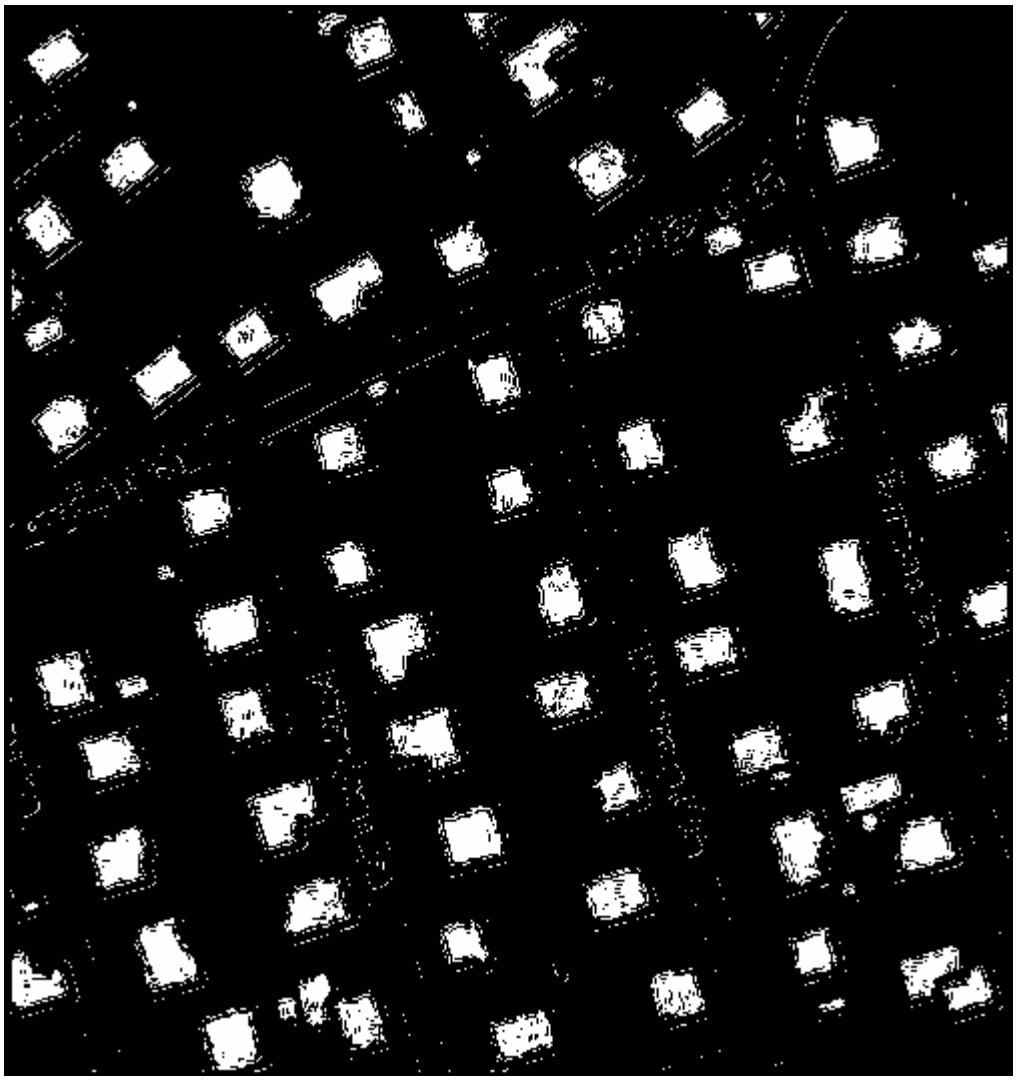


Slika 29: Rezultat zaznavanja gradienta z uporabo odvoda na rezultatu Sobelovega operatorja.

Na sliki je dobro viden šum, ki ga je zaznal Sobelov operator. Šum je še nadalje okrepilo vnovično odvajanje.

4.2.4.4 Apliciranje izvzete maske

Če je uporabnik v koraku 4.2.4.1 izbiral barve za iskanje objektov, potem se v tem koraku na obdelovano sliko aplicira izvzeta maska. Gre za preprosto "polaganje" maske na sliko.



Slika 30: Apliciranje maske na sliko

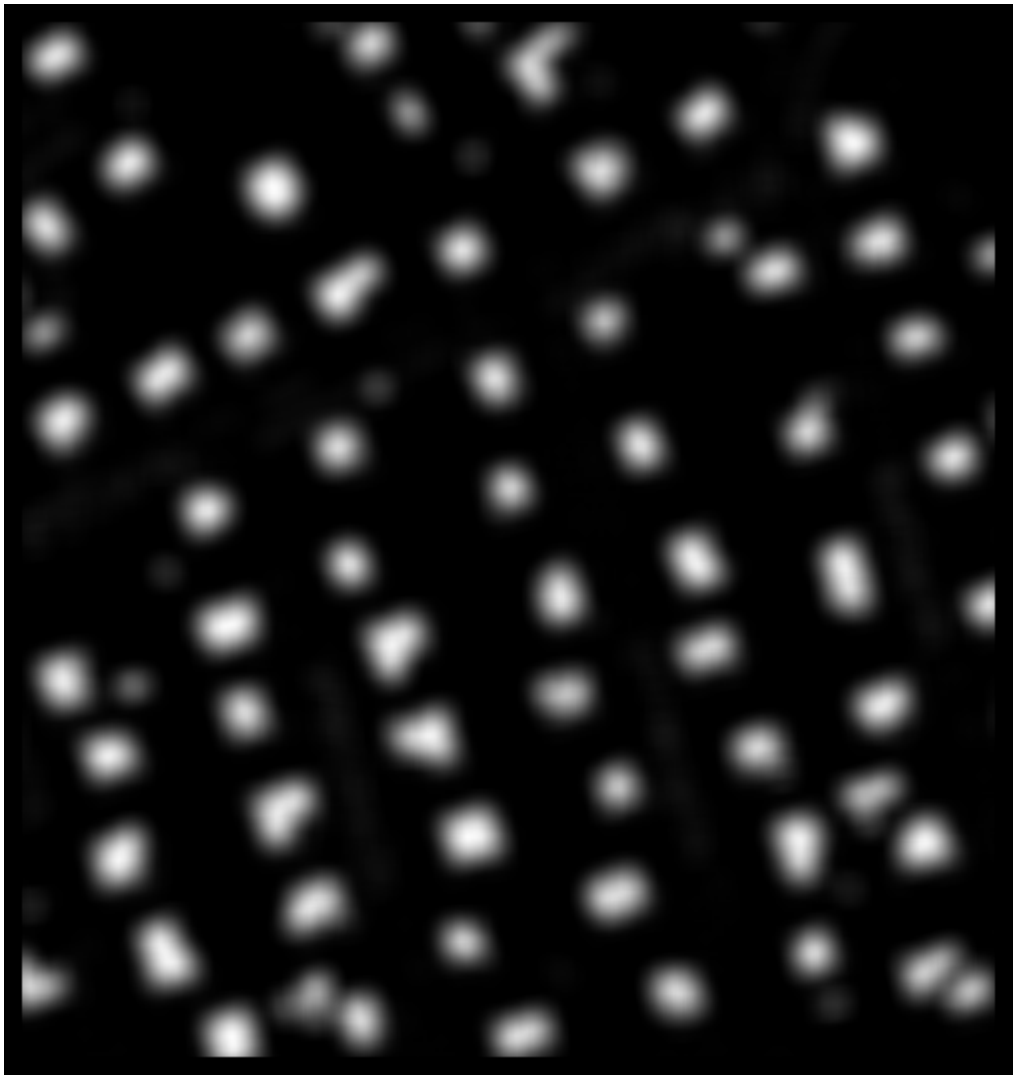
S tem iz nadaljnjega postopka izvzamemo objekte, ki uporabnika ne zanimajo. Ostanejo le bele gruče, ki predstavljajo objekte tiste barve, ki jo je uporabnik izbral v koraku 4.2.4.1.

