UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dejan Vidmar

ZLAGANJE SESTAVLJANK S POMOČJO RAČUNALNIKA

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: viš. pred. dr. Borut Batagelj

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

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

Št. naloge: 00541/2011
Datum: 07.03.2011

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

Kandidat: **DEJAN VIDMAR**

Naslov: **ZLAGANJE SESTAVLJANK S POMOČJO RAČUNALNIKA**
SOLVING JIGSAW PUZZLES BY COMPUTER

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Kandidat naj v svojem diplomskem delu predstavi problem reševanja sestavljenke s pomočjo računalnika. Pregleda naj področje in na podlagi tega izdela svojo rešitev v obliki aplikacije, ki bo pomagala pri sestavljanju. Program naj temelji na metodah računalniškega vida. Na koncu naj naredi tudi preizkus na konkretnih sestavljankah in poda rezultate uspešnosti.

Mentor:

viš. pred. dr. Borut Batagelj

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Dejan Vidmar,

z vpisno številko 63030015,

sem avtor diplomskega dela z naslovom:

Zlaganje sestavljank s pomočjo računalnika

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  viš. pred. dr. Boruta Batagelja
- 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 15.06.2011 Podpis avtorja/-ice:

Zahvala

Zahvalil bi se svojemu mentorju, viš. pred. dr. Borutu Batagelju, ki je zaslužen za idejo diplomske naloge. Prav tako se mu zahvaljujem za njegovo pomoč in nasvete pri izdelavi te diplomske naloge.

Kazalo

Povzetek

V diplomskem delu je predstavljen algoritem za računalniško zlaganje sestavljank s pomočjo digitalne kamere. Aplikacija preko spletne kamere zajame sliko nezloženih kosov sestavljanke, kot rezultat pa dobimo postopek sestavljanja in sliko zložene sestavljanke. V prvem delu je predstavljeno izločanje posameznih kosov, ki tvorijo sestavljanko iz vhodne slike in določanje tipov stranic in vogalnih točk vsakemu kosu. Dobljene lastnosti kosov uporabimo v drugem delu, kjer je prikazano primerjanje kosov med seboj. Kose primerjamo po obliki kot tudi po barvnih informacijah. Ker so primerjave časovno zahtevno opravilo, smo uvedli model kosov, s katerim število primerjav in s tem čas izvajanja močno zmanjšamo. V zadnjem delu je prikazano združevanje kosov v izhodno sliko, kjer s pomočjo rezultatov primerjanj določimo povezave med kosi in jih enega za drugim zložimo skupaj. Aplikacijo smo preizkusili na več sestavljankah z različnimi števili kosov, pri različnih resolucijah vhodne slike.

Ključne besede:

sestavljanke, segmentacija, iskanje robov, primerjanje oblik

Abstract

This thesis presents an algorithm for automatic solving of jigsaw puzzles with the help of digital camera. Application uses web camera to capture a picture of unassembled pieces of the puzzle and as a result we get a procedure of solving and a picture of assembled jigsaw. The first part presents extraction of individual pieces that make up the puzzle and determining the types of all four sides and calculating the corner points for each piece. The characteristics of pieces are used in the second part, where comparison of pieces to each other is shown. Pieces are compared by shape as well as by color informations. Since comparisons are time demanding tasks, we introduce a jigsaw piece model, which helps to greatly reduce number of necessary comparisons and thus reduces execution time. In the last part, we show the process of merging pieces into an output image where we, with the results of comparisons, determine the connections between the pieces and assemble them one by one. Application was tested on several puzzles with different numbers of pieces at different resolutions of input image.

Key words:

jigsaw puzzles, segmentation, edge finding, comparing shapes

Poglavje 1

Uvod

Na temo avtomatskega zlaganja sestavljank je bilo napisanih že precej del, vendar mnoga pri svojem delovanju uporabljajo rešitve sorodnih problemov. Primeri so rekonstrukcije arheoloških najdb [1, 2, 3] in združevanje površinskih delov skeniranih objektov [4]. Vendar pa je verjetno največja privlačnost zlaganja sestavljank to, da je sestavljanje naraven in zahteven izziv, ki ujame človeško domišljijo.

