UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Čerin

Iskanje in prilagajanje štirirobnih objektov na zaporedju fotografij s prezentacijskim platnom

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Tomaž Dobravec

Ljubljana, 2010

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: 00472/2009
Datum: 15.10.2009

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

Kandidat: **ANŽE ČERIN**

Naslov: **ISKANJE IN PRILAGAJANJE ŠTIRIROBNIH OBJEKTOV NA ZAPOREDJU FOTOGRAFIJ S PREZENTACIJSKIM PLATNOM**

THE DEVELOPMENT OF THE PROGRAM OF DETECTION AND CORRECTION OF QUADRILATERAL OBJECTS ON THE SET OF IMAGES WITH THE PRESENTATIVE PROJECTIONS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Izdelajte postopek, s katerim boste na seznamu fotografij samodejno zaznali prezentacijske projekcije, jim popravili perspektivo ter jih s tem usmerili neposredno k opazovalcu. Preglejte znane metode za iskanje objektov na slikah in ugotovite katere so najprimernejše za dani problem. Razvijte tudi lastne postopke, ki naj bodo prilagojeni specifičnim značilnostim fotografij s prezentacijskimi projekcijami.

Izdelajte računalniški program, v katerem boste prikazali delovanje razvitih metod na konkretnih fotografijah. Program naj bo napisan v programskem jeziku Java. Za zagotovitev čim večje prenosljivosti, se pri implementaciji metod izogibajte uporabi nestandardnih knjižnic. Program naj bo hiter in preprost za uporabo.

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 **Anže Čerin**,

z vpisno številko **63050240**,

sem avtor/-ica diplomskega dela z naslovom:

Iskanje in prilagajanje štirirobnih objektov na zaporedju fotografij s prezentacijskim platnom

S svojim podpisom zagotavljam, da:

sem diplomsko delo izdelal/-a samostojno pod mentorstvom

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 8.4.2010 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju doc. dr. Tomažu Dobravcu za vso pomoč, nasvete in usmeritve pri izdelavi diplomskega dela ter vsem, ki so me podpirali in pomagali v času študija.

Kazalo

Povzetek

Namen diplomskega dela je razvoj postopka, ki na seznamu fotografij samodejno zazna prezentacijske projekcije in jim popravi perspektivo na način, da so projekcije usmerjene neposredno k opazovalcu.

Pri razvoju postopka smo uporabili metode računalniškega vida in računalniške grafike. Obe področji računalništva sta predstavljeni v prvem poglavju. Pri računalniškem vidu predstavimo njegovo uporabo in ga primerjamo z računalniško grafiko. Pri predstavitvi računalniške grafike je poudarek na bitni oziroma rastrski grafiki in na postopkih s katerimi lahko sliko prikažemo ali spreminjamo njene lastnosti. V drugem poglavju je natančno opisan postopek iskanja štirirobnih objektov, ki sovpadajo s prezentacijskimi projekcijami na sliki. Predstavljenih je več postopkov za iskanje robov na sliki in iskanje ravnih črt s Houghovo linearno transformacijo. Sledi opis postopka, ki med najdenimi črtami poišče daljice in nato z njimi določi iskane štirirobne objekte na sliki. V tretjem poglavju je opisan postopek korekcije perspektive izbranega štirirobnega objekta z metodo perpektivne transformacije. Na koncu predstavimo še razviti program in njegovo praktično uporabo.

Ključne besede:

iskanje objektov, računalniški vid, računalniška grafika, iskanje robov, Houghova transformacija, interpolacija, perpektivna transformacija, konvolucija, prezentacijska projekcija, Gaussova eliminacijska metoda

Abstract

Subject of this bachelor's thesis is the development of a system capable of detecting presentative projections on the set of images and to correct their perspective in the way so that they are pointed directly to the viewer.

During the development of the program we used methods of computer vision and computer graphics. Both are decribed in the first chapter. We describe some computer vison's aplications and how it connects with computer graphics. When describing computer graphic we focus on bitmap or raster graphic and how we can use it to present images or to manipulate with them. In the second chapter we describe in detail steps required to detect quadrilateral shapes such as presentative projections. We present multiple edge detection algorithms and line detection using Hough linear transformation. Next we show how to use lines to get line segments and then use them to define quadrilateral shapes on the image. Third chapter describes the perspective correction of the selected shape using perpective tranformation. In the end we present the developed program and it's practical usage.

Key words:

object detection, computer vision, computer graphics, edge detection, Hough transform, interpolation, perpective tranform, convolution, presentative projection, Gaussian elimination

1 Uvodni del

Prezentacijske projekcije so sestavni del raznih predstavitev ali predavanj. Predavatelji velikokrat, zaradi boljše ponazoritve teme, uporabijo grafično predstavitev. Za razumevanje in dokazovanje v govoru pri predstavitvi projektne naloge predvidevamo uporabo učinkovitih grafičnih predstavitev v obliki ustreznih vizualnih sredstev, kot so prosojnice, diapozitivi, e-prosojnice, videoposnetki, plakati, tabelske slike in druga prezentacijska orodja. Te nato projicirajo na prezentacijsko platno na steni. Tako predstavitev ponavadi sestavlja serija zaporednih prosojnic ali drugih vizualnih sredstev, ki se menjavajo med predavanjem. Poslušalcu te prosojnice pogosto, v besedilni ali elektronski obliki, niso na voljo. Edini način, da ohrani nekaj predstavitvenega materiala, je, da ga posname oziroma fotografira. Nato mora, s pomočjo programa za obdelavo slik, na vsaki fotografiji določiti, kje se nahaja želena prezentacijska projekcija. Ker se ta v številnih primerih ne nahaja neposredno pred objektivom, ji moramo popraviti perspektivo, jo primerno povečati in shraniti. Tak postopek je zelo zamuden, zlasti pri velikem številu fotografij.

Slika 1: Primeri prezentacijskih platen

Namen diplomskega dela je razviti program, ki ta postopek poenostavi. Program na eni ali na zaporedju fotografij avtomatsko zazna štirirobne objekte, ki predstavljajo prezentacijske projekcije. Najdenim objektom nato samodejno popravi perspektivo, tako da so ti usmerjeni neposredno k opazovalcu.

Okolje, zajeto na fotografijah s prezentacijskimi projekcijami, ponavadi ni nadzorovano. Veliko aplikacij, ki se ukvarjajo z zaznavo objektov na slikah za uspešno in natančno zaznavo, zahteva točno določene pogoje. Ti so na primer lahko enobarvno ozadje, vnaprej določena velikost slike ali statičen položaj kamere, s katero so slike zajete. Lahko zahtevajo natančen opis iskanega objekta, katerega oblika mora biti na vseh vhodnih slikah zelo podobna.

Določiti projekcijo na fotografiji ni enostavno. Na fotografijah se ponavadi, poleg samih prejekcij nahajajo tudi gledalci, predavatelj, kosi pohištva, luči, slike na stenah in drugi predmeti. Program mora biti zato sposoben med vsemi temi objekti zaznati projekcijo.

Velik problem pri iskanju projekcije nam lahko predstavljajo različni svetlobni pogoji v prostoru. V zelo svetlih prostorih ali takrat, ko za projiciranje porabimo projektor z nizko svetilnostjo, se velikokrat zgodi, da projekcija ne bo imela vseh robov vidnih. Možno je celo, da jih ne bo imela in jo je zato skoraj nemogoče natančno določiti.

Najbolj so projekcije izrazite takrat, ko je edini vir svetlobe v sobi projekcija sama. V takem primeru jo je precej enostavno zaznati, ker predstavlja edini vidni objekt na sliki, medtem ko so ostali objekti zatemnjeni oziroma nevidni. Velikokrat se tudi zgodi, da predavatelj stopi pred samo projekcijo in jo s tem delno zakrije. Razviti program je dovolj zmogljiv, da zna predvideti, kje za njim se nahaja projekcija, če ta le ni preveč zakrita.

Program ni omejen samo na iskanje prezentacijskih projekcij. Lahko ga uporabimo za iskanje poljubnih štirirobnih objektov na vsebinsko zelo bogatih slikah.

Prilagojen program bi lahko uporabili:

- za klasifikacijo podobnih štirirobnih objektov na fotografijah,
- za analizo tekem na štirirobnih tekmovalnih površinah (npr. bilijard, nogomet, košarka),
- za samodejno zaznavo reklamskih panojev,
- za zajem slik iz dokumentov,
- za skeniranje dokumentov pod kotom.

1.1 Uporabljene oznake pojmov

Pri opisu postopkov iskanja prezentacijskih projekcij na sliki uporabljamo naslednje pojme:

x - koordinata x
y - koordinata y
p - število vseh slikovnih točk na sliki
e - število robnih točk na sliki robov
n - red konvolucijske maske
f(x,y) - funkcija, ki vrne intenziteto slikovne točke na koordinatah (x, y)
e(x,y) - funkcija točk na sliki robov, vrne 1 če je na koordinatah (x, y) rob, drugače 0
g(x,y) - funkcija gradientov (odvodov), ki vrne skupni približek gradienta na koordinatah (x,y)
h(x,y) - funkcija približkov drugih odvodov na koordinatah (x,y)
G_x - približek gradienta v smeri x
G_y - približek gradienta v smeri y
θ - naklonski kot

2 Analiza in sinteza slik

2.1 Računalniški vid

Računalniška grafika se ukvarja s sintezo slik iz umetno ustvarjenih objektov. Analiza slik pa je ravno nasprotni proces, ki se ukvarja z obstoječimi slikami [4].

Slika 2: Prikaz razmerja med računalniškim vidom in računalniško grafiko

Računalniški vid je področje računalništva, ki se ukvarja z analizo slik in interpretacijo objektov na njih. V nekaterih pogledih je nasprotje računalniški grafiki. Medtem ko se slednja ukvarja s sintezo slik iz modelov, je naloga računalniškega vida, da na podlagi slike izdela model, ki kar najbolje opisuje tisto, kar je na njej predstavljeno. Cilj računalniškega vida je razpoznati in razumeti nek prizor ali objekt na osnovi vizualne informacije, na podoben način, kot to stori človek.

Zaradi izjemne časovne kompleksnosti pri reševanju problemov je največji problem za računalniški vid predstavljala prenizka zmogljivost računalnikov. To je tudi vzrok za njegov relativno pozen razvoj. Prve resnejše raziskave segajo šele v 70. leta prejšnjega stoletja. Zaradi zelo dragih sistemov, potrebnih za implementacijo, je bil ta včasih poglavitna domena večjih organizacij, kot je na primer vojska. S povečanjem procesorske moči in pocenitvijo računalniške opreme pa je računalniški vid postal dostopen večjemu krogu raziskovalcev, kar je še pospešilo njegov razvoj.

Danes je razvitih veliko različnih aplikacij, ki se uporabljajo na številnih področjih v vsakdanjem življenju.

Medicina

V medicini se računalniški vid uporablja pri interpretaciji rentgenskih, tomografskih in mikroskopskih slik. Z ugotavljanjem meja obolelega tkiva na njih povečamo uspešnost kirurških posegov. To je zlasti pomembno pri rakavih obolenjih, kjer je poznavanje njihove raširjenosti bistvenega pomena za nadaljnje postopke zdravljenja.

Vojska

Vojska vlaga veliko denarja v razvoj naprednih bojnih sistemov, ki so sposobni sami identificirati tarčo in ji po potrebi slediti. Take sisteme najdemo v bojnih letalih ali vodenih raketah (angl. cruise missile). Pomembno vlogo ima tudi pri nadzoru bojnega področja. Tukaj govorimo predvsem o analizi satelitskih slik, kjer je interpretacija dogajanja ključnega pomena za strateške odločitve.

Industrija

V industriji se računalniški vid uporablja predvsem pri kontroli kakovosti v proizvodnih procesih in pri nadzoru pretoka materiala. Nadzor kakovosti je bistven del proizvodnega procesa v avtomobilskih, prehrambenih, fermacevtskih in ostalih panogah, kjer je pomembna kakovost in skladnost izdelkov. Ker gre tukaj ponavadi za zelo delikatne in monotone postopke, so to nalogo, za katero je bil včasih potreben človek, prevzeli računalniki. Ti so veliko bolj natančni in hitri. Za razliko od človeka se ne utrudijo, s čimer se zmanjša tudi število napak.

Video nadzor

V zadnjem času so priljubljeni varnostni sistemi za avtomatsko zaznavo in identifikacijo ljudi na slikah ali video posnetkih. Ti so zmožni določiti identiteto človeka na podlagi obraza ali drugih biometričnih značilnosti. Identifikacija je možna, recimo, z analizo prstnih odtisov ali očesne roženice. Taki sistemi se v današnjem času čedalje bolj uporabljajo na letališčih, mejnih prehodih, bankah in drugih javnih mestih. Z uporabo nadzornih video kamer lahko sledimo prometu na več mestih hkrati in na podlagi analize posnetkov tudi izvedemo ustrezne postopke proti kršiteljem prometnih predpisov in tistim, ki ogrožajo druge udeležence v prometu.

2.2 Računalniška grafika

Računalniška grafika se ukvarja z ustvarjanjem, pomnjenjem, predstavitvijo, obdelavo in uporabljanjem grafičnih objektov s pomočjo računalnika in s povezavo grafičnih objektov z ustreznimi negrafičnimi informacijami. Objekti so lahko krivulje, ploskve, telesa. Predstavimo jih lahko v dvodimenzionalnem ali tridimenzionalnem prostoru na papirju ali zaslonu.

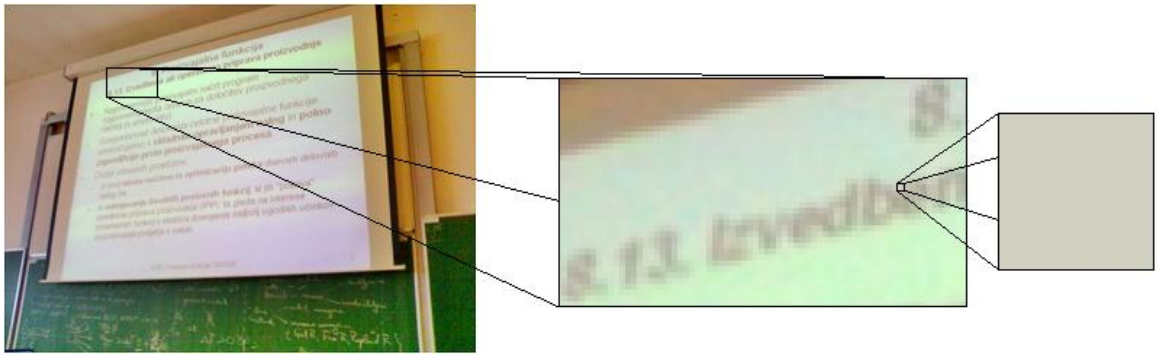
Računalniško grafiko uporabljamo:

- za uporabniške vmesnike,
- za simulacije in animacije,
- za računalniško podprto učenje,
- za navidezno rasničnost,
- za računalniško podprt prikaz in obdelavo slik,
- v medicini,
- za multimedijske sisteme.

Osredotočili se bomo na prikaz in obdelavo slik z računalnikom. Sliko v računalniku predstavimo z bitno grafiko.

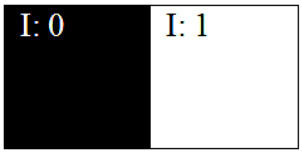
2.2.1 Bitna grafika

Bitna grafika je pojem, povezan z računalniško sliko, ki je sestavljena iz množice slikovnih točk (pikslov), razporejenih v mreži z določeno gostoto [1]. S sestavljanjem posameznih točk v mreži pridemo do bitne slike. Čim večje je število točk, ki sestavljano sliko, večja je ločljivost slike in več podrobnosti je na sliki vidnih. Vrednost vsake točke je predstavljena z barvnim modelom**,** s katerim predstavimo njeno barvo kot kombinacijo večih številčnih vrednosti. Poznamo veliko različnih modelov. Med najbolj poznane sodijo RGB, CMYK, HLS in RYB. Število različnih barv, ki jih je model sposoben prikazati, je odvisno od števila bitov, ki jih ima model na voljo. Zato to smer v računalniški grafiki imenujemo bitna grafika. Nekateri jo imenujejo tudi rastrska grafika.