4.2.4.5 Močno mehčanje slike

Na obdelovano sliko je potrebno aplicirati filter, ki bo sliko omehčal toliko, da bodo kot vrhovi zaznavne le največje enobarvne površine. Zaradi tehnike detekcije vrhov, ki bo opisana v naslednjem razdelku, je dobro, če je konvolucijska maska za mehčanje slike velika vsaj toliko, kot so povprečno veliki največji iskani objekti na sliki.

Predpostavimo, da obdelujemo topografsko karto z merilom, po katerem 1 slikovna točka na karti predstavlja 1 meter v naravi. Predpostavimo tudi, da obdelujemo primestno območje, kjer so največji objekti dvostanovanjske hiše dimenzij 10*10 metrov. Potem je za mehčanje dobro uporabiti konvolucijsko masko velikosti vsaj 11*11. S tako veliko masko namreč zavzamemo celoten objekt naenkrat. To pomeni, da bo najvišja izračunana vrednost maske prav v središčni točki, kjer se nahaja objekt. Ta točka nam bo kasneje služila kot referenčna točka za iskanje mej objekta.

Velikost in uteži uporabljene konvolucijske maske je mogoče nastavljati iz uporabniškega vmesnika. Zgornja velikost maske je omejena na 25*25. Pri tako velikih konvolucijskih maskah je izvajanje časovno zelo zahtevno, saj algoritem prebere kar do 625 okoliških slikovnih točk za izračun vsake nove slikovne točke.



Slika 31: Slika po apliciranju mehčanja

Slika je pripravljena na detekcijo vrhov – to je slikovnih točk z najvišjo intenziteto. Vrhovi približno ustrezajo središčem iskanih objektov.

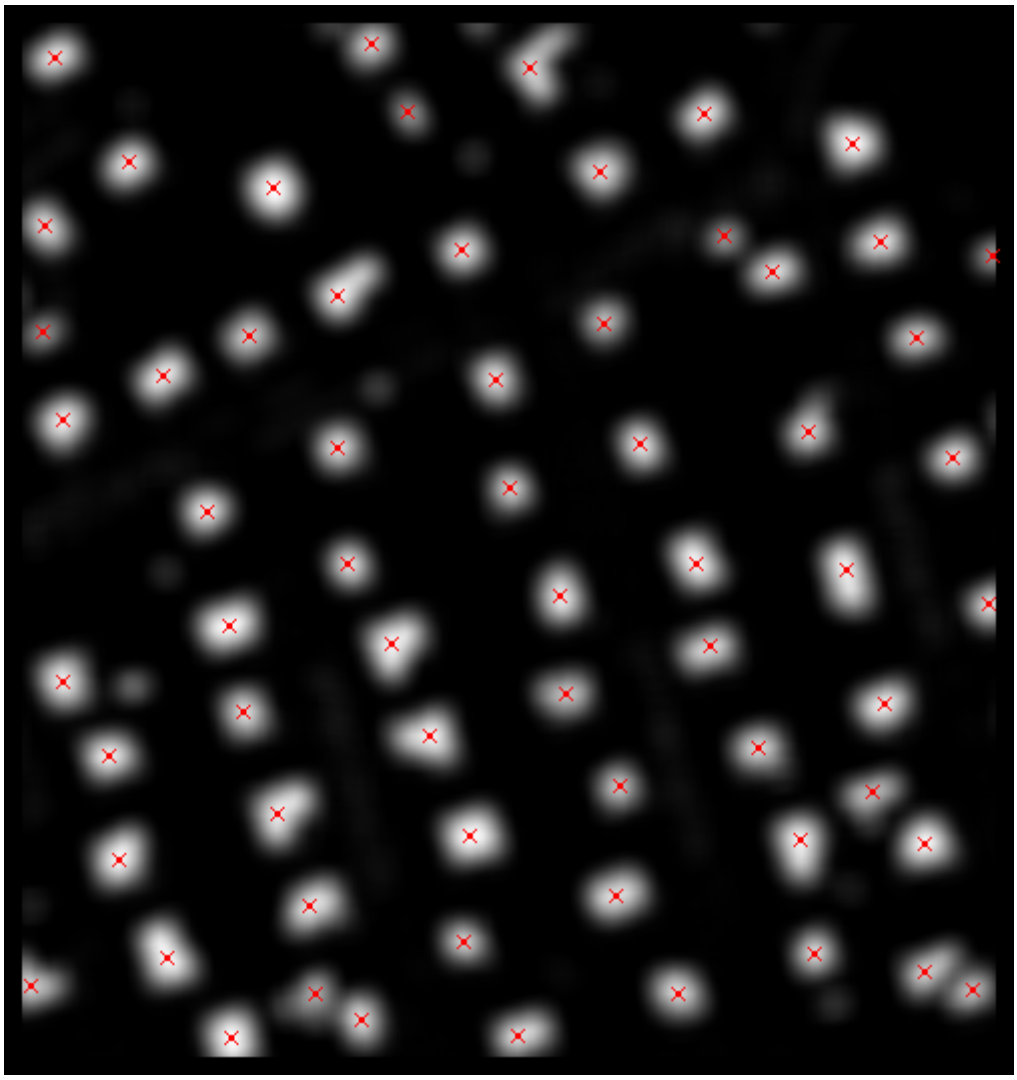
4.2.4.6 Detekcija vrhov na sliki

Idejo za detekcijo vrhov na sliki sem iskal dolgo časa. Hotel sem namreč čim bolj natančno določiti le točko, kjer se nahaja vrh, oziroma največja vrednost slikovne točke. Če je tak vrh na celotni sliki le eden, je iskanje enostavno. Ko pa je vrhov več, so med seboj različno oddaljeni in se razlikujejo po višini (slika 31), se iskanje nekoliko zaplete.

Sprva sem vrhove iskal z odvajanjem, a sem naletel na dve težavi:

- odvod je enak nič tudi na sedlih in grebenih, ki pa ne predstavljajo vrha,
- problem se pojavi tudi pri vrhu, ki je sestavljen iz večih točk enakih intenzitet.

Nato sem se domislil zelo preprostega in učinkovitega načina za odkrivanje vrhov. Vsak vrh namreč predstavlja točko, ki je po svoji intenziteti višja od točk v okolici. To lastnost se da s pridom izkoristiti. Način je podoben kot pri konvolucijskih maskah, ki za izračun nove točke upoštevajo n*n okoliških točk. Tu ne gre za izračunavanje nove točke, temveč le za preizkus, če je intenziteta določene točke višja od vseh ostalih v okolici n*n. Če se izkaže, da je točka vrh, potem njeno pozicijo zabeležimo, točki sami pa popravimo intenziteto na MaxInt. S tem preprečimo odkritje morebitnega drugega vrha, ki je zelo blizu prvega najdenega.



Slika 32: Najdeni vrhovi na sliki

4.2.5 Iskanje mej objektov

Iskanje mej se vrši na odvodu originalne slike (korak 4.2.3) s pomočjo točk, najdenih v koraku 4.2.4.6. Odvod služi kot "ustavitveni pogoj" za žarke, točke pa povejo, kje je potrebno iskati objekte.