Tipičen problem zlaganja ima dve glavni težavi. Prva je kombinacijska: obstaja zelo veliko načinov, na katere je mogoče kose sestavljanke postaviti v mrežo polj, ki tvorijo sestavljanko. Druga je geometrijska: težko je ugotoviti, če se par komplementarnih kosov res ujema. Kadar zlagamo ročno, lahko oseba običajno po občutku ve, da se dva kosa res ujemata ko ju zloži skupaj. Oblike kosov, zajetih iz slike (naše kot tudi prejšnjih raziskovalcev) pa niso dovolj natančne za takšno določanje. Očitno lahko tudi robot čuti takšno ujemanje, Burdea in Wolfson [5] sta uporabila robotsko roko, katere senzorji lahko zaznajo dobro prileganje dveh kosov pri strojnem zlaganju sestavljank.

Tipična sestavljanka iz otroške trgovine spoštuje določena pravila, ki naredijo težavo malce bolj obvladljivo kot bi bila sicer. Standardna pravila vključujejo:

- sestavljanka ima pravokoten zunanji rob;
- kosi oblikujejo mrežo kvadratnih polj tako, da ima vsak notranji kos natanko štiri sosede (levo, desno, zgoraj in spodaj);
- kosi se “zaklenejo”s sosednjimi z jezički, ki jih sestavljajo ”vbokline”na enem kosu in ”izbokline”na sosednjem.

Naš algoritem sledi enakemu splošnemu pristopu, kot ga je predlagal Wolfson [6], to je najprej reševanje okvirja sestavljanke, nato pa zapolnjevanje

notranjosti, vendar se razlikuje v več podkorakih. V diplomi predstavljena metoda, uporablja informacije o obliki kosov, kot tudi informacije barv ob stranicah kosov. Z uvedbo modela kosov te razdelimo v 18 kategorij, glede na njihovo obliko. To precej skrajša čas, potreben za računanje. Naš algoritem za reševanje sestavljanke lahko opišemo v petih korakih (slika 1.1):

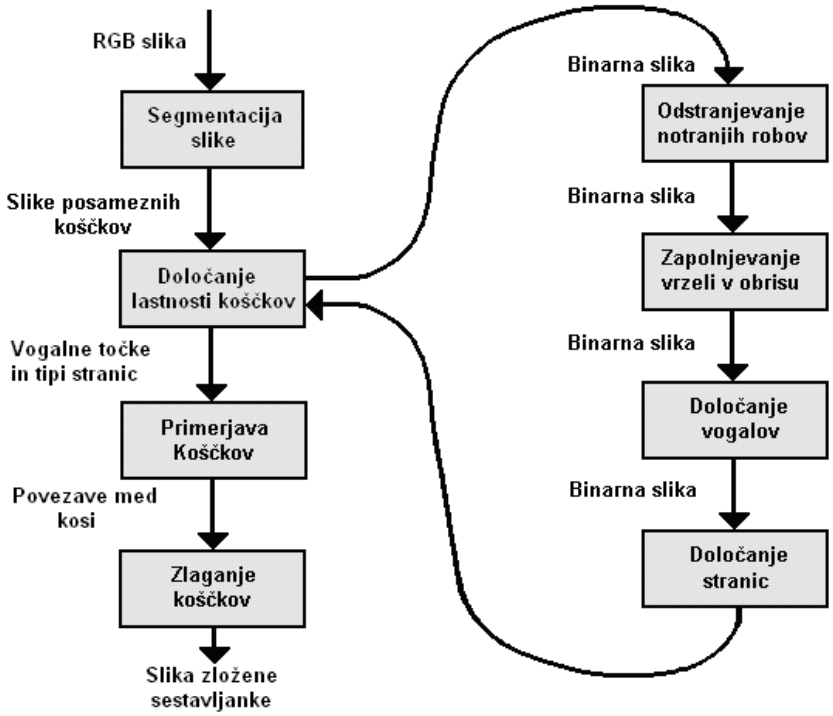
1. Izločitev posameznih kosov iz barvne vhodne slike.

2. Določanje lastnosti kosov (postopek poteka v štirih korakih: odstranjevanje notranjih robov kosa, zapolnjevanje vrzeli v obrisu, določanje vogalov in določanje stranic).

3. Primerjava kosov med trenutnim in vsemi kandidati (poteka v dveh korakih: primerjava oblike kosa in primerjava barvnih informacij).

4. Glede na rezultate iz 3. koraka določimo povezave med kosi.

5. Združevanje kosov na izhodno sliko.



Slika 1.1: Koraki algoritma.