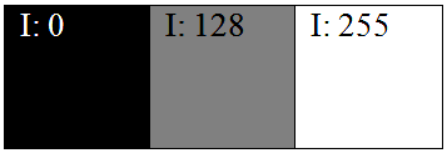
Slika 3: Slikovne točke na bitni sliki

Najpreprostejša slika je črno-bela ali enobarvna, pri kateri imamo opraviti z enobitno grafiko. Enobitne slike uporabljajo en bit za opis vrednosti točke. Prikazati so sposobne samo dve barvi. Najpogosteje prikažemo črno in belo.



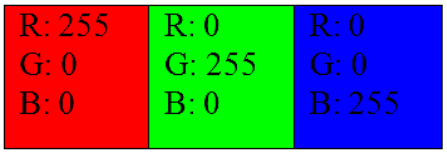
Slika 4: Predstavitev barv na bitni sliki z uporabo enega bita na slikovno točko

Sivinska slika za razliko od enobitnih slik za opis vrednosti točke, uporablja večje število bitov. Tako lahko poleg bele in črne barve prikaže tudi več vmesnih odtenkov sivine. Če uporabimo osem bitov za opis ene točke, jo lahko predstavimo z 256 različnimi odtenki sivine.



Slika 5: Predstavitev barv na sivinski sliki z uporabo osmih bitov na slikovno točko

Za prikaz barvnih slik najpogosteje uporabljamo barvni model RGB. Model uporablja tri kanale za prikaz rdeče, zelene in modre barve. S kombinacijo teh kanalov lahko prikaže velik razpon barvnih odtenkov. Čim več bitov ima model na voljo, bolj kvalitetne bodo barve na sliki. Najbolj pogosto se uporablja 24 bitov na slikovno točko. Vsakemu od treh kanalov je v tem primeru na voljo 8 bitov. Tako je kanal zmožen prikazati 2^8, oziroma 256 različnih odtenkov rdeče, zelene in modre. S skupno uporabo teh treh kanalov lahko prikažemo 16.777.216 različnih barvnih odtenkov.



Slika 6: Predstavitev barv z barvnim modelom RGB ter uporabo 24 bitov na slikovno točko

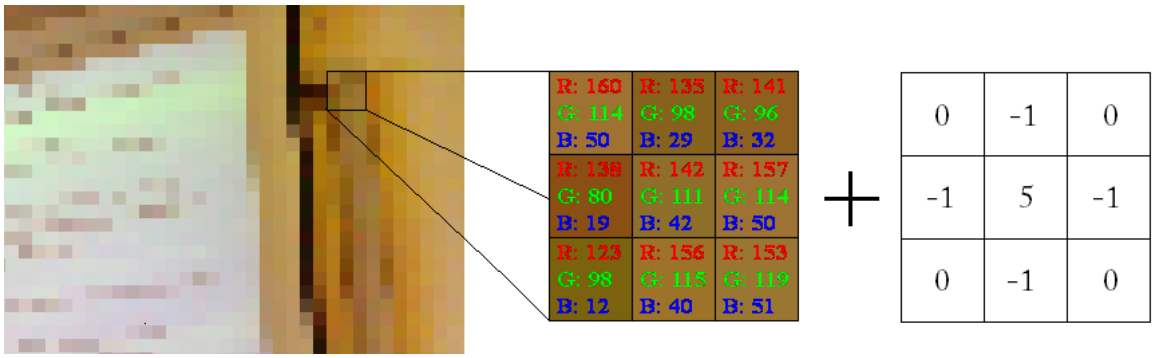
Rastrska grafika se uporablja v programih za slikanje in urejanje slik. Z njimi lahko spreminjamo lastnosti slike. To je pomembno predvsem pri urejanju fotografij. Fotoaparati namreč shranijo zajete fotografije v rastrski obliki in velikokrat se zgodi, da zajete slike niso idealne. Lahko so pretemne ali presvetle, velikokrat podrobnosti na sliki niso dovolj izrazite. Fotografije lahko računalniško, s pomočjo namenskih programov, popravimo. Ti že vsebujejo različne postopke za obdelavo slik (filtre). Z uporabo slednjih lahko popravimo sliki svetilnost, kontrast, poudarimo medsebojna razmerja barv in podobno. Filtri delujejo tako, da spreminjajo vrednosti slikovnih točk na sliki, za kar uporabljajo postopek konvolucije.

2.2.1.1 Konvolucija

Konvolucija je postopek, pri katerem s pomočjo konvolucijskih mask različnih velikosti spremenimo lastnosti vhodne slike. Bistvena je pri računalniški obdelavi in analizi slik. Pri konvoluciji slike potujemo skozi vsako slikovno točko na njej in nanjo položimo konvolucijsko masko. Nato izračunamo zanjo novo vrednost, tako da vzamemo slikovne točke, ki ležijo pod masko. Njihove vrednosti uravnotežimo glede na polja v maski, nato pa jih med seboj seštejemo, da dobimo novo vrednost slikovne točke (slika 4). Pri barvnih slikah je potrebno postopek ponoviti za vsak barvni kanal. Povprečna intenziteta slikovnih točk se ohranja, če je vsota vseh polj v konvolucijski maski enaka ena. Če je skupna vsota manjša dobimo temnejšo, če večja, pa svetlejšo sliko. Kadar skupna vsota polj v konvolucijski maski ni enaka ena, mi pa bi radi ohranili intenzitete slikovnih točk, dobljene vrednost pomnožimo s faktorjem, ki nevtralizira njihove vrednosti. Primer maske, pri kateri moramo ohraniti intenzitete točk, je maska za glajenje (slika 5). Pri konvoluciji moramo tudi paziti, da novih vrednosti ne zapisujemo nazaj v vhodno sliko. V tem primeru bi zaradi spremenjenih vhodnih slikovnih elementov dobili napačne vrednosti.

Časovna kompleksnost:

- O(p * n^2)



Slika 4: Primer apliciranja konvolucijske mase

Novo vrednost slikovne točke dobimo z računanjem novih vrednosti za vsak barvni kanal:

$$R = (0 * 160) + (-1 * 135) + (0 * 141) + (-1 * 138) + (5 * 142) + (-1 * 157) + (0 * 123) + (-1 * 156) + (0 * 153) = 124$$

$$G = (0 * 114) + (-1 * 98) + (0 * 96) + (-1 * 80) + (5 * 111) + (-1 * 114) + (0 * 98) + (-1 * 115) + (0 * 119) = 148$$

$$B = (0 * 50) + (-1 * 29) + (0 * 32) + (-1 * 19) + (5 * 42) + (-1 * 50) + (0 * 12) + (-1 * 40) + (0 * 51) = 72$$

S konvolucijskimi maskami lahko sliko izostrimo, tako da povečamo kontrast med sosednje ležečimi točkami na sliki (angl. sharpening). S tem povečamo število vidnih podrobnosti na njej. Lahko naredimo tudi obraten postopek in sliko zameglimo (angl. blur). V tem primeru sliko zamažemo, tako da zmešamo barve med sosednje ležečimi točkami. Na sliki lahko tudi ustvarimo tridimenzionalni senčni videz slike in jo prikažemo kot relief (angl. embossing).
Med pomembnejše naloge konvolucije sodi iskanje robov, ki je podrobneje opisano v naslednjem poglavju.

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

0	-1	0
-1	5	-1
0	-1	0

-1	0	0
0	0	0
0	0	1

Slika 5: Od leve proti desni: maska za glajenje, maska za ostrenje, maska za prikaz reliefa

2.2.1.2 Interpolacija

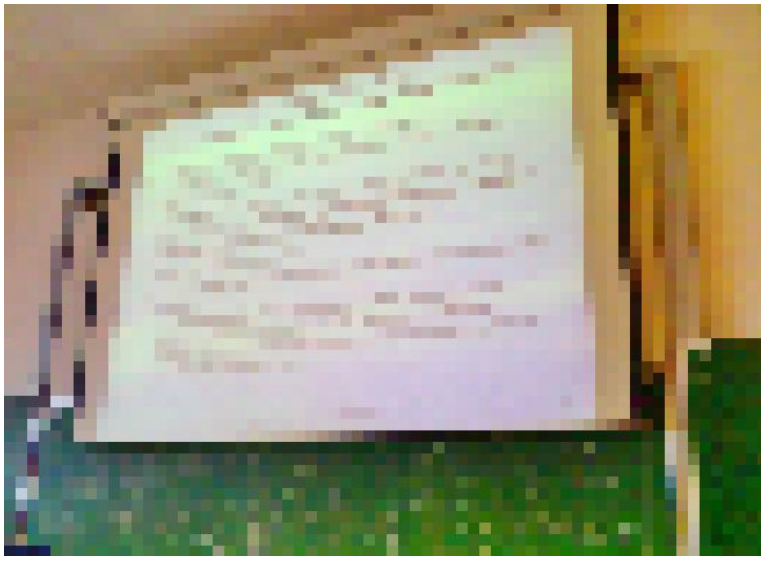
Sliko je velikokrat potrebno raztegniti oziroma skrčiti. To dosežemo s povečanjem ali zmanjšanjem števila barvnih točk, ki jo sestavljajo. Pri tej operaciji se velikokrat poslužujemo postopka interpolacije, da ohranimo čim več podrobnosti, ki bi se drugače ob transformaciji slike lahko izgubile.

Interpolacija je v matematiki neka približna vrednost funkcije znotraj obsega znanih vrednosti spemenljivke. V našem primeru so znane vrednosti spremenljivke vrednosti slikovnih točk, ki se nahajajo okoli točke, katere vrednost interpoliramo; vrednost funkcije pa je iskana barvna vrednost.

Vsakič ko spremenimo dimenzije slike, hkrati spremenimo tudi število slikovnih točk, ki jo sestavljanjo. Pri vsaki transformaciji slike je potrebno slikovne točke iz originalne slike preslikati na primerno mesto na novi transformirani sliki. Lahko storimo tudi obratno in za vsako slikovno točko na ciljni sliki poiščemo pripadajočo točko na izvorni. Postopek imenujemo obratna transformacija. Pri transformaciji nam na novi sliki ponavadi ostanejo tudi točke, ki jim s transformacijo nismo mogli določiti slikovnih točk iz originalne slike. Tem točkam moramo dodeliti neko barvno vrednost, ki mora biti čim bolj podobna vrednosti njej sosednjih točk. Čim bolj pravilno smo izbrali barvne vrednosti, bolj kakovostno sliko bomo na koncu dobili. Barvne vrednosti teh točk dobimo s pomočjo interpolacije.

2.2.1.2.1 Metoda najbližnjega soseda

Poznamo več interpolacijskih metod. Najbolj osnovna in tudi najhitrejša je metoda najbližnjega soseda. Pri tej metodi za novo vrednost slikovne točke vzamemo kar vrednost najbližje znane točke. Čeprav je ta metoda hitra in preprosta, daje povprečne rezultate. Tako dobljene slike so ponavadi videti zelo nazobčane (slika 6).



Slika 6: Slika, povečana z metodo najbližnega soseda

2.2.1.2.2 Bilinearna interpolacija

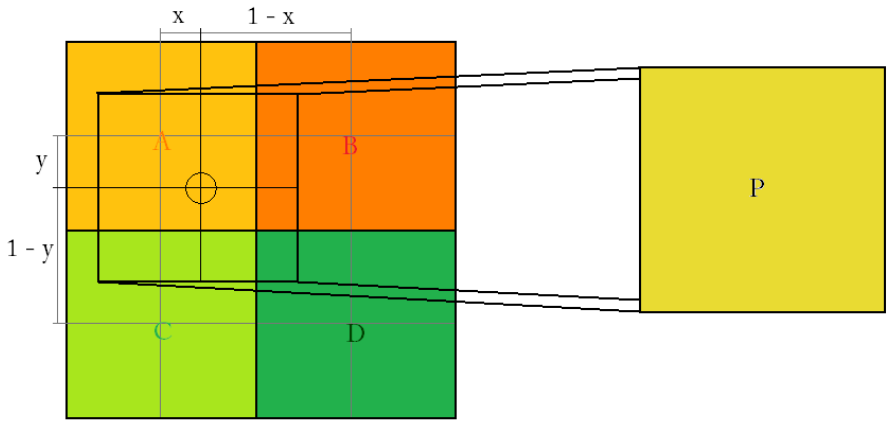
Boljša metoda je bilinearna interpolacija [7, 8]. Metoda uporablja večje število sosednjih slikovnih točk, z njo pa dosežemo boljše rezultate (slika 7). Pri interpolaciji nove barvne vrednost upošteva uravnotežene vrednosti štirih slikovnih točk, ki so najbližje točki, katere barvno vrednost interpoliramo. Čim bližje se slikovna točka nahaja, več barvne vrednoti bo prispevala ob interpolaciji. Pri barvnih slikah moramo interpolirati vsak barvni kanal posebej.

Slika 7: Slika, povečana z uporabo bilinearne interpolacije

Z uporabo obratne transformacije lahko za interpolacijo uporabimo sosednje barvne točke (slika 8). Novo interpolirano vrednost dobimo z uporabo naslednje enačbe:

$$P = (1 - x)(1 - y) * A + x * (1 - y) * B + (1 - x) * y * C + x * y * D$$

- Spremenljivki x in y sta novi koordinati slikovne točke, ki jo dobimo z obratno transformacijo.
- Spremenljivke A, B, C in D nam predstavljajo barvne vrednosti štirih sosednjih slikovnih točk.
- Spremenljivka P je nova interpolirana barvna vrednost preslikane točke.



Slika 8: Prikaz bilinearne interpolacije nad štirimi slikovnimi točkami

3 Podrobni opis iskanja štirirobnih objektov

V tem poglavju bomo podrobno predstavili postopek iskanja štirirobnih objektov na sliki.

Povzetek postopka za iskanje štirirobnih objektov:

- Na začetku sliko po potrebi zmanjšamo, s čimer zmanjšamo čas iskanja objektov.
- Barvno sliko nato pretvorimo v sivinsko.
- Na sivinski sliki poiščemo robove in jih shranimo v sliko robov.
- S Houghovo linearno transformacijsko metodo poiščemo na sliki robov vse ravne črte (premice).
- Poiščemo daljice, ki ležijo na teh premicah in se hkrati nahajajo na robu.
- Na koncu poiščemo vse kombinacije daljic, ki predstavljajo štirirobne objekte in so primerne oblike.

3.1 Spreminjanje velikosti slike

Analiza velikih slik je računsko zelo zahtevna, ker morajo algoritmi v večini primerov upoštevati vse barvne točke na sliki. Čim večja je slika in iz večjega števila točk je sestavljena, več časa bo program potreboval, da bo na njej določil morebitne štirirobne objekte.

Ker pri določanju koordinat štirirobnih objektov ne potrebujemo popolne natančnosti, postopek iskanja pospešimo tako, da vsako sliko na začetku zmanjšamo na določeno velikost. Pri tem ohranimo razmerja stranic slike. Enakomernemu spreminjanju velikosti slike pravimo tudi skaliranje.

Vsaka slika mora imeti za naslednje korake v analizi manjše ali enako število točk od predpisanega števila. Omejili smo se na 300.000 točk na sliko.