4.2.5.1 Streljanje žarkov in detekcija trčenja žarka z robom

Vsaka najdena točka v koraku 4.2.4.6 pove, kje bi se lahko nahajal objekt. Slika, na kateri so bile najdene točke, je enakih dimenzij kot odvod slike iz tretjega koraka, zato lahko nanj prenesemo najdene točke ter jih uporabimo kot središča iz katerih se razpršijo žarki.

Tako iz vsake točke na sliki iz koraka 4.2.3 izstrelimo n žarkov v vse smeri, začenši v najdeni točki. Žarki so razporejeni enakomerno, med sosednjima je kot 360/n stopinj. Vzdolž vsakega žarka se sproti prebere intenziteta in pozicija slikovne točke, po kateri žarek trenutno potuje. Ti podatki se shranijo v dvodimenzionalno tabelo za nadaljnjo obdelavo.

Psevdokoda streljanja žarkov:

```
vmesni kot = 360 / število žarkov

za i od 0 do število žarkov - 1
     kot = vmesni kot * i
     izračunaj nosilno premico žarka s kotom kot in začetno
     točko, iz katere streljamo žarke
     za j od 0 do konca žarka
          izračunaj x in y, ki pripadata nosilni premici
          shrani pozicijo in intenziteto v točki[x,y]
```

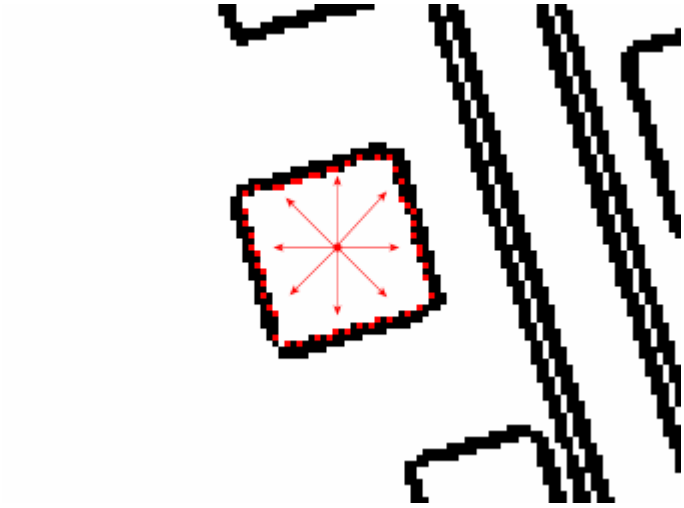
Časovna zahtevnost je O(število žarkov * dolžina žarka).

Rezultat streljanja žarkov nato pošljemo v obdelavo funkciji, ki za vsak žarek izračuna točko trka z robom na odvodu slike ter vrne pozicije teh točk v tabelo rezultatov.

Psevdokoda detekcije trčenja žarka:

```
za i od 0 to število žarkov - 1
     za j od 0 do konca žarka - 1
          če abs(žarki[i, j - 1] - žarki[i, j]) > meja
          potem
               v tabelo rezultatov vpiši koordinate trčenja
```

Tako dobimo množico točk, ki odkrivajo, kje so se žarki zaleteli v rob (slika 33). Če so točke razporejene ob ravnem robu, potem lahko za ta rob izračunamo premico, ki se mu najbolje prilega.



Slika 33: Streljanje žarkov iz najdene točke in njihova ustavitev na robu

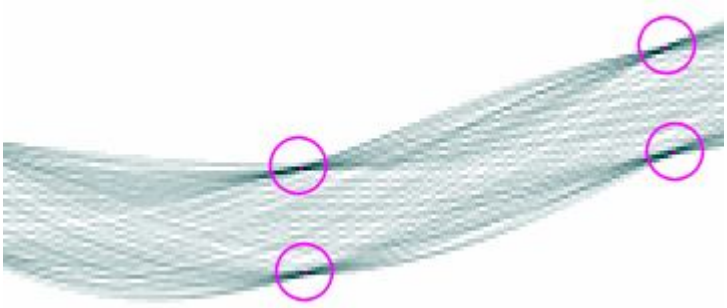
4.2.5.2 Detekcija ravnih črt

Za detekcijo ravnih črt sem uporabil Houghovo linearno transformacijo. Ta deluje na principu glasovanja. Točke, ki so v tem primeru rezultati trkov žarkov, glasujejo za dva parametra premice v ravnini (r,θ).

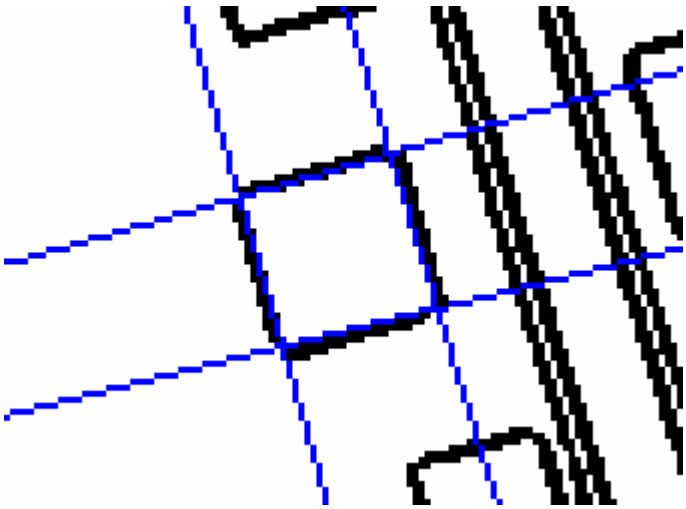
Postopek je sledeč:

- V vsaki točki se določi parameter θ (kot) in izračuna parameter r (oddaljenost od izhodišča) za 180 premic. Kot θ teče za vsako točko od 0° do 179°.
- V Houghovem akumulacijskem prostoru se na točki, ki jo določajo koordinate (r,θ), poveča vrednost.

Po končanem glasovanju je potrebno s Houghovega akumulacijskega prostora prebrati rezultate glasovanja (slika 34). V ta namen lahko uporabimo proceduro, ki zazna vrhove na sliki, ali v tem primeru, točke v katerih se je nabralo največ glasov. Najdeni vrhovi so točke, katerih koordinate (r,θ) so parametri iskanih premic. Premice, ki se najbolje prilegajo robovom objekta, so tako najdene (slika 35). Sedaj je potrebno poiskati še stranice objekta.