Ker je število oblik kosov sestavljank zelo veliko, se v tej nalogi omejimo na običajne sestavljanke, katerih kosi imajo štiri robove. Sistem se lahko razširi na sestavljanke z drugačnimi oblikami kosov, saj lahko za njihovo prepoznavanje uporabimo podoben algoritem. Sistem ima trenutno nekaj omejitev:

- prepozna samo običajne sestavljanke;
- zlagamo lahko samo sestavljanke z vsemi kosi;
- kosi ne smejo biti preveč skupaj;
- ozadje na vhodni sliki mora biti kar najbolj kontrastno glede na sestavljanko;
- z večanjem števila kosov, čas potreben za preračunavanje, hitro narašča.

1.1 Sorodna dela

Radack in Badler [5] sta prvotno povzela problem dvodimenzionalne sestavljanke kot nabor enostavno povezanih ravninskih regij (silhuete kosov sestavljanke), vrtenje kosov tako, da se prilegajo skupaj v eno regijo, brez večjih vrzeli ali področij prekrivanj. Vendar ta povzetek ne omenja tekstur in barvnih informacij. Tukaj lahko problem zlaganja sestavljank opredelimo kot: S_P={P_0, P_1, ..., P_{N-1}}, kjer P_i predstavlja i-ti kos (i ∈ {0, 1, ..., N-1}) in ima lastnost zaprte meje ter določeno teksturo; in za P_i obstaja tak P_j (i≠j, i,j ∈ {0, 1, ..., N-1}), da je robni del P_i popolnoma enak robnemu delu P_j, teksture in barve v teh dveh robnih predelih pa so si čimbolj podobne. Torej P_i in P_j sta soseda (lahko ju povežemo), nato poiščemo soseda za P_k (k ∈ {0, 1, ..., N-1} in vse kose povežemo v S_p v en velik kos.

Reševanje sestavljanke z računalnikom vključuje številne naloge povezane z različnimi področji računalniškega vida: detekcija oblik, ujemanje delnih mej, prepoznavanje vzorcev, izločanje značilk in hevristično ujemanje. Zato je veliko raziskovalcev sprejelo izziv reševanja tega problema. Freeman in Garder [12] sta se prva dotaknila problema leta 1964 in njuno delo ostaja temeljnega pomena na tem področju. Zaradi omejitev v računalniškem jeziku in napravah za slikanje sta uspela rešiti problem samo z uporabo informacij v obliki kosov. Od takrat je bilo predlaganih več algoritmov. Radack in Badler [5] sta predlagala algoritem sposoben učinkovitega določanja ujemanj mej dvodimenzionalnih kosov sestavljanke. Ta postopek uporablja polarni koordinatni sistem, ki se nahaja v maksimumih in minimumih krivulj. Je metoda

za splošno predstavitev krivulj in njihovih ujemanj. Webster [13], predlaga "isthmus" kritične točke za reševanje sestavljank. Njegova rešitev se ukvarja z običajnimi sestavljankami, za primerjanje kosov pa uporablja ožine na konkavnih ali konveksnih predelih krivulje kosa. Kosiba [14], predstavi nov nabor funkcij na podlagi oblike in barvnih značilnosti običajnih sestavljank. Delo Yaoa [15], je pripravljalno delo, ki poskuša rešiti sestavljanko z računalnikom. Opozoril je na težave, ki obstajajo v delu Freemana in Garderja, in predlagal uporabo fine segmentacije robne krivulje običajnih sestavljank. Vsa dela je mogoče razvrstiti po vrstah informacij, uporabljenih pri reševanju problema in po tem, ali so cilj navadne sestavljanke s štirimi stranicami. Povzetki so prikazani v tabeli 1.1 Delo Freemana in Garderja ter delo Radacka in Badlerja se ne ukvarja z običajnimi sestavljankami. Vendar pa vse metode uporabljajo informacije o obliki kosov. Metode Kosibe in Chunga [16] oboje uporabljajo informacije o barvi. Tako kot Kosiba je tudi Chung predlagal reševanje tega problema z uporabo oblike in barvnih informacij. Ujemanje oblike določijo z razdaljo točk na robni krivulji, od premice, ki jo določata sosednji vogalni točki.

Metodo predstavil	Rešuje navadne sestavljanke	Uporablja informacije o obliki kosa	Uporablja informacije o teksturah ali barvi
Freeman in Garder (1964)	Ne	Da	Ne
Radack in Badler (1982)	Ne	Da	Ne
Webster (1991)	Da	Da	Ne
Kosiba (1994)	Da	Da	Da
Yao (1997)	Da	Da	Ne
Chung (1998)	Da	Da	Da

Tabela 1.1: Kategorizacija metod za reševanje sestavljank.