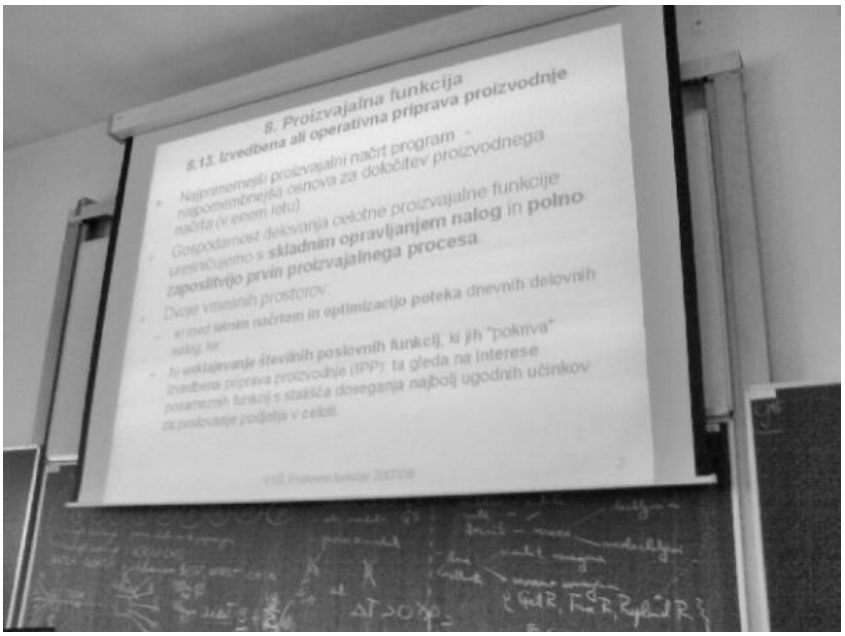
Če je slika večja in ima več točk, jo toliko zmanjšamo, da skupno število njenih točk ne presega tega števila. V primeru da je slika manjša, ohranimo njeno originalno velikost.

Upoštevati moramo, da tako na koncu izgubimo nekaj na natančnosti prikaza najdenih štirirobnih objektov. Z zmanjševanjem slike dobimo namreč napako pri koordinatah odkritih površin. Ta je v praksi zelo majhna, ker sliko zmanjšamo samo za majhen faktor. Koordinate najdenih površin se od pravih koordinat razlikujejo samo za nekaj slikovnih točk.

Sliko zmanjšamo tako, da zmanjšamo število barvnih točk, ki jo sestavljajo. S tem izgubimo nekaj informacij iz slike, kar pa se pri iskanju večjih površin, kot so prezentacijske projekcije, izkaže za pozitivno. Ko sliko zmanjšamo, izločimo na njej tudi manjše podrobnosti, ki se lahko nahajajo na sami površini ali okoli nje in so za nas nepomembne. S tem zmanjšamo kompleksnost slike, tako da postane celotna iskana oblika štirirobnega objekta bolj izrazita.

3.2 Pretvorba barvne v sivinsko sliko

Pri iskanju objektov na sliki nas ne zanimajo barve slikovnih točk, temveč samo njihove intenzitete. Algoritmi za iskanje robov, ki se uporabijo v naslednjih korakih, temeljijo na iskanju gradientov intenzitet in ne potrebujejo barvne informacije o točkah. Barvno sliko zato pretvorimo v sivinsko sliko (slika 9) in s tem poenostavimo nadaljnji postopek analize slike.



Slika 9: Sivinska slika

Barvno sliko lahko pretvorimo v sivinsko s povprečenjem vrednosti barvnih točk. Vrednosti novih sivinskih slikovnih točk dobimo tako, da seštejemo vrednosti barvnih kanalov istih točk in dobljeno vsoto delimo z njihovim številom.

```
/** psevdo koda za pretvorbo barvne slike v sivinsko 1 **/

za vsako slikovno točko P na sliki
   P.intenziteta = (P.rdeča + P.zelena + P.modra)/3
```

Nekoliko boljše rezultate dobimo, če upoštevamo tudi intenzitetne uteži posamezdnih kanalov. Vsak barvni kanal namreč ne prispeva enakega deleža končni vrednosti intenzitete slikovne točke. Zeleni kanal ima med vsemi kanali najmanjši kontrast, modri pa največjega. Standardizirane uteži za model RGB so 0,299 za rdeči, 0,587 za modri in 0,114 za zeleni kanal.

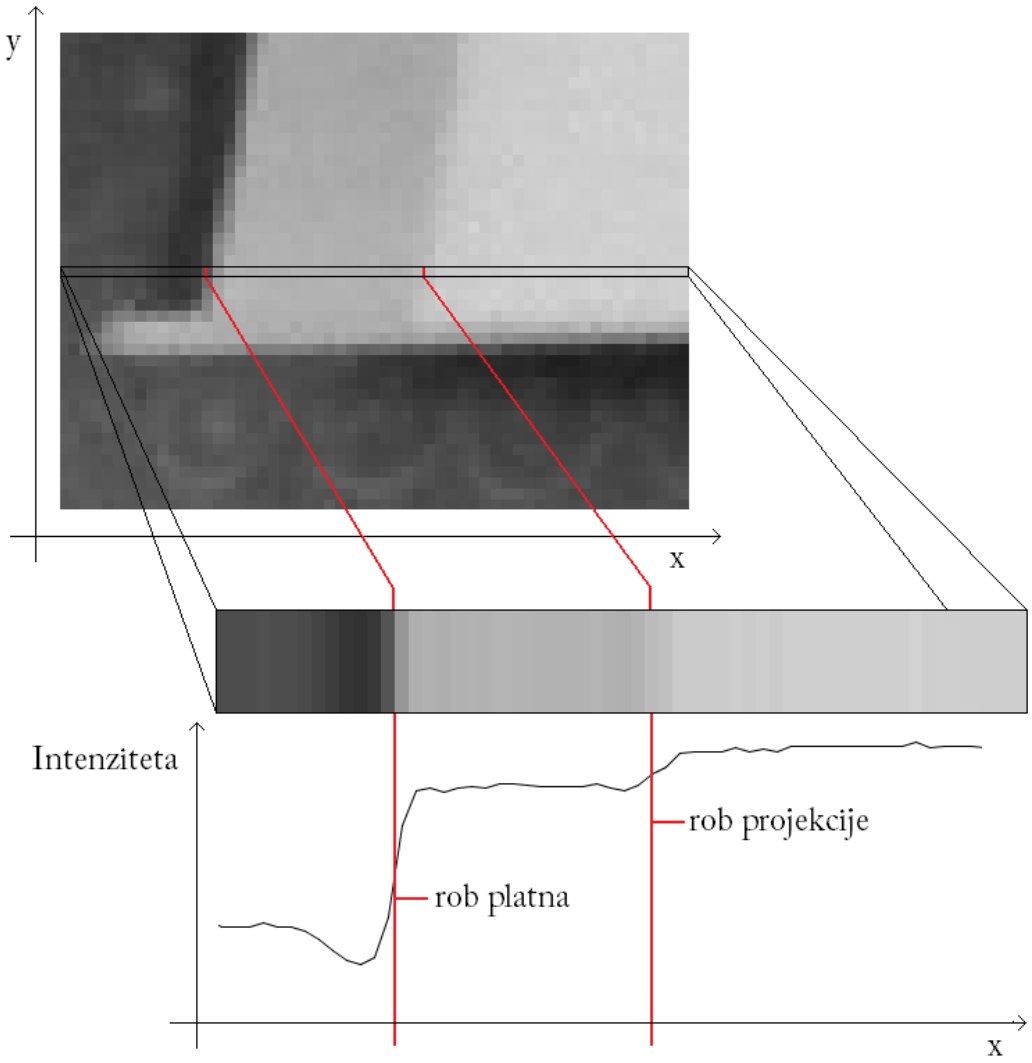
```
/** psevdo koda za pretvorbo barvne slike v sivinsko 2 **/

za vsako slikovno točko P na sliki
   P.intenziteta = P.rdeča  * 0,299 +
                   P.zelena * 0,587 +
                   P.modra  * 0,114
```

Na tej stopnji ne govorimo več o barvah točk, temveč o njihovih intenzitetah.

3.3 Iskanje robov

V računalniškem vidu je ena najpomembnejših operacij pri sliki iskanje robov. Ti na sliki predstavljajo meje zaključenih oblik, ki so v našem primeru tudi prezentacijske projekcije (slika 10). Lahko pa predstavljajo tudi ločnico med svetlobo in senco, ki pada na posamezno površino. S pretvorbo slike v sliko robov, na kateri so vidni samo robovi, izločimo iz nje veliko količino za nadaljnjo analizo slike nepomembnih podatkov.

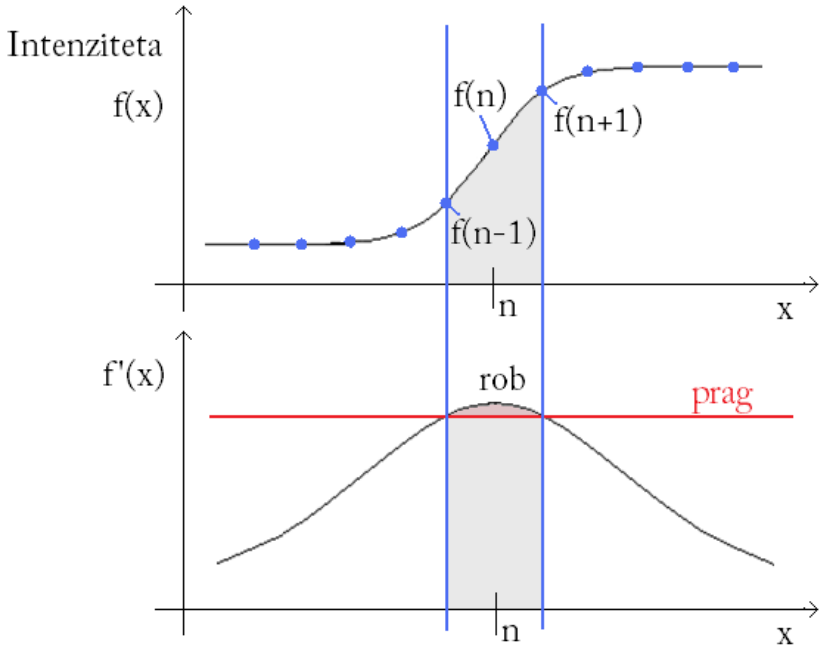
Najdene robove prikažemo z dvodimenzionalno tabelo, kjer imajo polja z robovi vrednost ena, ostala pa nič. Zaradi boljše vizualne predstave in poenostavitve nadaljnih postopkov pri analizi slike jih velikokrat predstavimo kar z enobitno sliko. Slikovne točke, ki predstavljajo robove, so na njej bele, ostale pa črne barve.



Slika 10: Spremembe v intenzitetah med slikovnimi točkami

3.3.1 Iskanje robov na podlagi odvoda

Formalno so robovi na sliki predstavljeni kot nenadne spremembe v intenzitetah sosednjih slikovnih točk. Čim hitrejši je prehod v intenzitetah teh točk (gradient), večja je verjetnost, da se nahajajo na robu. Hitrost spremembe v intenzitetah dobimo tako, da izračunamo odvod [5]. Če je vrednost odvoda večja od nekega vnaprej določenega praga, potem smo na tem mestu naleteli na rob (slika 11).



Slika 11: Odvod intenzitet točk na robu

Odvod f'(x) je definiran kot sprememba vrednosti funkcije f(x) pri spremembi njenega argumenta Δx:

$$f'(x) = \frac{d(f(x))}{dx}$$

Približno vrednost odvoda f'(x) pri vrednosti argumenta x = n izračunamo po naslednji formuli:

$$f'(n) = \frac{d(f(n))}{dn} = \frac{f(n+1) - f(n-1)}{(n+1) - (n-1)} = 0.5 * (f(n+1) - f(n-1))$$

Slikovna funkcija f(x,y) nam vrne intenziteto slikovne točke na x in y koordinatah bitne slike. Ker je slika dvodimenzionalno polje točk, moramo z uporabo parcialnih odvodov izračunati približka gradientov robov v smereh x (Gx) in y (Gy). S prvim zaznamo navpične robove, z drugim pa vodoravne robove.

Odvod po x-u:

$$\frac{\partial f(x,y)}{\partial x} = Gx = 0.5 * (f(x+1,y) - f(x-1,y))$$

Odvod po y-u:

$$\frac{\partial f(x,y)}{\partial y} = Gy = 0.5 * (f(x,y+1) - f(x,y-1))$$

Približke teh odvodov lahko na bitni sliki dobimo z apliciranjem dveh konvolucijskih mask (z levo dobimo Gx, z desno Gy):

$$0.5 * \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad 0.5 * \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Dobljeni vrednosti obeh gradientov (odvodov) na koncu združimo, da dobimo absolutno intenziteto gradienta. Absolutna vrednost gradienta na slikovni funkciji $\nabla f(x,y)$ dobimo na naslednji način:

$$g(x,y) = \sqrt{Gx^2 + Gy^2} = \nabla f(x,y) = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$$

Zaradi hitrejšega računanja ponavadi uporabimo kar približek:

$$g(x,y) = |Gx| + |Gy|$$

Če je absolutna intenziteta gradienta večja od praga, potem se slikovna točka, na koordinatah (x,y), nahaja na robu:

$$e(x,y) = \begin{cases} 1, & g(x,y) > prag \\ 0, & sicer \end{cases}$$

Vrednost praga moramo vnaprej določiti. Če določimo visok prag, bomo dobili samo najbolj izrazite robove, se pravi tiste z največjim naklonom (gradientom) in s tem največjo vrednostjo odvoda. Ob izbiri nizkega praga pa bomo velikokrat poleg pravih dobili tudi robove, ki ne predstavljajo meje dveh neenakih regij v sliki, temveč so lahko posledice motenj (šuma) na sliki. Šum je na sliki viden kot niz drobnih pikic, ki so bolj vidne v temnejših delih slike. Če je bolj izrazit, je lahko navzoč tudi na celotni sliki.

Primera detektorjev robov, ki temeljita na računanju odvoda, sta Robertsov in Sobelov operator.

3.3.1.1 Robertsov operator

Robertsov operator je eden prvih algoritmov za zaznavo robov [9]. Sestavljata ga dve konvolucijski maski drugega reda. Druga maska je enaka prvi, rotirani za 90°. Maski sta narejeni tako, da najbolje zaznata robove, ki potekajo pod kotom 45° glede na os x.

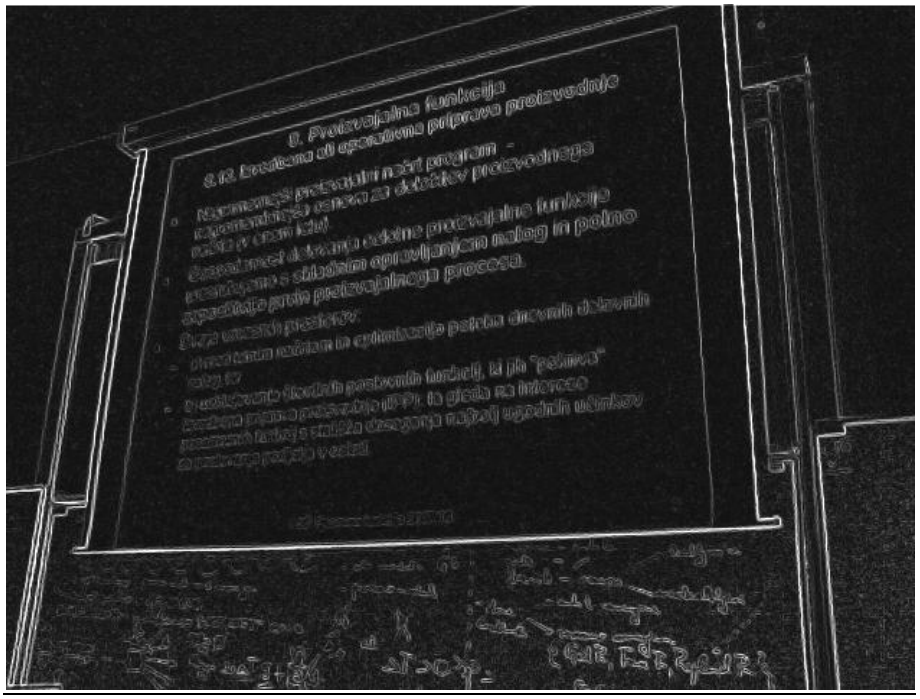
1	0
0	-1

0	1
-1	0

Slika 12: Maski Robertsovega operatorja. Na levi maska Gx, na desni maska Gy.

Operator je zelo hiter, ker mora za določitev gradienta vsake točke upoštevati samo štiri slikovne točke. Prav tako pri računanju potrebujemo samo operaciji seštevanja in odštevanja, kar še dodatno pospeši celoten proces.