Slika 34: Houghov akumulacijski prostor (obkrožene so najvišje vrednosti)



Slika 35: Objekt, obdan z najdenimi premicami – nosilkami stranic

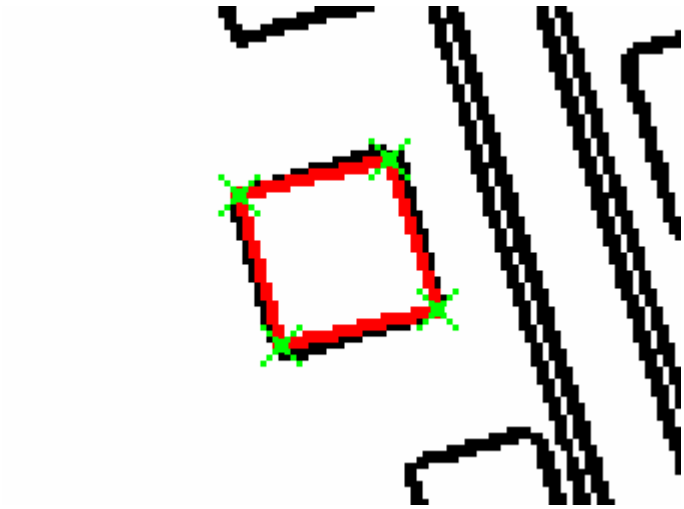
Psevdokoda polnjenja Houghovega akumulacijskega prostora:

```
za i od 0 do stevilo točk - 1
     za kot od 0 do 179
          razdalja = točka[i].x * cos(kot)
                   + točka[i].y * sin(kot)
          vpiši kot in razdaljo v
          Houghov akumulacijski prostor
```

4.2.5.3 Iskanje najboljših omejujočih stranic

Zdi se, da je z odkritjem premic delo že končano, a žal ni tako. Premice so le nosilke stranic. Treba je določiti daljice, ki obdajajo objekt. Če izračunamo vsa presečišča vseh najdenih premic, potem smo že na dobri poti, saj nekaj presečišč ustreza vogalnim točkam objekta. Treba je le ločiti prave točke od tistih, kjer se premici sekata izven iskanega objekta ali znotraj njega.

Vsako premico sem obravnaval posebej in najprej izračunal vsa njena presečišča z drugimi premicami. Nato sem dobljena presečišča uredil po vrsti v tabelo. Pri urejanju presečišč sem upošteval naklon premice. Če je naklon premice od 45 do 135 stopinj, potem so točke urejene po koordinati x, drugače pa po koodinati y. S tem sem se izognil nevšečnostim v primeru, da je premica skoraj navpična ali navpična. Ko so točke urejene, jih po vrsti jemljem v parih. Ti pari točk predstavljajo daljice, za katere je potrebno preveriti, če so usklajene z rezultati iz postopka 4.2.5.1. Tiste, ki najbolje ustrezajo, so omejujoče stranice objekta (slika 36).



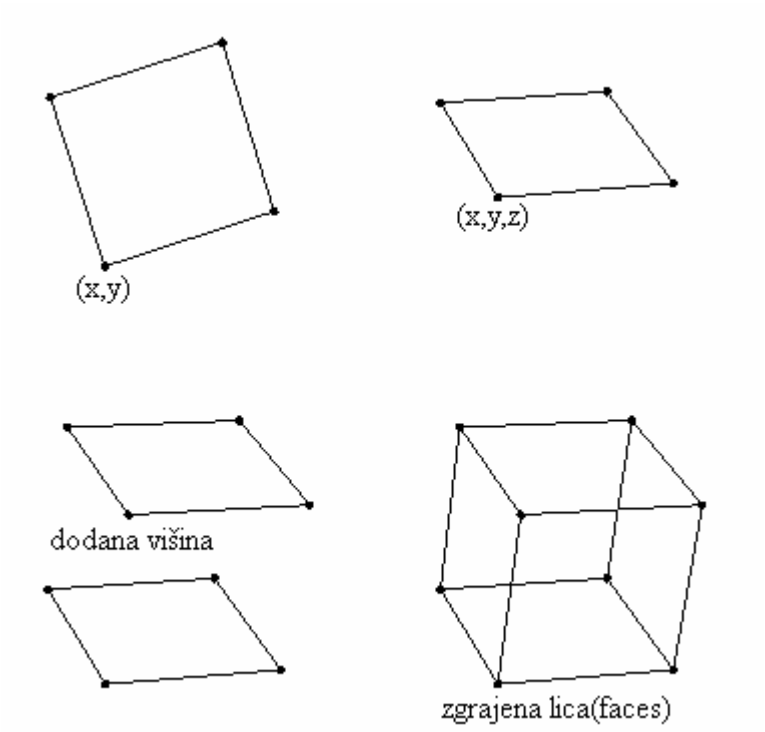
Slika 36: Najdene omejujoče stranice objekta

Psevdokoda iskanja najboljših omejujočih stranic:

```
za j od 0 do število premic - 1
     za j od 0 do število premic - 1
          če j <> i
          potem
               poišči presečišče

     uredi presečišča ene premice po vrsti

     za k od 1 do število presečišč - 1
          prva točka  = presečišča[k - 1]
          druga točka = presečišča[k]

     na daljici od prve točke do druge točke preveri
     sovpadanje s tabelo rezultatov streljanja žarkov

     tista daljica, ki najbolj ustreza se zapiše v
     tabelo najdenih stranic
```

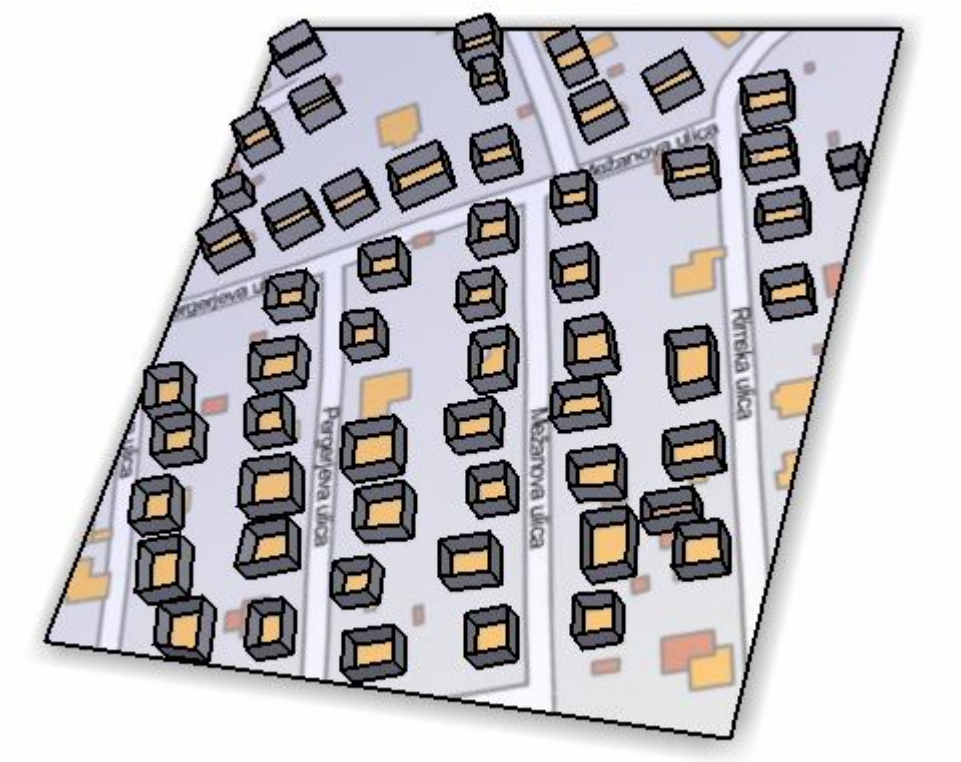
4.2.6 Zapis rezultatov

Vsak najdeni objekt je predstavljen kot mnogokotnik v dveh dimenzijah. Da bi lahko rezultate nazorno predstavili v prostoru, jim je potrebno dodati še tretjo dimenzijo. Le dodajanje tretje dimenzije pa ni dovolj. Potrebno je tudi podvojiti mnogokotnik (tloris) vsakega objekta ter podvojeni mnogokotnik prestaviti v smeri *z* in s tem ponazoriti višino objekta (slika 37). Ker iz topografske karte ni mogoče razbrati višine, jo mora podati uporabnik sam. Načeloma so dvostanovanjske hiše, ki so v Sloveniji zelo pogoste, visoke okrog 10 metrov.



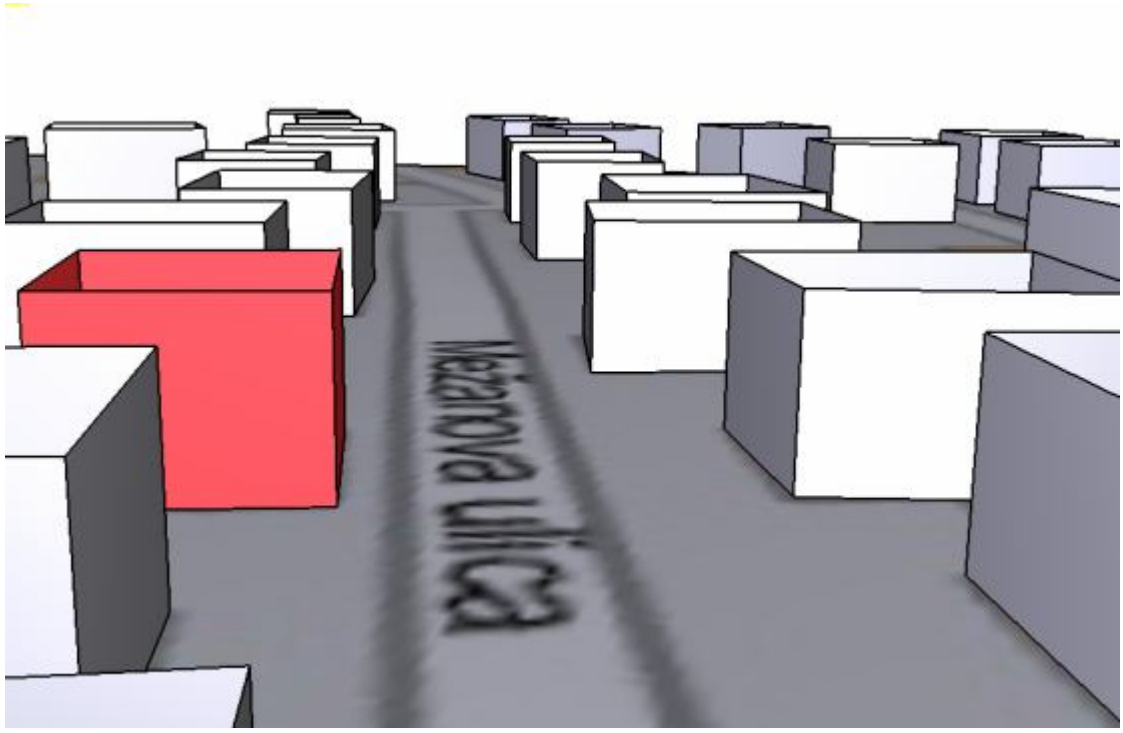
Slika 37: Dodajanje tretje dimenzije, višine in izgradnja lic

Pomemben je enostaven in učinkovit zapis objektov v datoteko. Oblika datoteke mora biti berljiva z večino programov za grafično modeliranje (Maya, 3D Studio Max, Blender ...), tako da lahko 3D modelar nadaljuje delo kar na dobljenih rezultatih. Zaradi zelo dobre podprtosti in dokumentacije sem se odločil za obliko .obj. Oblika datoteke .obj je vsestransko sprejeta oblika zapisa. Omogoča zapis podatkov o objektih v čitljivi obliki ter omogoča uporabo tekstur in materialov.



Slika 38: Rezultat postopka

Ker so vsa lica enega objekta združena v svojo skupino, je vsak objekt samostojen (slika 39). Tako je mogoče vsakega premikati, obračati, skalirati in izbrisati, neodvisno od ostalih. Pod vse najdene objekte sem za boljšo predstavo dodal tudi nespremenjeno topografsko karto.



Slika 39: Izbira enega samega objekta in pogled na celotno ulico

5 Predstavitev programa in nekatere podrobnosti implementacije

Program za odkrivanje omejujočih stranic objektov na topografskih kartah je implementiran v programskem okolju Delphi. Zanj sem se odločil zato, ker imam z njim večletne izkušnje in mi trenutno predstavlja osnovno programsko okolje.

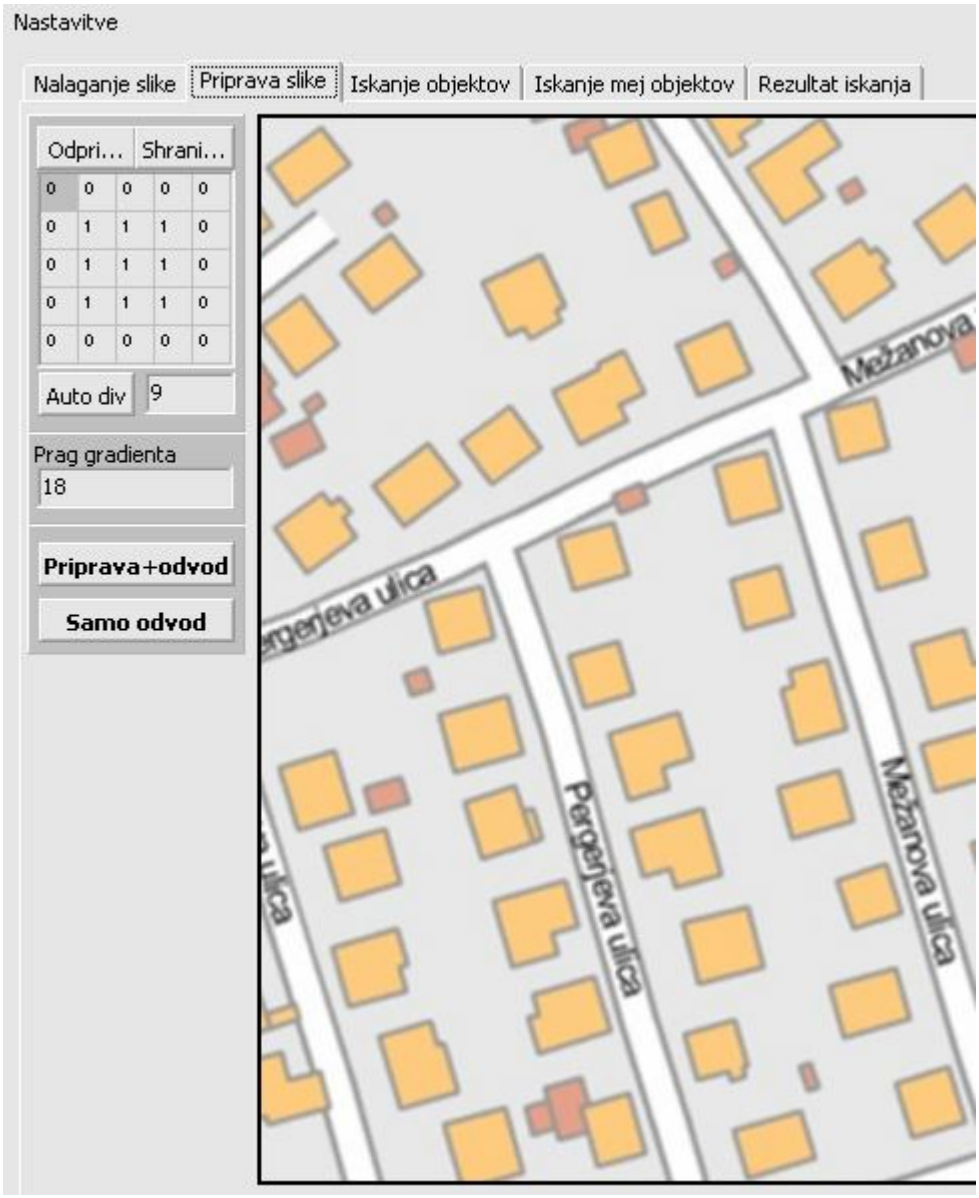
Interakcija uporabnika s programom se odvija preko uporabniškega vmesnika, ki je zasnovan na jezičkih (tabs). Jezički Nalaganje slike, Priprava slike, Iskanje objektov, Iskanje mej objektov in Rezultat iskanja v grobem refleksirajo korake v prej opisanem postopku.