Glavna slabost operatorja je v tem, da je zaradi majhne maske sposoben zaznati robove samo tam, kjer je razlika v intenzitetah slikovnih točk zelo velika. Iz istega razloga je hkrati tudi dovzeten na šum na sliki.



Slika 13: Slika robov, dobljena z Robertsovim operatorjem

3.3.1.2 Sobelov operator

Podobno kot prej omenjeni Robertsov operator, Sobelov išče robove z ugotavljanjem gradientov [10]. Uporablja dve maski, ki sta za razliko od mask Robertsovega operatorja nekoliko večji.

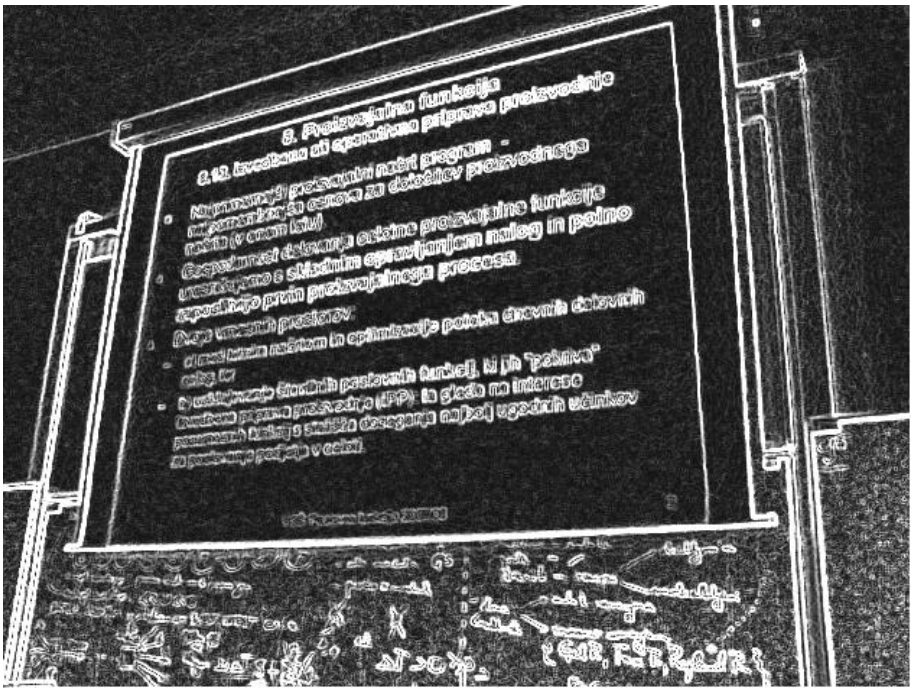
1	0	-1
2	0	-2
1	0	-1

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

Slika 14: Maski Sobelovega operatorja. Na levi maska Gx, na desni maska Gy.

Operator zaradi večjih mask bolj gladi sliko, kar ga naredi manj dovzetnega na šum. Večji maski povzročita, da operator lahko zazna tudi manjše spremembe v intenzitetah sosednjih slikovnih točk in manj izrazite robove. Intenzitete najdenih robov so nekoliko večje, zaradi večje maske pa so ti tudi nekoliko širši.

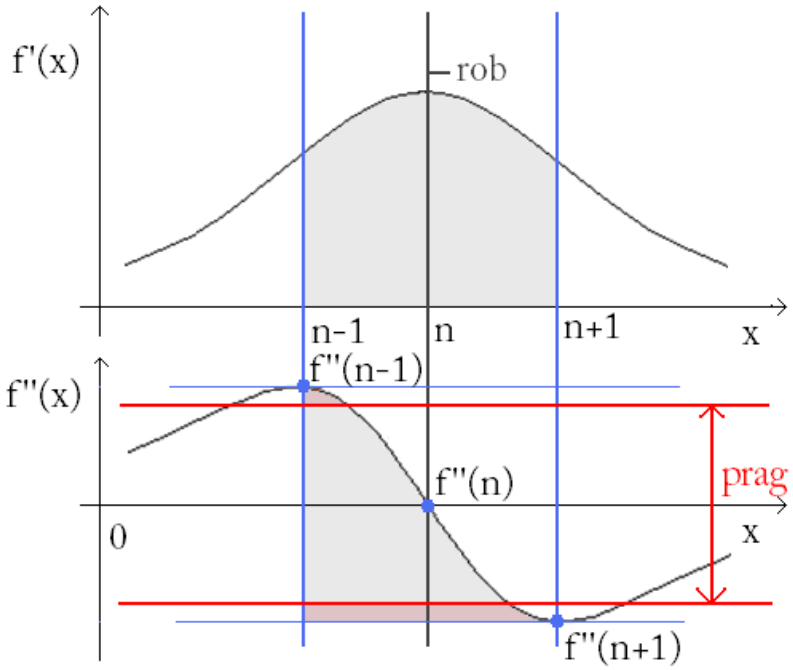
Operator je počasnejši od Robertsovega, ker mora algoritem zaradi večje maske upoštevati večje število sosednjih slikovnih točk.



Slika 15: Slika robov, dobljena s Sobelovim operatorjem

3.3.2 Iskanje robov na podlagi drugega odvoda

Drugi tip detektorjev išče robove na podlagi drugega odvoda [5]. Rob se nahaja tam, kjer drugi odvod slikovne funkcije f"(x,y) seka točko, kjer je njena vrednost enaka nič. Tam se nahaja lokalni minimum oziroma maksimum odvoda funkcije. Kot smo videli v razdelku 3.3.1, lahko ta, če se nahaja nad pragom, predstavlja rob. Podobno kot pri gradientno usmerjenih detektorjih lahko tudi tukaj s pragom omejimo število najdenih robov. Prag določa, kako hiter mora biti prehod f"(x,y) na točki, kjer je njena vrednost enaka nič. Čim večji je prag, intenzivnejši mora biti ta prehod, da jo določimo kot rob.



Slika 16: Drugi odvod intenzitet točk na robu

Približno vrednost drugega odvoda funkcije f"(x) pri vrednosti argumenta x = n (slika 16) izračunamo na naslednji način:

$$f''(n) = f'(f(n+1) - f(n)) = f'(n+1) - f'(n)$$

Nato izračunamo oba odvoda:

$$f'(n+1) = \frac{d(f(n+1))}{d(n+1)} = f(n+1) - f(n)$$

$$f'(n) = \frac{d(f(n))}{dn} = f(n) - f(n-1)$$

Dobljena odvoda vstavimo v prvo enačbo, da dobimo približek drugega odvoda:

$$f''(n) = [f(n+1) - f(n)] - [f(n) - f(n-1)] = f(n-1) - 2f(n) + f(n+1)$$

Prehod na dvodimenzionalno slikovno funkcijo f(x,y) predstavimo s parcialnima odvodoma.

Drugi odvod po x-u:

$$\frac{\partial^2(f(x,y))}{\partial \mathrm{x}^2} = f(x-1,y) - 2f(x,y) + f(x+1,y)$$

Drugi odvod po y-u:

$$\frac{\partial^2(f(x,y))}{\partial \mathrm{y}^2} = f(x,y-1) - 2f(x,y) + f(x,y+1)$$

Drugi odvod slikovne funkcije f"(x,y):

$$\nabla^2 f(x,y) = \frac{\partial^2\big(f(x,y)\big)}{\partial \mathrm{x}^2} + \frac{\partial^2(f(x,y))}{\partial \mathrm{y}^2}$$

$$\nabla^2 f(x,y) = \mathrm{h(x,y)} = f(x-1,y) + f(x+1,y) - 4f(x,y) + f(x,y-1) + f(x,y+1)$$

Druge odvode lahko na bitni sliki dobimo z apliciranjem naslednje konvolucijske maske:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Ko dobimo drugi odvod slikovne funkcije h(x,y), lahko s pomočjo praga preverimo, če se na koordianatah (x,y) nahaja rob. To storimo tako, da v okolici teh koordinat na h(x,y) poiščemo maksimum in minimum. Če ima maksimum pozitivno, minimum negativno vrednost, njuna razlika pa je večja od praga, se na koordinatah (x,y) nahaja rob.

Detektorji robov, ki temeljijo na drugem odvodu, so sposobni zaznati robove tudi tam, kjer so intenzitetni prehodi med sosednjimi točkami bolj postopni. Ker upoštevajo samo vrhove robov, so najdeni robovi tudi precej tanjši.

Primera detektorjev robov, ki temeljita na drugem odvodu, sta Laplacov in LoG (Laplacian of Gauss).

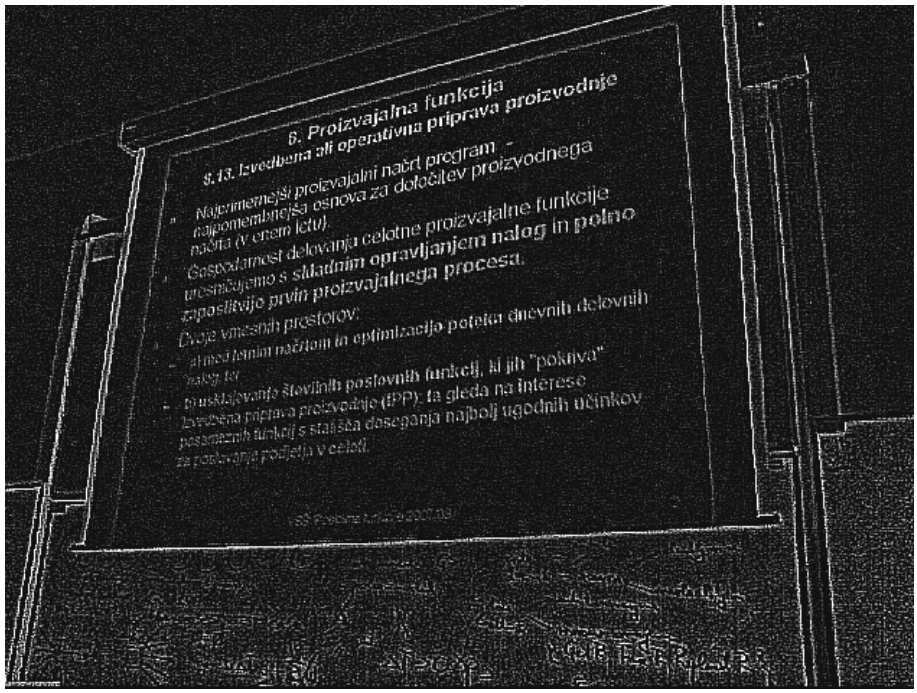
3.3.2.1 Laplacov operator

Laplacov operator je enostaven in hiter algoritem za iskanje robov, ki temelji na drugem odvodu. Za razliko od detektorjev robov, ki temeljijo na računanju gradienta, uporablja Laplace eno samo konvolucujko masko, s katero detektira robove v smereh x in y.

0	-1	0
-1	4	-1
0	-1	0

Slika 17: Konvolucijska maska za Laplacov operator

Slabost operatorja je v tem, da z njim ne dobimo podatkov o smeri robov in da je občutljiv na šum.



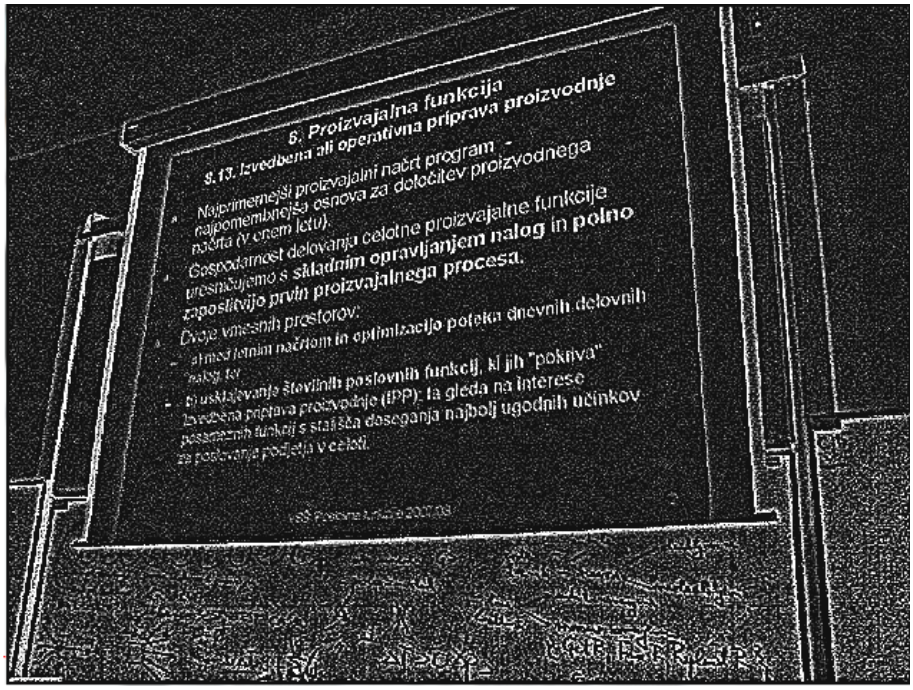
Slika 18: Slika robov, dobljena z Laplacovim operatorjem

3.3.2.2 Operator LoG (Laplacian of Gauss)

LoG operator nadgradi Laplacovega z Gaussovim filtrom za glajenje slike (glej razdelek 3.3.3.1). To povzroči, da je operator manj občutljiv na šum.

0	0	1	0	0
0	1	2	1	0
1	2	-16	2	1
0	1	2	1	0
0	0	1	0	0

Slika 19: Konvolucijska maska za LoG



Slika 20: Slika robov, dobljena z operatorjem LoG

3.3.3 Detektor robov Canny

Detektor robov Canny velja za enega najboljših in največkrat uporabljenih algoritmov za zaznavo robov. Sestavljen je iz večjega števila podalgoritmov. Uporaba teh omogoči detektorju, da daje najbolj točno sliko robov. Slabost detektorja je, zaradi uporabe velikega števila podalgoritmov, njegova velika časovna kompleksnost.

Algoritem Canny je sestavljen iz naslednjih podalgoritmov [6], ki si sledijo v naslednjem zaporedju:

- izločanja šuma na sliki z metodo glajenja,
- iskanja intenzitetnega gradienta slike,
- tanjšanja robov (Non-Maximum Supresion),
- določitve spodnjega in zgornjega praga (Hysteresis Thresholding),
- sledenja robovom.

3.3.3.1 Izločanje šuma na sliki z metodo glajenja

Prvi korak je izločanje šuma na sliki, ki bi lahko bil moteč na naslednjih stopnjah algoritma. S tem se znebimo šuma, ki velikokrat nastane pri fotografiranju in se mu težko povsem izognemo.

Šum na sliki zmanjšamo tako, da na sliko najprej apliciramo Gaussov filter za glajenje.

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Slika 21: Približek Gaussove funkcije s konvolucijsko masko

Gaussov filter z uporabo Gausove funkcije določi nivo glajenja glede na uteženo povprečje okoliških slikovnih točk. Bližnje slikovne točke prispevajo večjo vrednost od bolj oddaljenih. S tem dobimo bolj enakomerno glajeno sliko z bolj izrazitimi robovi, kot če bi s preprostim filtrom za glajenje preprosto povprečili vrednosti okoliških točk.

3.3.3.2 Iskanje intenzitetnega gradienta slike

Naslednji korak je poiskati intenzitete robov. Za to uporabimo Sobelov operator. Z njim lahko poleg intenzitete robov določimo tudi smer, v kateri ti potekajo.

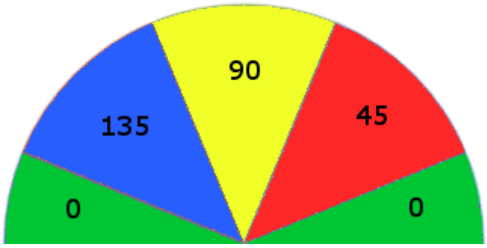
Kot θ, pod katerim poteka rob, izračunamo po formuli:

$$\theta = arcTan\left(\frac{Gy}{Gx}\right)$$

Smer, v kateri poteka rob, nato zaokrožimo na eno od štirih smeri (slika 22). Te predstavljajo vertikalo, horizontalo in obe diagonali.

Smeri robov zaokrožimo na naslednji način:

- robovi s koti med 0° in 22,5° ter 157,5° in 180° pripadajo smeri horizontale (smer 0°),
- robovi s koti med 22,5° in 67,5° pripadajo smeri prve pozitivne diagonale (smer 45°),
- robovi s koti med 67,5° in 112,5° so dodeljeni vertikalni smeri (smer 90°),
- robove s koti med 112,5° in 157,5° dodelimo drugi (negativni) diagonali (smer 135°).



Slika 22: Štiri smeri, v katere zaokrožimo potek posameznega roba

Poleg robov, dobljenih z Sobelovim operatorjem, si za vsak posamezden rob zapomnimo tudi njegovo smer. Ta podatek bomo potrebovali v naslednjem koraku za tanjšanje robov.

Robove shranimo v sliko robov, kjer intenziteta vsake točke na sliki hkrati določa tudi intenziteto roba.

Prav tako si zapomnimo tudi smeri robov, s to razliko, da vrednost vsake točke predstavlja eno izmed štirih dodeljenih smeri, v kateri poteka.

3.3.3.3 Tanjšanje robov (Non-Maximum Supresion)

Robovi, najdeni s Sobelovim operatorjem so v večini primerov zelo široki. Naloga tega koraka je stanjšati le te na širino enega slikovnega elementa.

Robove stanjšamo tako, da za vsako slikovno točko, ki predstavlja rob, preverimo, če se nahaja na njegovem vrhu. Oba sosednja slikovna elementa, pravokotna na smer, v kateri poteka rob, morata imeti v tem primeru nižjo intenziteto. Ohranimo samo robne točke, ki se nahajajo na vrhu roba.

Na tej stopnji zgradimo tudi histogram moči robov, ki predstavlja število robnih točk za vsako možno intenziteto roba. Vsakič, ko naletimo na točko, ki se nahaja na vrhu roba, v njem za eno stopnjo povečamo ustrezno polje. S pomočjo histograma v naslednjem koraku določimo spodnji in zgornji prag.

3.3.3.4 Določitev spodnjega in zgornjega praga (Hysteresis Thresholding)

Posebnost detektorja robov Canny je v tem, da za razliko od večine drugih detektorjev namesto enega uporablja dva praga. Imenujemo ju spodnji in zgornji prag. Spodnji prag mora biti vednjo manjši od zgornjega.

Praga sta lahko določena statično, vendar pa tako velikokrat pridemo do nezadovoljivih rezultatov. Če izberemo nizek prag, bomo poleg pravih robov lahko detektirali tudi točke, ki so posledica šuma in v resici ne ležijo na robu. Če pa izberemo visok prag, bomo izločili veliko pravih, vendar manj izrazitih robov.

Boljše rezultate dobimo, če oba praga dinamično določimo glede na karakteristike slike oziroma glede na razmerje intenzitet robov, ki smo jih v prejšnem koraku shranili v histogram moči robov. Za določitev zgornjega in spodnjega praga na podlagi histograma je možnih več rešitev. Ponavadi najprej določimo zgornji prag. Tega določimo tako, da v histogramu robov poiščemo neko srednjo vrednost intenzitet (mediano) in jo določimo za prag. Spodnji prag pa postavimo na vrednost dveh tretjin zgornjega. Na splošno se tako razmerje med spodnjim in zgornjim robom v praksi izkaže za najbolj uporabno.

Pri naši implementaciji določimo praga tako, da omejita skupno število najdenih robnih točk, ki se nahajajo na sliki robov. Te vedno predstavljajo samo najbolj izrazite robove. Na ta način, na bolj kompleksnih slikah, zmanjšamo število vseh robov in tako sliko poenostavimo.

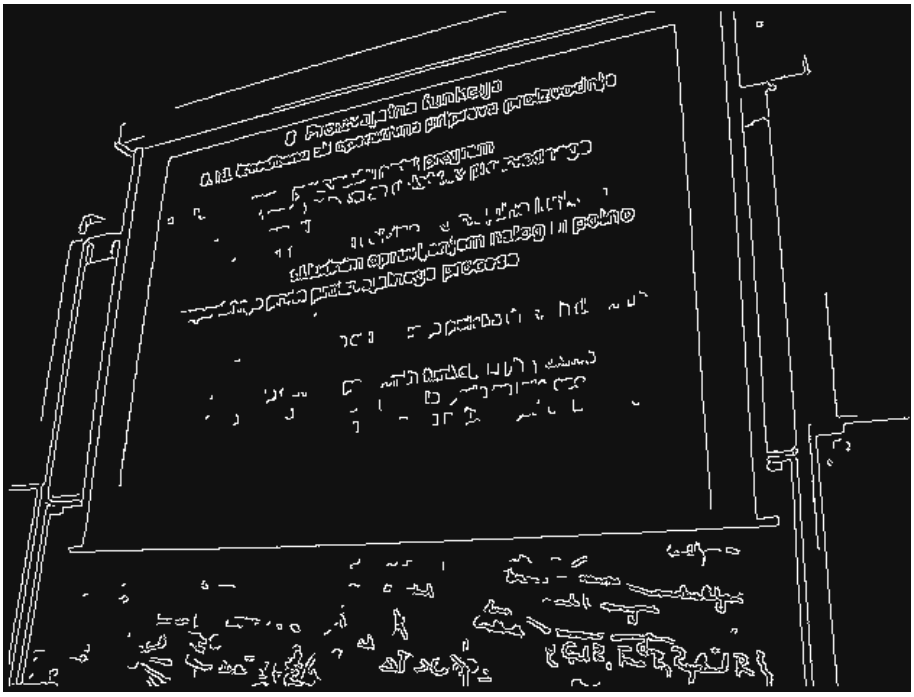
Zgornji rob določimo tako, da se pomikamo po histogramu robov od robov z večjimi intenzitetami proti tistim z manjšimi in sproti seštevamo število robnih točk. Ko vsota preseže določeno število, postavimo zgornji prag na zadnjo izbrano vrednost v histogramu.
Na tak način zagotovimo, da dobimo najbolj izrazite robove, hkrati pa omejimo tudi število vseh robnih točk. Spodnji prag postavimo na vrednost dveh tretjin zgornjega praga.

3.3.3.5 Sledenje robovom

Glede na intenziteto ločimo tri skupine robov:

- Robove z intenziteto, večjo ali enako zgornjemu robu, označimo za absolutne. Zanje smo prepričani, da na sliki resnično omejujejo različna področja.
- Robov, ki imajo inteziteto manjšo od spodnjega praga, ne upoštevamo več.
- Robove, ki imajo vrednost intezitete med spodnjim in zgornjim pragom, označimo za možne robove. To so robovi, za katere še ne moremo točno reči, da so res robovi.Te določimo tako, da potujemo skozi sliko robov. Ko naletimo na robno točko, ki ima inteziteto, večjo od zgornjega praga, se na njej ustavimo. Glede na smer, v kateri poteka najdeni rob, začnemo iskati sosednje robne točke, ki imajo intenzitete večje od spodnjega praga. Če najdemo tako robno točko, se postavimo nanjo in postopek rekurzivno ponovimo. Pri vseh najdenih robnih točkah lahko predpostavimo, da so nadaljevanje absolutnega roba in jih določimo kot nove absolutne robne točke.

```
/** Psevdo koda algoritma za sledenje robovom **/

za vsako robno točko T na sliki robov
   če intenziteta T > zgornji_rob
      če T ni določena kot absolutni rob
         določi T kot absolutni rob
         T2 = T
         * za vsako robno točko T3, ki leži ob T2
            če intenziteta T3 > spodnji_rob
               če T3 ni določena kot absolutni rob
                  določi T3 kot absolutni rob
                  T2 = T3
                  pojdi na *
```



Slika 23: Slika robov, dobljena z detektorjem robov Canny

3.4 Iskanje premic

Naslednji korak pri iskanju površin je, da s pomočjo slike robov poiščemo vse ravne črte, ki lahko predstavljajo meje iskanih površin. Predpostavimo, da so iskane površine omejene z daljicami. Če želimo na sliki najti daljice, moramo najprej poiskati premice, na katerih te ležijo. Premice najdemo z uporabo Houghovo transfomacijske metode za iskanje objektov..

3.4.1 Houghova transfomacija

Houghovo transformacijo uporabljamo za zaznavo najrazličnejših objektov na slikah. Z njo zaznamo premice, kroge, kvadrate in druge matematično opisljive oblike, ki jih lahko opišemo z enačbo. Algoritem je robusten in zmožen zaznave tudi manj izrazitih objektov.

Na začetku ustvarimo večdimenzijsko akumulacijsko polje, v katerem vsaka dimenzija predstavlja eno spremenljivko v enačbi, ki opisuje iskani objekt. Vse točke v polju imajo na začetku vrednost nič. Nato za vsako robno točko na sliki robov izračunamo vse možne vrednosti parametrov enačbe iskanega objekta. Pri tem v enačbo vnesemo njene koordinate na sliki robov. V akumulacijskem polju potem poiščemo točko, kjer se stikajo vsi parametri, in ji za eno stopnjo poveča vrednost. Vsaka točka v polju predstavlja en objekt. Z akumulacijskim poljem lahko tako opišemo vse možne predstavitve objekta v prostoru.

Objekte na sliki določimo tako, da v akumulacijskem polju poiščemo točke z največjimi vrednostmi (glasovi). Te določimo s pomočjo vnaprej določenega praga. Čim večja je vrednost točke, bolj je izrazita podoba objekta na sliki. Točke, ki imajo v polju večjo vrednost od praga, predstavljajo najdene objekte na sliki. Ker poznamo dimenzije (koordinate), v katerih se te nahajajo, poznamo hkrati tudi vse parametre v enačbah objektov in s tem tudi same objekte.

Časovna in prostorska kompleksnost Houghove transformacije je odvisna od števila najdenih robnih točk in števila parametrov, potrebnih za opis objekta. Zato le ta za detekcijo kompleksnejših oblik velikokrat ni primerna. Generiranje akumulacijskega polja z velikim številom dimenzij postane namreč časovno preveč kompleksno.

3.4.2 Houghova linearna transformacija

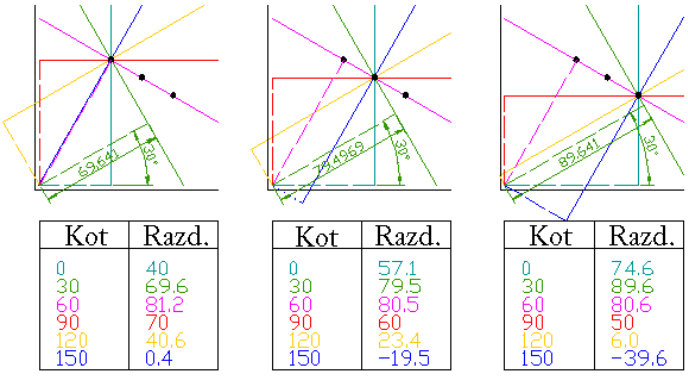
Za iskanje premic uporabimo Houghovo linearno transformacijo [11]. Z njo poiščemo premice, ki omejujejo iskane štirirobne površine. Premica je predstavljena z enačbo $y = k * x + b$ in jo v akumulacijskem polju lahko predstavimo, če poznamo parametra k in b. Taka predstavitev premice je slaba, ker ne more predstaviti navpične premice. Pri navpični premici ima namreč parameter k neskončno vrednost, kar pa ni smiselno. Zato namesto kartezične enačbe rajši uporabimo parametrično predstavitev premice. V tej obliki lahko opišemo poljubno premico.

$$r = x * \cos(\theta) + y * \sin(\theta)$$

Premico v tej obliki opisujeta dva parametra:

- parameter r predstavlja oddaljenost premice do središča koordinatnega sistema (središča slike),
- parameter θ predstavlja orientacijo premice.

Za detekcijo premic potrebujemo dvodimenzijsko akumulacijsko polje, v katerem ena dimenzija (višina) predstavlja razdaljo, druga (širina) pa naklonski kot premice. Višina polja je odvisna od velikosti slike. Čim večja je slika, večje je število slikovnih točk in s tem razdalja med središčem slike in od njega najbolj oddaljeno premico. Širina polja, ki predstavlja naklonski kot, je odvisna od tega kako natančno želimo določiti naklonski kot premice. Da pokrijemo vse možne orientacije premice, mora ta potekati od 0 do 180° s poljubno velikostjo koraka. Pri naši implementaciji je velikost koraka 1°.



Kot	Razd.
0	40
30	69,6
60	81,2
90	70
120	40,6
150	0,4

Kot	Razd.
0	57,1
30	79,5
60	80,5
90	60
120	23,4
150	-19,5

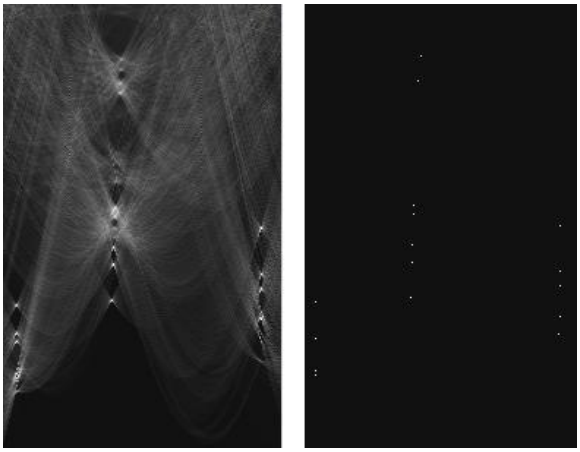
Kot	Razd.
0	74,6
30	89,6
60	80,6
90	50
120	6,0
150	-39,6

Slika 24: Primer računanja parametrov premice

Za vsako najdeno robno točko izračunamo parameter r za vse kote θ. Tako moramo izračunati parameter r za 179 premic. V akumulacijskem polju povečamo vrednosti točk, ki jih določajo koordiane (θ, r).

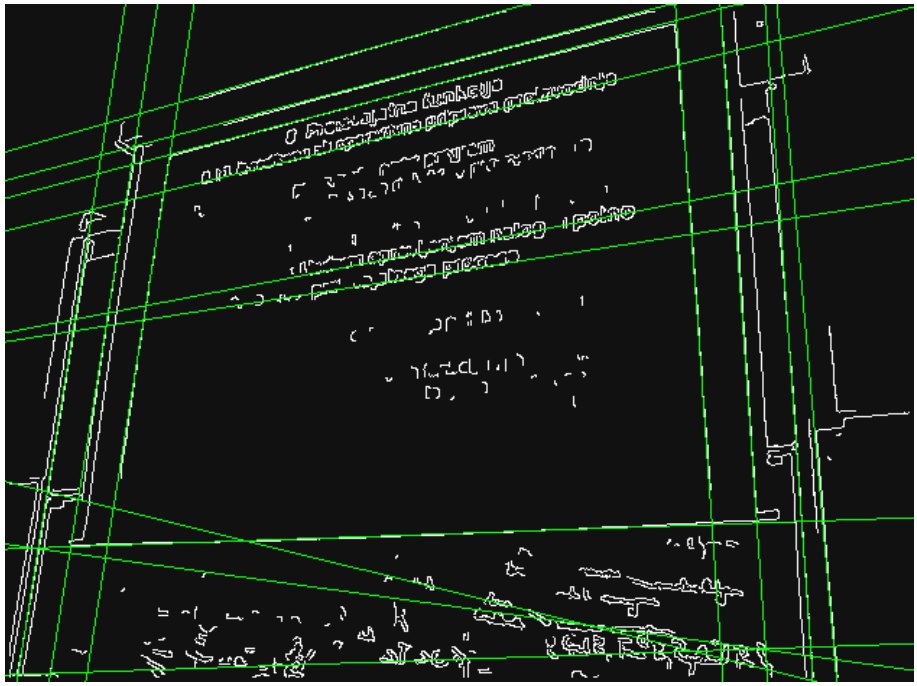
```
/** psevdo koda polnjenja akumulacijskega polja AK_POLJE **/

za vsako robno točno E na sliki robov
   za vsak naklonski kot FI od 0 do 179 z velikostjo koraka 1
      razdalja = E.x * cos(FI) + E.y * sin(FI)
      AK_POLJE[FI][razdalja]++;
```

V akumulacijskem polju izberemo najbolj izrazite premice, tako, da v akumulacijskem polju izberemo samo tiste točke (premice), ki predstavljajao lokalne maksimume oziroma vrhove. Za iskanje vrhov smo uporabili preprost algoritem za iskanje vrhov na akumulacijskem polju. Algoritem se pomika skozi vse točke v akumulacijskem polju (slika 26 levo). Na vsakem koraku primerja določeno število točk v njeni okolici. Označi samo robno točko z največjo vrednostjo, ostale pa odznači. S spreminjanjem števila točk, ki jih preverimo za vsako robno točko, določamo toleranco podobnosti premic. Čim večje je število, manj podobnih premic dobimo. Dve premici sta si podobni, če imata podobni vrednosti (r, θ).