5.1 Nalaganje slike

Slika 40: Nalaganje slike v uporabniški vmesnik

Uporabnik s pomočjo dialoga za izbiro datoteke sliko topografske karte naloži v uporabniški vmesnik.

Priprava slike



Slika 41: Obdelovanje slike s poljubno konvolucijsko masko

Uporabniku je v tem koraku, za izboljšavo slike, omogočena uporaba poljubne konvolucijske maske. Uporabniški vmesnik mu omogoča nalaganje obstoječe konvolucijske maske s podatkovnega medija ali pa kreiranje nove maske z ročnim vpisom vrednosti. Z gumbom *Auto div* se samodejno izračuna delitelj konvolucijske maske.

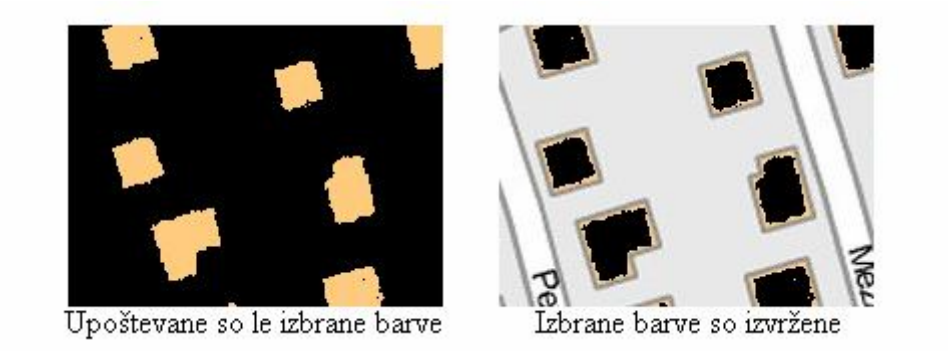
V drugem razdelku stranskega panela na levi (slika 41) je mogoče nastaviti prag gradienta. Ta se uporabi pri primerjanju gradienta in določanju roba na sliki.

Z gumbom *Priprava+Odvod* se na sliko aplicira trenutna konvolucijska maska ter izračuna odvod na sliki za preverjanje gradienta. Robovi slike, ki so rezultat odvoda in preverjanja gradienta, se zapišejo v sliko, ki bo uporabljena pri streljanju žarkov (jeziček *Iskanje mej objektov*).

V primeru, da slika ne potrebuje izboljšave, lahko uporabnik apliciranje konvolucijske maske preskoči s pritiskom na gumb *Samo odvod*.

5.2 Iskanje objektov

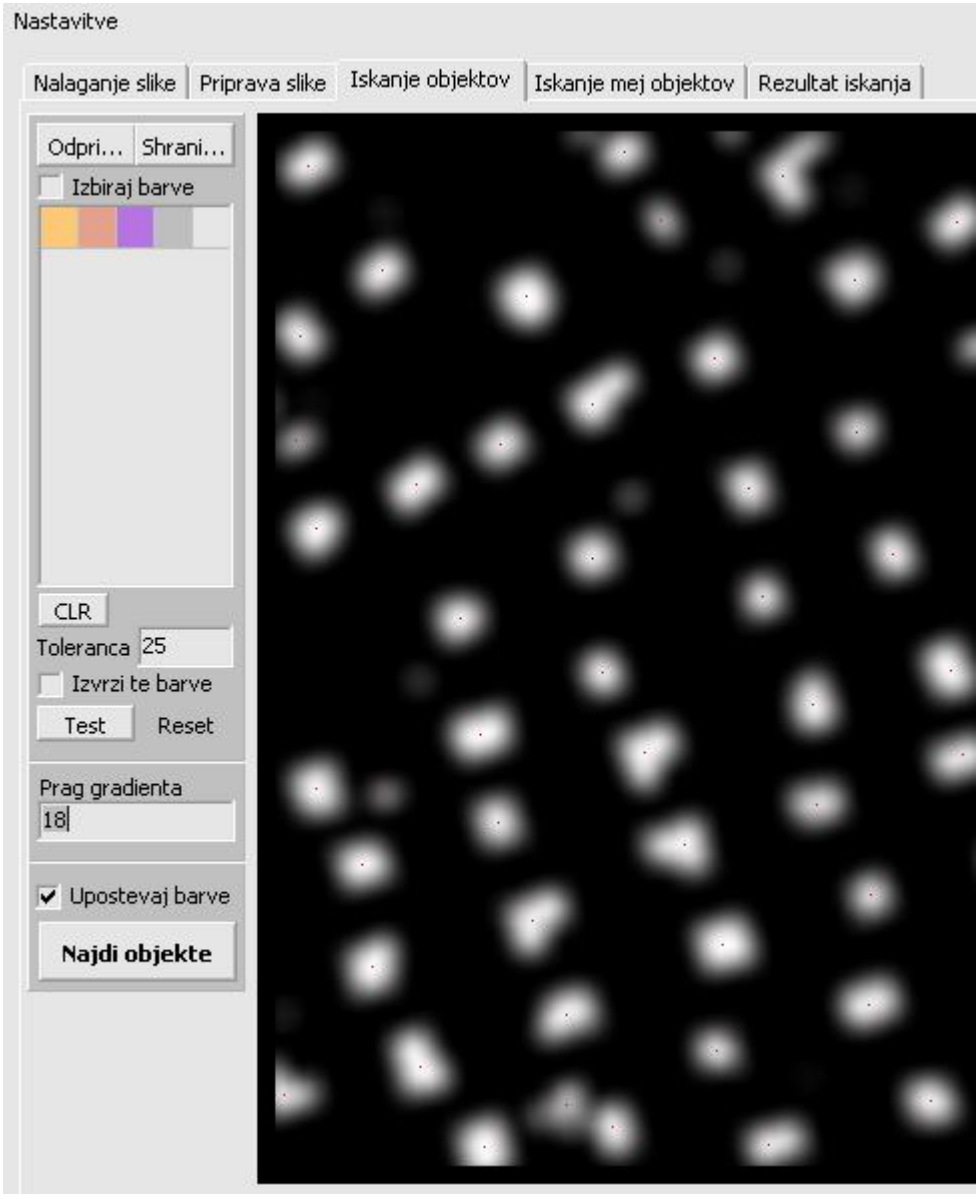
Uporabnik si lahko v prvem razdelku stranskega panela na levi (slika 43) izbere barve, ki predstavljajo iskane objekte. Za izbiranje barv je najprej potrebno označiti polje *Izbiraj barve* in nato izbrati barvo na sliki. Barva se uvrsti v razpredelnico, od koder jo s pritiskom na gumb *CLR* uporabnik lahko tudi odstrani. Nastavitev tolerance pove, kolikšno je dovoljeno odstopanje od izbranih barv. Izbor barv je mogoče tudi obrniti, če uporabnik označi polje *Izvrzi te barve*. To pomeni, da se tiste barve, ki so izbrane in nanizane v razpredelnici, ne upoštevajo v nadaljnjem procesiranju. Uporabnik lahko preveri, katera območja na sliki se bodo oziroma se ne bodo upoštevala (slika 42) s pritiskom na gumb *Test*. S tem se bo ustvaril predogled maske, ki bo aplicirana pri iskanju objektov.