Slika 25: Prikaz dvodiminzionalnega Houghovega akumulacijskega polja

Rezultat algoritma je akumulacijsko polje (slika 26 desno), v katerem so vidni samo intenzitetnostni vrhovi, ki predstavljajo premice. Premice dobimo tako, da preberemo njihove koordinate in s tem za vsako dobimo njen naklonski kot in razdaljo.



Slika 26: Premice, najdene s Houghovo linearno transformacijo

3.5 Iskanje daljic

Ko smo na sliki poiskali premice, določimo na njih ležeče daljice. Te lahko omejujo iskane štirirobne površine.

Daljice določimo tako, da na vsaki premici poiščemo vsa presečišča premice z drugimi premicami. Med temi presečišči ležijo daljice.

```
/** psevdo koda algoritma za iskanje daljic **/

za vsako premico P1 v seznamu premic SEZ_P
   za vsako premico P2 v SEZ_P
      če P1 != P2
         če P1.naklonski_kot != P2.naklonski_kot
            poišči stičišče S1 med P1 in P2
               če nahajanje S1 znotraj okvira slike
                  shrani S1 v seznam stičišč SEZ_S
   za vsako stičišče S1 v SEZ_S
      odstrani S1 iz SEZ_S
         za vsako stičišče S2 v SEZ_S
            kreiraj daljico D s krajiščema S1 in S2
             če dolžino D > min_dolžina
               če polnost D > min_polnost
                  shrani D v seznam daljic SEZ_D1
                  shrani S1 in S2 v polje intenzitet POL_I

   izprazni SEZ_S

za vsako daljico D v SEZ_D1
   če stik D z ostalimi daljicami v POL_I
      shrani D1 v seznam daljic SEZ_D2

vrni SEZ_D2
```

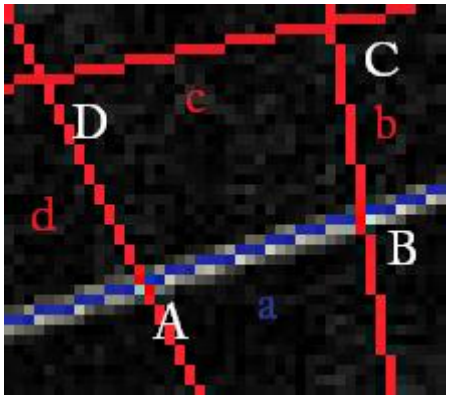
Daljica mora ustrezati trem zahtevam:

- Potekati mora skozi robne točke.
- Imeti mora primerno dolžino.
- Stikati se mora z vsaj še dvema drugima daljicama.

3.5.1 Polnost daljice

Iščemo tiste daljice, pod katerimi se, na sliki robov, v zadostni meri nahajajo robne točke.

Polnost premice preverimo tako, da se po njej premikamo z neko vnaprej določeno velikostjo koraka. Pri vsakem koraku vzamemo koordiante trenutne točke na daljici in preverimo, če se na teh koordiantah, na sliki robov, nahaja robna točka. Čim večje je skupno število najdenih robnih točk pod daljico, bolj verjetno je, da daljica predstavlja realno daljico na sliki. Polnost daljice izrazimo v odstotkih. To informacijo shranimo pri vsaki najdeni daljici.



Slika 27: Prikaz štirih potencialnih daljic, od katerih ima samo daljica AB primerno polnost

Pri uporabi detektorja robov Canny za iskanje robov uporabimo sliko robov, ki smo jo dobili kot rezultat pri iskanju intenzitetnega gradienta s Sobelovim operatorjem. Canny uporablja metodo tanjšanja robov, tako da z njim dobimo robove širine ene slikovne točke. Ti zato velikokrat ne ležijo točno pod daljico, ampak se nahajajo v njeni neposredni bližini. S Sobelovim operatorjem pa dobimo nekoliko širše robove in zato dobimo tudi pravilnejšo oceno polnosti daljice.

3.5.2 Dolžina daljice

Najdena daljica mora biti dovolj dolga, da jo lahko štejemo kot daljico, ki lahko oklepa iskano površino. Veliko najdenih stičišč premic si je namreč relativno zelo blizu skupaj. Daljice, krajše od neke minimalne vrednosti, izločimo in s tem kasneje pospešimo iskanje površin.

Dolžino daljice (d) s krajiščema $A(x_1, y_1)$ in $B(x_2, y_2)$ izračunamo z naslednjo enačbo:

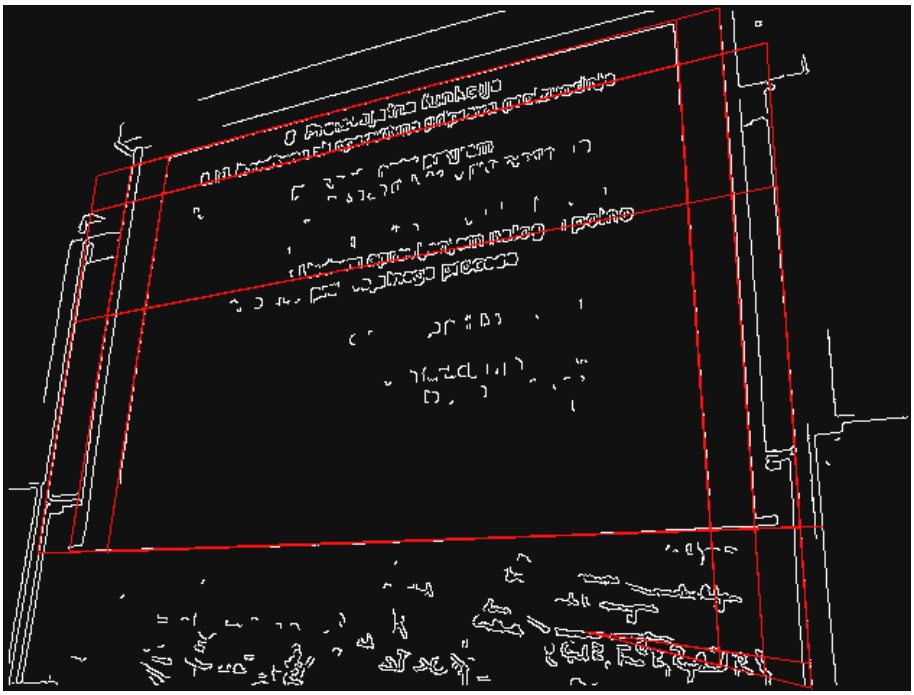
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3.5.3 Iskanje stikov med daljicami

Ta korak izvedemo na koncu, nad množico daljic, ki ustrezajo prejšnjima dvema pogojema. Ker bomo kasneje daljice uporabili za iskanje štirikotnih površin, na tej stopnji predpostavimo, da mora vsaka daljica sekati vsaj še dve drugi daljici. Tako daljico odkrijemo tako, da preverimo, če sta obe njuni stičišči skupni še kakima drugima daljicama. V nasprotnem primeru smo sicer odkrili daljico, ki pa ni del nobenega lika in za nas ni več zanimiva.

Iskanja skupnih stičišč se lotimo z izdelavo dvodimenzionalnega polja intenzitet stičišč. Vsaka točka v polju nam pove, kolikokrat je uporabljeno stičišče na koordiantah, enakih koordiantam točke. Velikost polja je enaka velikosti slike, na kateri iščemo daljice. Vse točke imajo na začetku vrednost nič. Vsakič, ko najdemo primerno daljico, v polju poiščemo koordinati njunih krajišč in jima za eno stopnjo povečamo vrednosti.

S pomočjo intenzitetnega polja stičišč na koncu odstranimo vse daljice, katerih krajišča imajo v polju vrednost, manjšo od dva.



Slika 28: Najdene daljice na sliki

3.6 Iskanje površin

Zadnji korak pri iskanju štirirobnih površin je, da med vsemi najdenimi daljicami poiščemo tiste, ki jih omejujejo.

Potek iskanja površin:

1. Poiščemo daljice, ki se stikajo v krajiščih.
2. Med pari daljic poiščemo tiste, ki se stikajo med seboj.
3. Najdene pare daljic združimo v štirikotne like.

```
/** psevdo koda algoritma za iskanje štirikotnikov **/

za vsako daljico D1 v seznamu daljic SEZ_D
   odstrani D1 iz SEZ_D
   za vsako daljico D2 v SEZ_D
      če D1.naklonski_kot != D2.naklonski_kot
         če stik D1 in D2
            shrani stik v seznam stikov daljic SEZ_ST

za vsak stik daljic ST1 v SEZ_ST
   odstrani ST1 iz SEZ_ST
   za vsak stik daljic ST2 v SSD
      če stik ST1 in ST2 v obeh stičiščih
         če ST1.naklonski_kot_1 != ST2.naklonski_kot_1
            če ST1.naklonski_kot_2 != ST2.naklonski_kot_2
               kreiraj površino P iz ST1 in ST2
                  če podobnost P z ostalimi povšinami v SEZ_P
                     če velikost P > min_velikost
                        če P predstavlja konveksen lik
                           shrani P v seznam površin SEZ_P
                  odstrani ST2 iz SEZ_SD

vrni SEZ P
```

Najdena površina mora ustrezati naslednjim zahtevam:

- Ne sme biti preveč podobna ostalim površinam.
- Velikost površine mora biti večja od nastavljene vrednosti.
- Predstavljati mora izbočen (konveksen) lik.

3.6.1 Edinstvenost površine

Najdene štirirobne površine so si velikokrat med seboj zelo podobne. Zato vsako na novo najdeno površino primerjamo z ostalimi površinami. Podobnost površine preverimo tako, da med seboj primerjamo ogljišča. Če se ogljišča nove površine nahajajo preblizu ogljiščem kake druge površine, jo izločimo.

Slika 29: Primerjava dveh štirirobnih površin

Površini si nista podobni pod naslednjim pogojem:

$$(\overrightarrow{AA'} > \text{min\_}razd\ ) \vee (\overrightarrow{BB'} > \text{min\_}razd) \vee (\overrightarrow{CC'} > \text{min\_}razd) \vee (\overrightarrow{DD'} > \text{min\_}razd\ )$$

3.6.2 Velikost površine

Razmerje med ploščino štirirobne površine in celotno površino slike mora biti večje od zahtevanega. Razmerje je v naprej določeno, uporabnik pa ga lahko po potrebi tudi prilagodi. S tem izločimo med manjše površine. Iščemo površine, ki predstavljajo predstavitvene projekcije. Te ponavadi zavzemajo večjo površino v okviru slike.

Za izračun razmerja potrebujemo ploščini slike in štirirobne površine. Ploščino slike izračunamo tako, da zmnožimo njeno širino in višino oziroma vzamemo število vseh slikovnih točk.

Ploščino poljubnega konveksnega štirikotnika, ki na sliki predstavlja štirirobno površino, pa izračunamo po Bretschneiderjevi formuli:

$$P = \frac{1}{4}\sqrt{4 * f^2 * e^2 - (b^2 + d^2 - a^2 - c^2)^2}$$

Parametri v formuli:

- P predstavlja površino štirirobne površine,
- a,b,c in d predstavljajo dolžine stranic,
- f,e predstavljata dolžini diagonal.

3.6.3 Izbočenost (konveksnost) površine

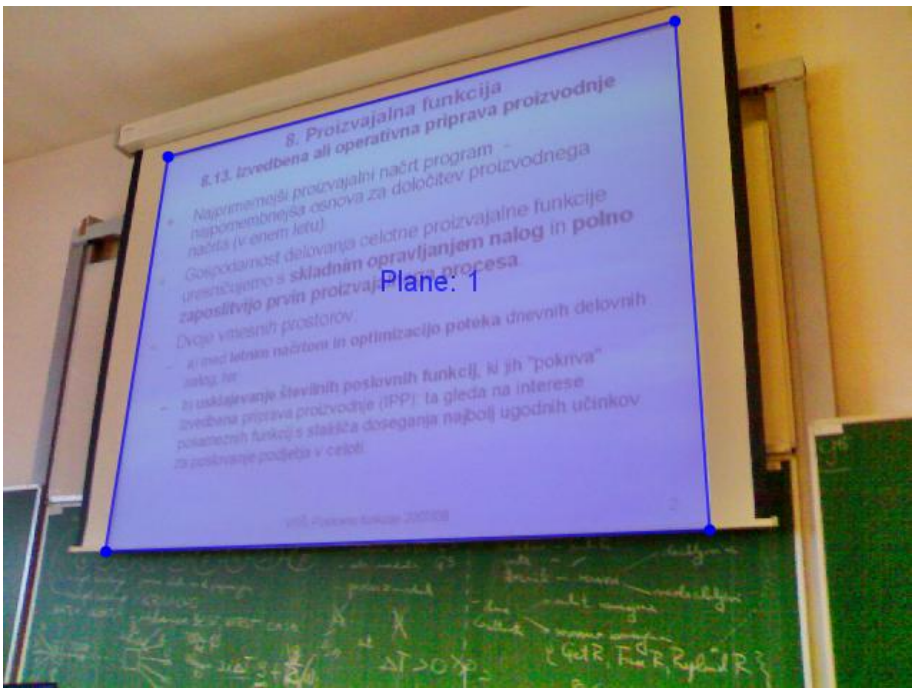
Vse najdene površine morajo predstavljati izbočene (konveksne) like. Samo taki liki lahko namreč omejujejo predstavitvene projekcije v perpektivi.

3.6.4 Izbira površine

Program ponavadi najde na sliki več površin. Ker ne ve, katera površina je za uporabnika zanimiva, mu predlaga na izbiro vse. Pri tem jih uredi glede na dodeljene ocene. Površina z največjo oceno je predlagana prva.

Površini dodelimo oceni na dva načina, glede na to, ali je uporabnik neko površino določil kot iskano ali ne:

- Če nobena površina ni določena za iskano, za oceno vzamemo vsoto polnosti daljic, ki jo oklepajo. Površina, ki bo imela največjo polnost (pod njenimi robovi se bo procentualno, na sliki robov, nahajalo največ robnih točk), bo dobila največjo oceno in bo predlagana prva. V tem primeru bo program na vsaki kot prvo predlagal najbolj izrazito površino, ne glede na obliko in na to kje na njej se nahaja.
- V primeru da uporabnik neko površino določi za iskano, za oceno uporabimo podobnost površine z iskano površino. Podobnost površin preverimo tako, da primerjamo njihova ogljišča. Čim bližje so ogljišča trenutne površine ogljiščem iskane površine, večjo oceno dobi površina. Površina, ki je najbolj podobna izbrani površini, bo predlagana kot prva. Ta način uporabimo takrat, ko vemo da se iskana površina na vseh slikah nahaja na podobnem mestu, in bi radi, da program to površino najde na vseh slikah.