Slika 42: Predogled maske s pritiskom na gumb *Test*

V drugem razdelku uporabnik nastavi prag gradienta. Ta se uporabi pri primerjanju gradienta in določanju roba na sliki, ki je že obdelana s Sobelovim operatorjem.

V tretjem razdelku ima uporabnik možnost, da s potrditvijo oziroma nepotrditvijo polja *Upostevaj barve* omogoči izločanje barv oziroma ga ne omogoči. Z gumbom *Najdi objekte* sproži celoten postopek za iskanje objektov.



Slika 43: Rezultat obdelave v jezičku *Iskanje objektov*

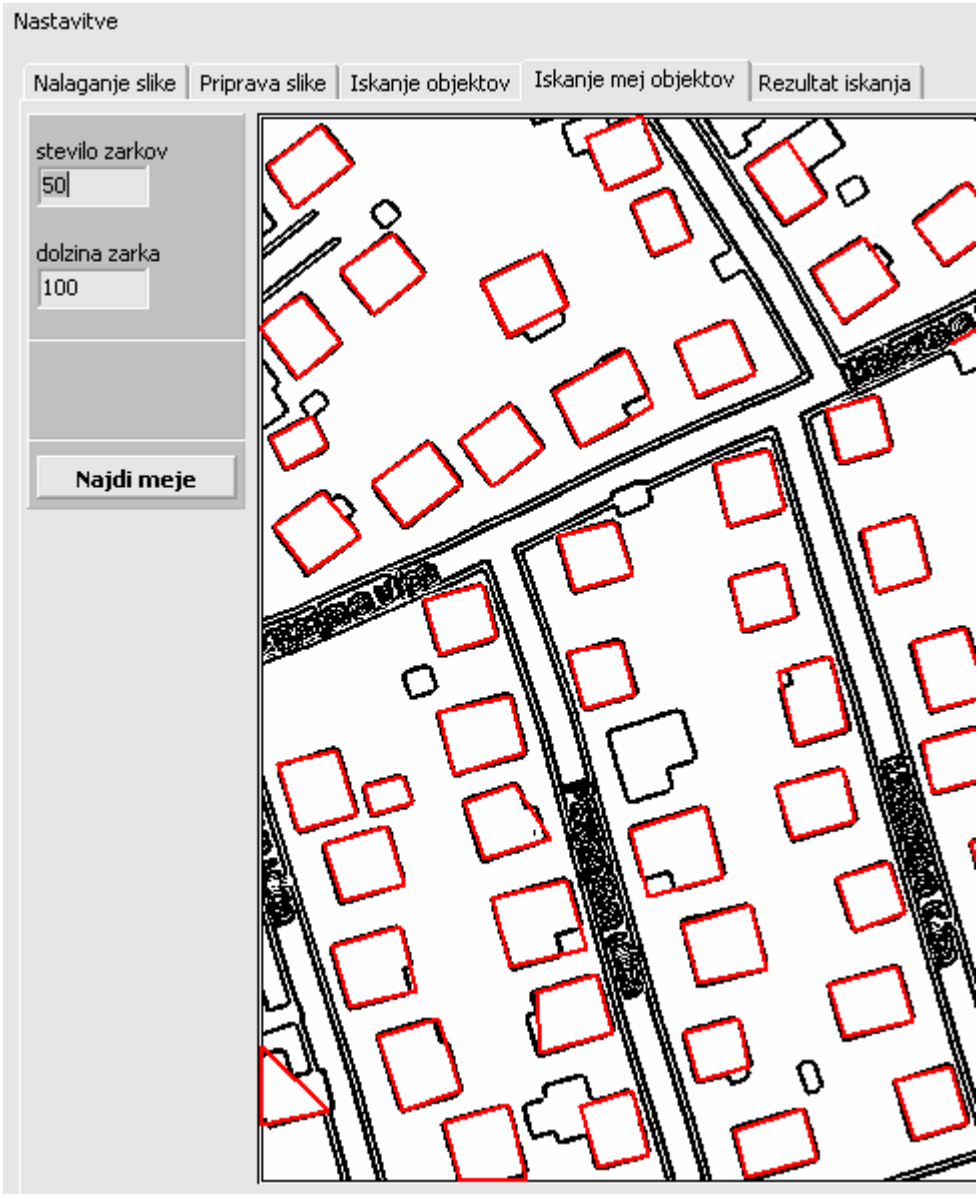
Psevdokoda detekcije vrhov na sliki:

```
radij = število pregledovanih točk

za i od 0 do širina slike - 1
  za j od 0 do višina slike - 1
    če slika[i,j] > prag vrhov
    potem začni
      vrh najden = true
      za ni od -radij do radij
        za nj od -radij do radij
          ii = i + ni
          jj = j + nj
          če jj > -1 in jj < višina slike
          potem začni
            če ii < 0 ali ii > širina slike - 1
            potem začni
              če ii < 0 potem ii = ii + širina slike
               ii = ii po modulu širina slike
               jj = višina slike - jj
            končaj
            če slika[i,j] < slika[ii,jj]
            potem
              vrh najden = false in končaj trenutno zanko
          končaj
        konec zanke nj
        če vrh najden = false potem končaj trenutno zanko
      konec zanke ni
      če vrh najden = true
      potem začni
        zapiši najdeno pozicijo
        popravi vrednost na najdeni poziciji
      končaj
    končaj

zapiši rezultat v uporabniški vmesnik
```

Časovna zahtevnost je O(širina slike * višina slike * n^2), pri čemer je n število pregledovanih točk v eni smeri od središčne točke (radij pregledovanih točk).

Zgoraj zapisani algoritem išče vrhove tudi v Houghovem akumulacijskem prostoru. Houghov akumulacijski prostor je grafična predstavitev glasovanja za parametre premic v polarni obliki. Vsebina slike glasovanja ima lastnost, da se na navpičnih robovih dopoljnuje. Tako lahko sliko "zvijemo" v valj in podatki se dopolnijo na obeh robovih slike. To lastnost upošteva tudi algoritem za detekcijo vrhov na sliki in temu je primerna tudi nekoliko večja kompleksnost le-tega.