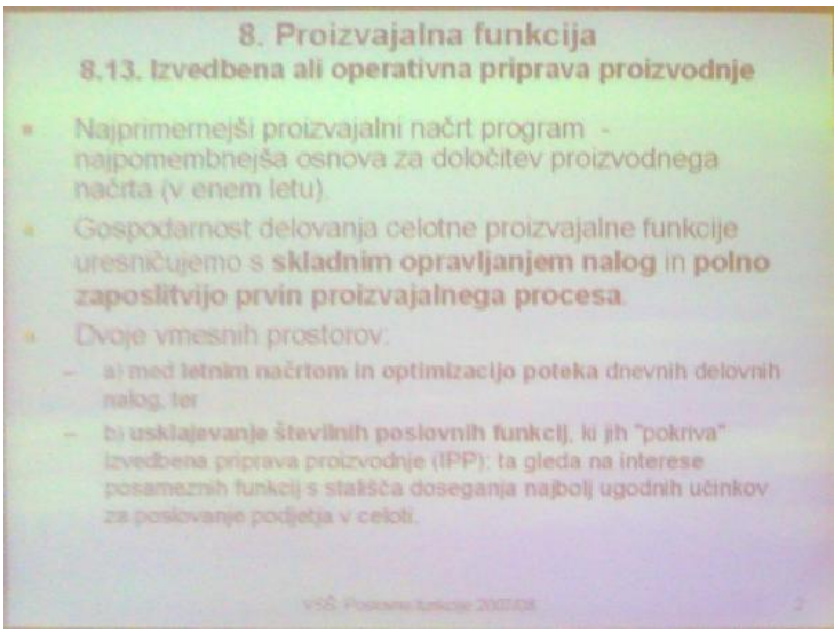
Slika 30: Primer najdene štirirobne površine na sliki

4 Korekcija perspektive površine

Naloga programa je, da popravi perspektivo pravokotnika v projekciji, tako da bo ta postavljen neposredno pred opazovalca.

Perspektiva je projekcija tridimenzionalnega prostora na dvodimenzionalno slikovno površino. Njena glavna značilnost je zmanjševanje upodobitve predmetov z oddaljenostjo od točke snemanja.

Predpostavimo, da najdene štirirobne površine na sliki predstavljajo pravokotnike like, gledane pod določenim kotom v perspektivi. Naša naloga je, da najdene štirirobne površine preslikamo v pravokotnike. Tako preslikana površina je usmerjena neposredno k opazovalcu (slika 32).



Slika 31: Projekcija s popravljeno perspektivo

Najpogostejša razmerja v uporabi za grafično predstavitev so 3:4, 9:16 in 10:16. To so rezmerja med krajšo, navpično stranico in daljšo vodoravno stranico pravokotnika, ki omejuje grafično predstavitev. Pravokotnik lahko na primer predstavlja meje zaslona ali prezentacijske projekcije.

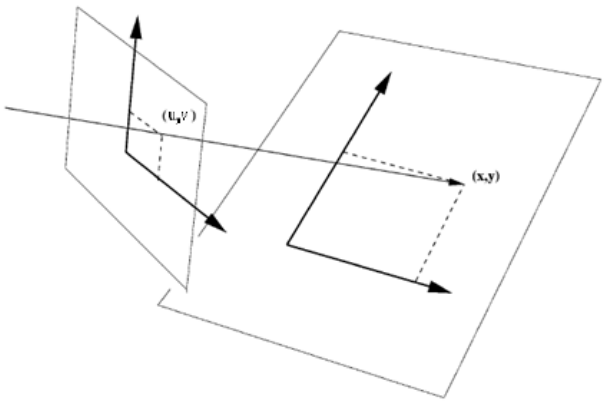
Za pravilno korekcijo perpektive moramo poznati razmeja stranic pravokotnika v perspektivi.

Poleg razmerja lahko določimo tudi velikost oziroma širino pravokotnika. Ker pravokotnik sam zavzema celotno končno sliko, ta hkrati določa tudi velikost končne slike.

Štirikotnik preslikamo v pravokotnik s pomočjo perpektivne transformacije.

4.1 Perspektivna transformacija

Perspektivna transformacija preslika poljuben štirikotnik v drugega. Pri tem ohranja ravnost črt v vseh smereh, ne ohranja pa vzporednosti med njimi. Ker poznamo koordinate vseh štirih ogljišč, na obeh štirikotnih površinah, lahko določimo transformacijo, ki opisuje preslikavo med njima. Ko to enkrat poznamo, preslikamo vsako slikovno točko, ki leži na prvi površini, na primerne koordinate na drugi.



Slika 32: Preslikava ene štiririobne površine v drugo

Preslikava za perspektivno transformacijo med izvornimi (u,v) in ciljnimi (x,y) koordinatami [2]:

$$x = \frac{a * u + b * v + c}{g * u + h * v + 1}$$

$$y = \frac{d * u + e * v + f}{g * u + h * v + 1}$$

Za rešitev sistema potrebujemo sistem osmih enačb (k=0..3):

$$x_k = u_k * a + v_k * b + c - u_k * x_k * g - v_k * x_k * h$$

$$y_k = u_k * d + v_k * e + f - u_k * y_k * g - v_k * y_k * h$$

Enačbe predstavimo matrično kot sistem velikosti 8 x 8:

$$\begin{bmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0 * x_0 & -v_0 * x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1 * x_1 & -v_1 * x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2 * x_2 & -v_2 * x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3 * x_3 & -v_3 * x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0 * y_0 & -v_0 * y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1 * y_1 & -v_1 * y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2 * y_2 & -v_2 * y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3 * y_3 & -v_3 * y_3 \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Ko v enačbe vnesemo koordinate vseh štirih ogljišč, rešimo linerani sistem s pomočjo Gaussove eliminacijske metode (glej razdelek 4.1.1).

Dobljene vrednosti osmih spremenljivk vnesemo v enačbo, s katero lahko nato vsako izvorno slikovno točko na najdeni površini preslikamo (transformiramo) na primerno mesto na ciljni sliki. Zaradi preprostejše implementacije uporabimo obratno transformacijo, kar pomeni, da za vsako slikovno točko na ciljni sliki poiščemo pripadajočo barvno vrednost na izvorni sliki in ne obratno. Tako se izognemu potrebi po interpolaciji vrednosti vmesnih slikovnih točk in dobimo enak rezultat, kot če bi sliko interpolirali po metodi najbližjega soseda (glej razdelek 2.2.1.2.1). Ker pa so s to metodo dobljene slike zelo nazobčane, barvno vrednost vsake slikovne točke na ciljni sliki dobimo z interpolacijo po metodi bilinearne interpolacije (glej razdelek 2.2.1.2.2).

```
/** psevdo koda algoritma za perspektivno transformacijo **/

izračunaj transformacijo T glede na podana ogljišča

za vsako slikovno točko ST_CILJ na ciljni sliki
   glede na T določi točko ST_IZVOR na izvorni sliki
   določi barvo B z bilinearno interpolacijo nad ST_IZVOR
   priredi ST_CILJ barvo B
```

4.1.1 Gaussova eliminacijska metoda

Gassova eliminacijaka metoda je postopek reševanja sistemov linearnih enačb. Metoda je primerna za reševanja manjših sistemov [3]. Računska kompleksnost algoritma je namreč precej velika, $O(n^3)$, kjer n predstavlja število koeficientov v enačbi. Metoda zato, za reševanje večjih sistemov, ni primerna. Ker pa imamo v našem primeru majhen sistem z osmimi enačbami, metodo uporabimo.

Gaussova eliminacijska metoda je sestavljna iz dveh delov:

- V prvem koraku vse koeficiente linearnih enačb predstavimo z matriko, rezultate enačb pa z vektorjem. Sistem nato preuredimo v zgornji trikotniški. V tej obliki imajo vsi koeficienti pod diagonalo matrike vrednosti nič. Paziti moramo tudi, da noben koeficient na diagonali nima vrednosti nič, saj bi v tem primeru kasneje prišlo do deljenja z nič, kar pa ni smiselno. Ker vsi računalniki računajo z omejeno natančnostjo, se izkaže, da ob deljenju z zelo malimi števili lahko dobimo na koncu veliko napako in netočne rezultate.Temu se izognemo z zamenjavo vrstic (angl. pivoting) v matriki tako, da bo imela ta na diagonali čim večjo vrednost.
- V drugem koraku z obratnim vstavljanjem od spodaj navzgor rešimo sistem.

Za ponazoritev vzamemo za primer preprost sistem enačb s tremi neznankami:

$$\begin{aligned} 2x + y + z &= 1 \\ x + 2y + z &= 1 \\ x + y + 2z &= 1 \end{aligned}$$

Sistem enačb zapišemo v matrični obliki:

$$\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Sistem preoblikujemo v zgornji trikotniški (prvi korak):

$$\begin{bmatrix} 1 & 1/2 & 1/2 \\ 0 & 1 & 1/3 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/3 \\ 1/4 \end{bmatrix}$$

Na koncu z obratnim vstavljanjem poiščemo vrednosti spremenljivk (drugi korak). V našem primeru imajo vse spremenljivke x, y in z vrednost ¼:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

5 Implementacija programa

Program je izdelan v programskem jeziku Java na platformi J2SE 1.5. Ta jezik izberemo zato, ker omogoča dobro prenosljivost kode med različnimi platfomami, zaradi razširjenosti je dobro dokumetiran in zanj obstaja veliko literature. Veliko primerov programske kode v strokovnih člankih in knjigah je spisanih v tem programskem jeziku, kar še pospeši razumevanje tematike.

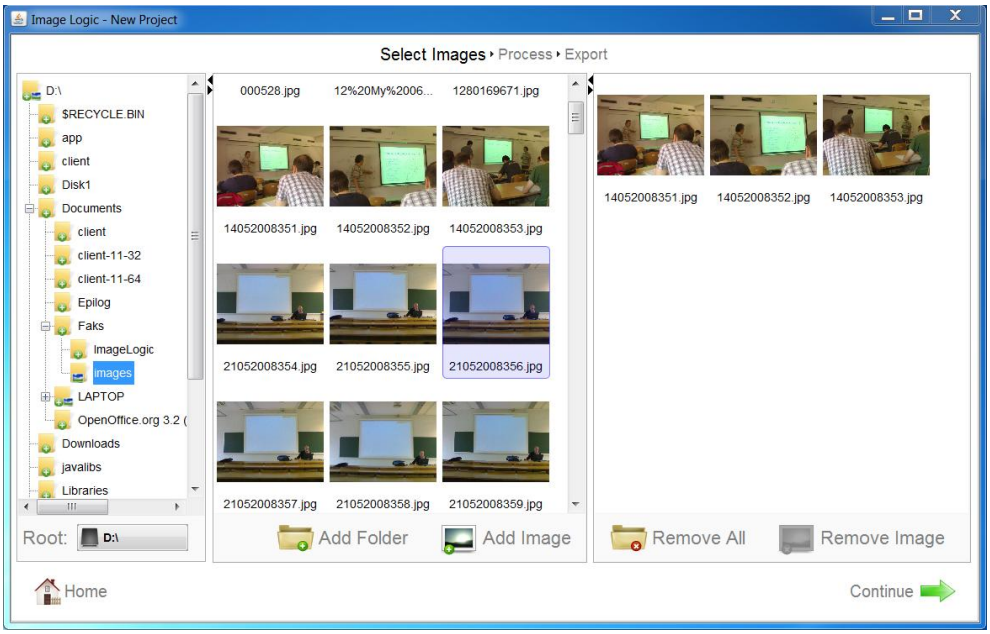
Za poenostavitev prevajanja programa in generiranja izvršljivih datotek smo uporabili programsko orodje Ant. Ant je orodje za avtomatsko prevajanje programov, podobno orodju Make, le da je implementirano s programskim jezikom Java.

Program uporablja eno zunanjo knjižnico. Knjižnica Gnujpdf omogoča enostavno branje in generiranje datotek PDF. Uporabili smo jo zaradi preprostosti in majhne velikosti.

Eden izmed ciljev pri razvoju tega programa je bil razviti čim bolj prenosljiv in kompakten program. Sprva smo za implemtacijo perspektivne transformacije uporabili metode v Java Advanced Imaging (JAI). To je vmesnik API, ki vsebuje veliko naprednejših rutin za obdelovo slik. Ker vseh v našem programu nismo potrebovali in ker je sama velikost potrebnih knjižnic precej velika, okoli 2 Mb, smo potrebne metode spisali sami in s tem znatno zmanjšali velikost programa. Ker smo uporabili eno samo knjižnico, je velikost končnega programa skupaj z njo in slikovnim materialom (ikone) 217 Kb.

6 Predstavitev programa

Uporabnik na začetku, prek preprostega vmesika, izbere zbirko slik, na katerih bo program kasneje identificiral štirirobe površine. Program uporabniku tudi pomaga, da ta čim hitreje najde želene slike. Vsako sliko mu poda v predogled. Prav tako vsak direktorij vsebuje tudi informacijo o tem, ali se v njem nahajajo slikovne datoteke oziroma direktoriji.

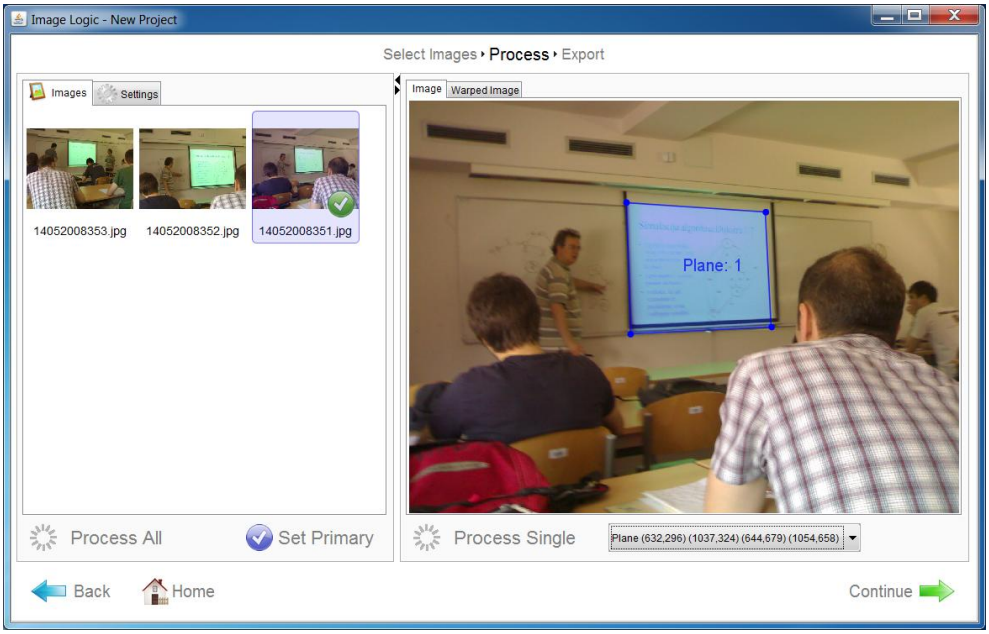


Slika 33: Meni za izbiro slik

Velikost in število vhodnih slik sta omejena samo s količino pomnilnika, ki ga dodelimo programu za izvajanje.

Program prepozna slikovne datoteke naslednjih slikovnih formatov:

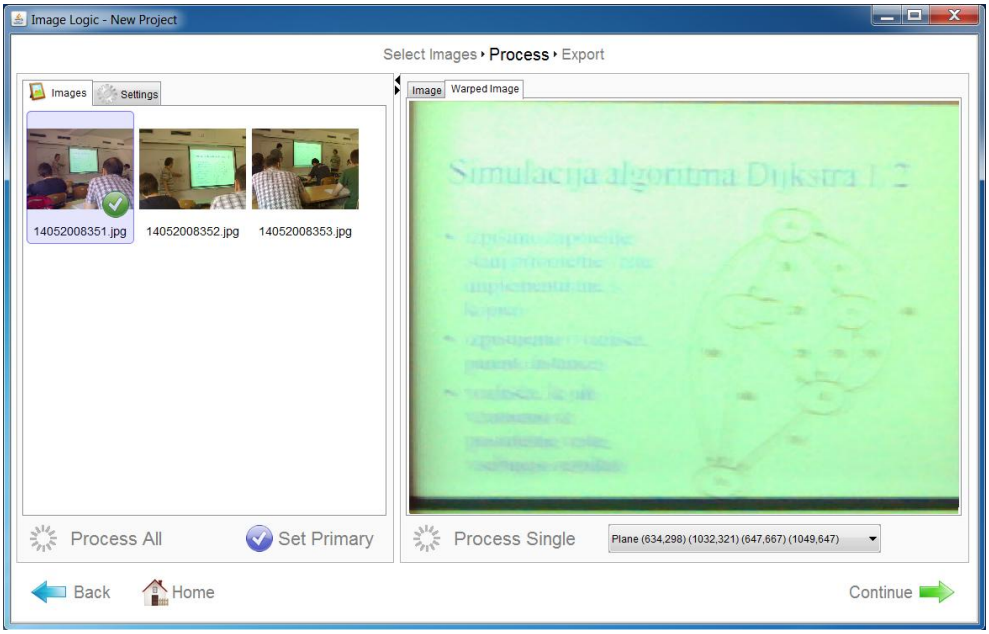
- JPEG,
- PNG,
- GIF,
- BMP.



Slika 34: Prikaz najdenih površin

V naslednjem koraku lahko uporabnik izbira med dvema načinoma iskanja površin:

- Lahko izbere eno sliko, na kateri potem program odkrije štirirobne površine.
- Lahko najde površine na celotnem seznamu izbranih slik.



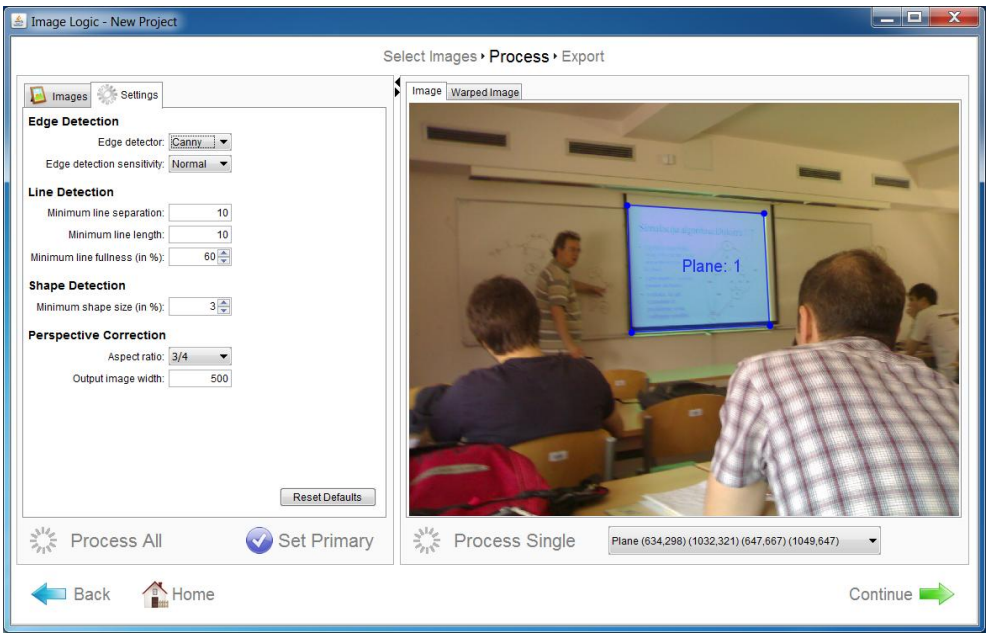
Slika 35: Predstavitev prezentacijske projekcije s popravljeno perspektivo

Nemalokrat se zgodi, da se na sliki nahaja več štirirobnih površin. Ko program konča z iskanjem površin, predlaga uporabniku vse, ki jih je zaznal. Uporabnik nato izbere tisto, ki je zanj zanimiva. Na tej stopnji lahko tudi ročno popravi meje vsake površine.

V primeru, da uporabnik z rezultatom iskanja ni zadovoljen, oziroma da program ni odkril želenih štirirobnih površin, lahko tudi prilagodi nastavitve, ki vplivajo na potek iskanja.
Uporabnik lahko izbira med petimi različnimi algoritmi za iskanje robov. Čeprav načeloma privzeti algoritem Canny velja za najboljšega, pa se včasih ostali izkažejo za bolj primerne. Ko imamo opravka z enostavnejšimi slikami, lahko z izbiro drugega preprostejšega algoritma skrajšamo čas iskanja površin.

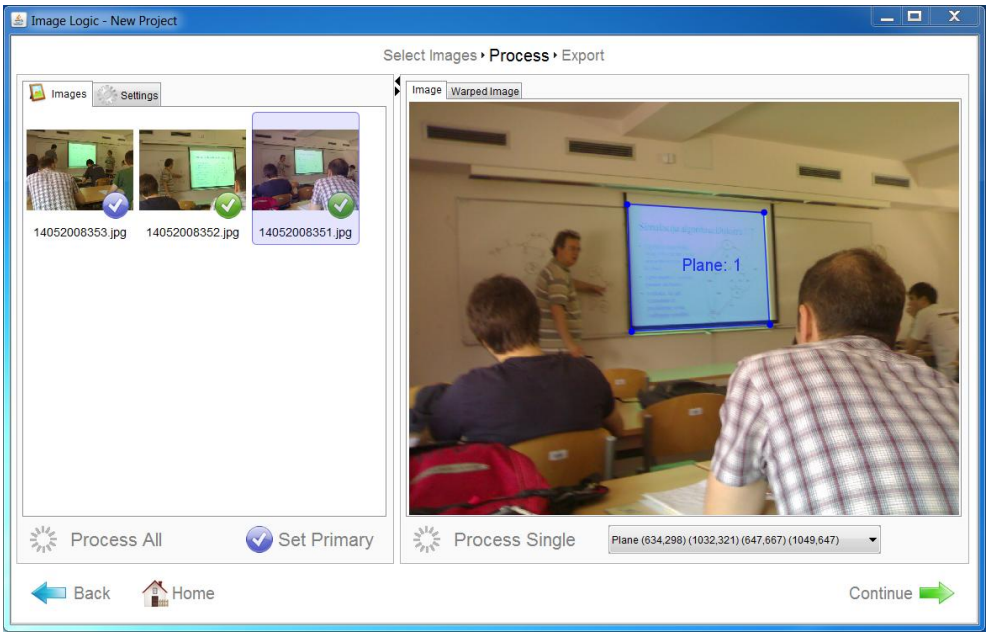
Nastavimo lahko naslednje parametre, ki vplivajo na iskanje površin:

- Izbiramo lahko med naslednjimi algoritmi za iskanje robov:
  - Canny,
  - LoG (Laplacian of Gauss),
  - Sobel,
  - Roberts,
  - Laplace.
- Če smo v prejšnjem koraku izbrali algoritem Canny, lahko določimo, koliko robov želimo dobiti na sliki. Število najdenih robov zelo vpliva na hitrost in pravilnost iskanja. Na izbiro imamo tri možnosti:
  - Malo (Low): Izberemo jo kadar imamo zelo bogate slike, na katerih je veliko ostrih prehodov med predmeti in tako veliko izrazitih robov. Z izbiro te vrednosti bomo dobili samo najbolj izrazite robove in s tem le bolj izrazite površine.
  - Normalno (Normal): Ta vrednost je izbrana kot privzeta in najde optimalno število robov.
  - Veliko (High): Vrednost izberemo ob enostavnih ali zamegljenih slikah, ki nimajo izrazitih robov. S tem povečamo število najdenih robov in hkrati tudi možnost, da bo program odkril manj izrazite površine.
- Določimo lahko, kako podobne so si lahko daljice, ki omejujejo površino. Z izbiro večjih vrednosti pospešimo iskanje, vendar pa hkrati izgubimo zmožnost zaznave podobnih površin.
- Izberemo lahko najmanjšo velikost daljice, ki oklepa površino. Z izbiro večjih vrednosti program ne bo zaznal manjših površin.
- Lahko določimo tudi, v kakšni meri so daljice lahko prekinjene oziroma kako polne morajo biti, da se upoštevajo. Ta nastavitev je pomembna takrat, kadar se pred površino nahaja predmet, ki jo delno zakriva. Z določitvijo nižjih vrednosti lahko zaznamo tudi delno zakrite površine.
- Določimo lahko minimalno površino štirirobnih objektov. Ta je predstavljena z razmerjem med njeno velikostjo in velikostjo celotne slike.
- Za pravilno korekturo perspektive površine moramo nastaviti tudi razmerje njenih stranic.
- Nastavimo lahko tudi širino končnih slik s površinami s popravljenimi perspektivami. Višina se pri vsaki sliki nato preračuna glede na izbrano razmerje stranic.



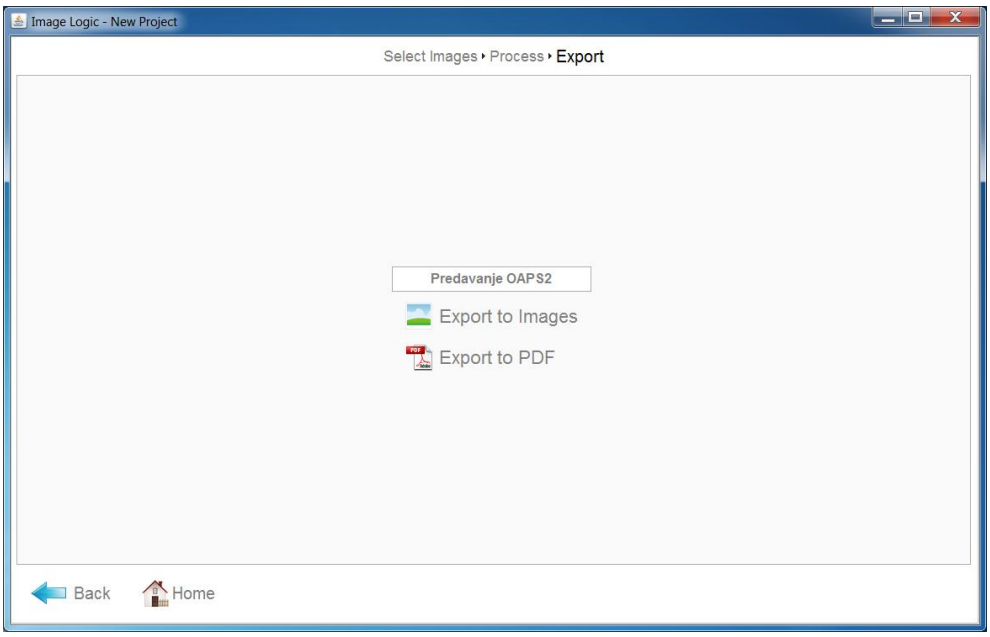
Slika 36: Nastavitev parametrov in toleranc

Ko fotografiramo predstavitev, v večini primerov stojimo na istem mestu. Projekcija je zato na vseh zajetih slikah na približno enaki lokaciji in je opazovana pod podobnim kotom. Kot in lokacija projekcij se od slike do slike minimalno spreminjata. Če želimo, da program na vseh slikah najde isto projekcijo, jo moramo na eni sliki označiti.



Slika 37: Iskanje izbrane površine na seznamu slik

Ko na sliki določimo površino, ki predstavlja projekcijo, jo lahko označimo kot primarno (Set Primary). S tem programu povemo, katero površino na seznamu slik iščemo. Ta bo nato na vsaki naslednji sliki poskušal odkriti njej čim bolj podobno površino. Tako dosežemo, da bodo na vseh slikah kot prve predlagane iste površine. Hkrati s tem tudi skrajšamo čas iskanja površin, ker se program omeji samo na izbrano območje in ne analizira celotne slike.



Slika 38: Meni za izvoz slik v slikovne datoteke ali v zapis PDF

Zadnji korak je izvoz slik s popravljenimi perspektivami štirirobnih objektov.

Končne slike lahko izvozimo na dva načina:

- Lahko izberemo direktorij, v katerega nato shranimo vse slike.
- Kreiramo datoteko PDF, v katero vstavimo vse slike.

7 Zaključek

V diplomskem delu smo z uporabo znanih metod za iskanje objektov na slikah predstavili postopek iskanja poljubnih štirirobnih objektov. Osredotočili smo se na predstavitvene projekcije, ki se v današnjem času vse bolj uporabljajo za predstavitev informacij širšemu krogu ljudi.

Rezultat diplomskega dela je program, ki z uporabo splošno uveljavljenih metod iz računalniškega vida, računalniške grafike in z nekaj na lastno iniciativo razvitimi metodami, prikaže rešitev samodejnega iskanja in korekcije perspektive predstavitvenih projekcij na podanem seznamu slik.

Program se je med testiranjem izkazal za najbolj uspešnega takrat, ko so bili iskani štirirobni objekti primarni objekti na sliki in se na sliki ni nahajalo drugih štirirobnih objektov. V primeru, da se teh na sliki nahaja več, program primerja njihove robove in porabniku predlaga tiste z bolj izrazitimi.

Velik iziv je predstavljalo iskanje robov, na podlagi katerih lahko poiščemo premice in štrikotnike, ki jih te omejujejo. Uporabili smo pet različnih detektorjev robov: Roberts, Sobel, Laplace, LoG in Canny. Prvi štirje so se izkazali za preveč enostavne in na kompleksnejših fotografijah niso dajali zadovoljivih rezultatov. Med vsemi detektorji se je za najboljšega izkazal detekor Canny. Ta je edini, ki na fotografijah vrača zadovoljivo sliko robov. Se pa močno razlikuje od ostalih v večji kompleksnosti in daljšem času, ki ga potrebuje za detekcijo robov.

Za iskanje premic smo uporabili Houghovo linearno transformacijo. Ker je hitrost iskanja premic s to metodo odvisna od števila najdenih robnih točk, smo vsako sliko na začetku nekoliko zmanjšali in določili največje število robnih točk, ki jih lahko dobimo z detektorjem robov.

Daljice, ki omejujejo štirirobni objekt, najdemo z lastno razvito metodo. Ta omogoča kasnejšo zaznavo delno zakritih štirirobnih objektov. To je zlasti pomembno, ko predavatelj stopi pred projekcijo, jo delno zakrije in s tem tudi daljice, ki jo omejujejo.

Čas iskanja objektov bi lahko skrajšali z uporabo hitrejšega programskega jezika ali z uporabo paralelnega procesiranjem. V zadnjem času so na področju računalniškega vida, pričeli uporabljati računsko moč grafičnih kartic. Te so danes izjemno hitre, kar dosežejo z velikim številom grafičnih procesnih enot. Vzporedna uporaba teh bi znatno pospešila nekatere uporabljene postopke, kot sta na primer konvolucija in transformacija.

Seznam slik

Literatura

[1] J. Corrigan, Računalniška grafika, 1995.

[2] (2010) P. Heckbert, »Projective Mappings for Image Warping« v Fundamentals of Texture Mapping and Image Warping, Master's thesis, UCB/CSD 89/516, CS Division, U.C. Berkeley, June 1989. Dostopno na: http://www.cs.cmu.edu/~ph/texfund/texfund.pdf

[3] B. Orel, Osnove numerične matematike, 3. izdaja, 2004.

[4] (2010) F. Solina, Računalniški vid nekdaj in danes, 2006. Dostopno na: http://eprints.fri.uni-lj.si/199/1/Solina_ROSUS2006.pdf

[5] (2010) Edge detection. Dostopno na: http://www.cs.cmu.edu/afs/cs/academic/class/15385-s06/lectures/ppts/lec-7.ppt

[6] (2010) Canny Edge Detection Tutorial. Dostopno na: http://www.pages.drexel.edu/~weg22/can_tut.html

[7] (2010) Bilienar interpolation. Dostopno na: http://en.wikipedia.org/wiki/Bilinear_interpolation

[8] (2010) Understanding Digital Image Interpolation. Dostopno na: http://www.cambridgeincolour.com/tutorials/image-interpolation.htm

[9] (2010) Roberts edge detection. Dostopno na: http://homepages.inf.ed.ac.uk/rbf/HIPR2/roberts.htm

[10] (2010) Sobel edge detection. Dostopno na: http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

[11] (2010) Hough transform. Dostopno na: http://en.wikipedia.org/wiki/Hough_transform