5.3 Iskanje mej objektov

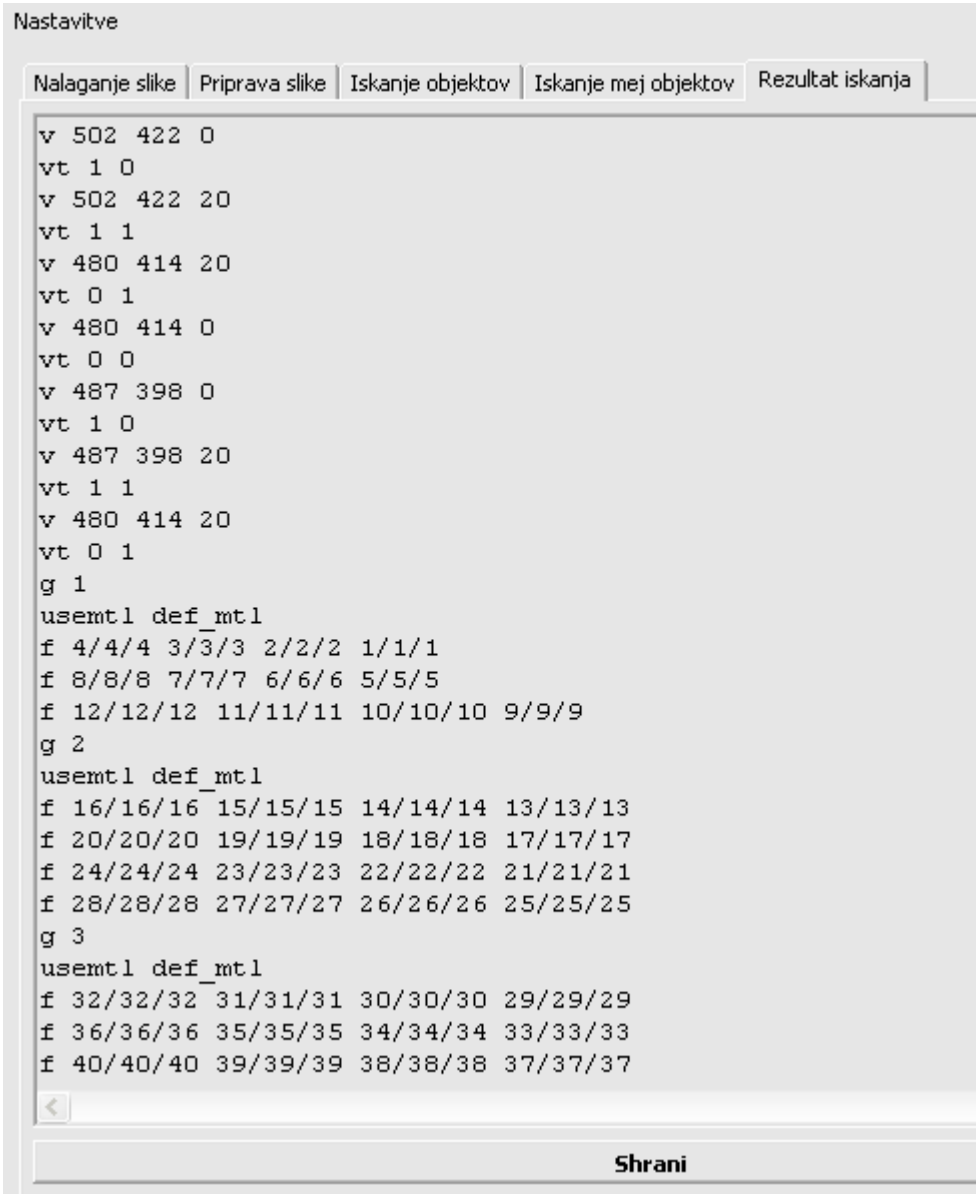


Slika 44: Najdene stranice objektov

Uporabnik lahko v prvem razdelku stranskega panela (slika 44) določi število žarkov, ki bodo izstreljeni iz vsake najdene točke (jeziček *Iskanje objektov*). Določi lahko tudi končno dolžino žarkov, ki se upošteva, če žarek ne trči z robom.

S pritiskom na gumb *Najdi meje* uporabnik požene postopek za iskanje mej objektov. Meje se po končanem postopku prikažejo na sliki kot rdeči mnogokotniki, ki omejujejo objekte (slika 44).

5.4 Rezultati iskanja



Slika 45: Zapis rezultatov iskanja

Tu uporabnik lahko pregleduje in shrani podatke, ki se bodo zapisali v datoteko tipa .obj. Prikazani podatki so že pripravljeni za prikaz v trodimenzionalnem prostoru. S pritiskom na gumb *Shrani* se videni podatki skupaj z datoteko materiala shranijo na željeno lokacijo, in to z imenom, ki ga vpiše uporabnik.

6 Zaključek

V diplomskem delu sem skušal uporabiti splošno znane in nekaj lastnih metod za iskanje objektov in njihovih stranic na topografskih kartah.

Sprva sem poskušal razbrati pozicije objektov na sliki, da bi to informacijo lahko uporabil za iskanje stranic. Tega sem se postopil z uporabo Sobelovega operatorja, računanja odvoda za preverjanje gradienta, konvolucijskih mask, vrhove na sliki pa sem iskal s pomočjo lastne metode. Celoten postopek se je pokazal kot dokaj uspešen, seveda ob upoštevanju dveh prej omenjenih predpostavk:

- da je objekt ločen od okolice z zaznavnim robom;
- da je objekt, ki ga iščemo, ali ustrezne barve ali pa je gradient njegove površine zelo blizu nič ali nič (je enobarven).

Nato sem poskušal zaznati rob vsakega objekta. Tega sem se lotil z metodo streljanja žarkov iz tistih točk, ki sem jih našel po postopku, opisanem v prejšnjem odstavku. Metoda deluje dobro, a le pri objektih preprostih oblik. Če pa so objekti bolj zapleteni in konkavni, začne metoda preveč posploševati omejujoče stranice ali pa žarki preprosto ne dosežejo vseh robov. Taki objekti potem izpadejo, saj jih procedura, ki preverja zaprtost mnogokotnika, izvrže. Te težave bi se dalo odpraviti z drugačnim pristopom; morebiti z žarki, ki bi se odbijali od robov, in pa z izboljšanjem zaznave ravnih črt na sliki.

Program, ki je nastal ob raziskovanju in preizkušanju metod za iskanje objektov in njihovih stranic, bi se dalo praktično uporabiti, denimo kot enostavno orodje za pomoč 3D modelarjem, kot orodje za predpripravo urbanega območja, na katerem je potrebno izračunati najboljšo pot (path finding), in kot orodje, ki generira osnovni predogled območja v 3D načinu. S prej naštetimi izboljšavami in še nekaterimi dodatnimi opcijami za uporabnika, kot naprimer:

- izbiranje med generičnimi oblikami streh in dodajanje le-teh na objekte,
- izbiranje med generičnimi teksturami za lica (faces) najdenih objektov,
- določanje in zaznava drugih topografskih značilnosti, kot so travniki, ceste, gozdovi,

pa bi bil program zelo uporaben pri izdelovanju virtualnih predstavitev poljubnih okolij.

Seznam slik

Literatura

[1] *Leksikoni Cankarjeve založbe, Geografija, tretja izdaja, 1985, str. 238.*

[2] *T. Banovec, Topografski priročnik, Partizanska knjiga, Ljubljana, 1972, str. 8.*

[3] *(2009) What is a Topographic Map? Dostopno na: http://maps.nrcan.gc.ca/topo101/faq_e.php*

[4] *(2009) Object files (.obj). Dostopno na: http://local.wasp.uwa.edu.au/~pbourke/dataformats/obj/*

[5] *(2009) MTL material format (Lightwave, OBJ). Dostopno na: http://local.wasp.uwa.edu.au/~pbourke/dataformats/mtl/*

[6] *(2009) Hough Line Transform. Dostopno na: http://planetmath.org/encyclopedia/HoughTransform.html*

[7] *(2009) Hough Transform. Dostopno na: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm*

[8] *(2009) Sobel Edge Detector. Dostopno na: http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm*

[9] *(2009) Interaktivni spletni atlas in zemljevid Slovenije. Dostopno na: http://www.geopedia.si*

[10] *(2009) Google maps. Dostopno na: http://maps.google.com*

[11] *Program Google Earth, ver. 4.3.7284.3916. Novejša verzija dostopna na: http://earth.google.com*

[12] *(2009) Color depth. Dostopno na: http://en.wikipedia.org/wiki/Color_depth*