Obstajajo tudi novejši poskusi rešitve problema z uporabo nevronskih mrež (npr. Suganthan, 1999), vendar so rezultati manj kot zadovoljivi. Kennedy [18] v svojem diplomskem delu predstavi rešitev z verigo točk, kjer robne točke poveže v verigo in za vsako izračuna notranji kot glede na naslednjo točko. Nato verigo primerja z verigo drugega kosa in išče odseke ujemanj. Kosi z najdaljšimi odseki ujemanj so predlagani kot kandidati za sosednji kos. Cho, Avidan in Freeman [19] predstavijo algoritem, ki rešuje sestavljanke s kvadra-

tnimi kosi. Vsi kosi so enake oblike, zato za zlaganje uporabijo samo barvne informacije. Njihov program analizira barve kosov in jih razporedi v skupine nato pa jih s pomočjo baze slik postavi na najbolj verjetno pozicijo (modri kosi so verjetno nebo zato so na vrhu, zelena vegetacija v ospredju, ...). Ko so kosi postavljeni na njihove približne pozicije, program primerja barve točk ob robovih kosov in izračuna kateri sosednji kosi se najbolj ujemajo. Na koncu postavi nekaj fiksnih kosov in glede na njih sestavi končno sliko. Orientacije vseh kosov so pravilne, kar močno zmanjša število potrebnih primerjav, zato pa lahko njihov algoritem reši sestavljanko s precej velikim številom kosov (več kot 400).

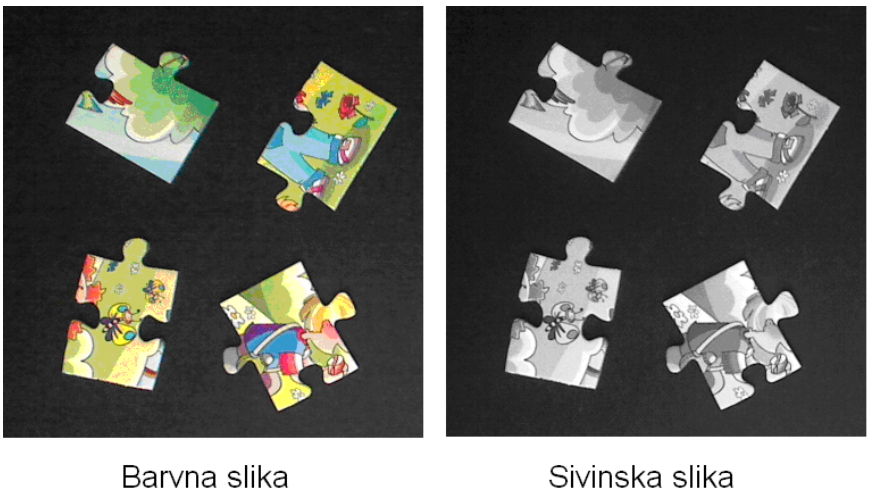
Obstajata dve strategiji pri reševanju tega problema. Prva strategija je zelo podobna načinu s katerim je sestavljenka narejena. Ponavadi za izdelavo sestavljanke razrežemo večjo sliko v več manjših neenakih kosov, z uporabo pravil rezanja. Glassner [17], je uporabil ta pristop. Vzel je sliko celotne sestavljanke in nato narisal posamezne kose v Adobe Photoshopu. Potem je program kose razporedil, vendar ohranil njihovo pravilno orientacijo, nato pa jih je s pomočjo programa poskušal sestaviti nazaj. To ni realna sestavljanka, zato njegova metoda ni vključena v tabelo 1.1. Druga strategija pa poizkuša poiskati sosednji kos danemu kosu z uporabo informacij oblike in tekstur. Dela v tabeli 1.1 uporabljajo tak pristop. Običajno pravila rezanja sestavljalcu niso podana in ker je druga strategija podobna procesu, ki ga ljudje uporabljamo za zlaganje sestavljank, smo tudi mi za reševanje uporabili ta način.

Poglavje 2

Teoretično ozadje

2.1 Pretvorba v sivinsko sliko

Sivinska slika (ang. Grayscale image) predstavlja digitalno sliko, kjer vsaka točka oz. piksel (ang. pixel), nosi le informacijo o intenziteti [7]. Slike te vrste, znane tudi kot črno-bele, so sestavljene izključno iz odtenkov sive, kjer je najmanj intenziven odtenek črne, najbolj intenziven odtenek pa bele barve (slika 2.1). Pretvorba barvne v sivinsko sliko poteka tako, da v vsaki točki dobimo vrednosti rdeče, zelene in modre komponente, nato pa seštejemo 30% vrednosti rdeče, 59% vrednosti zelene in 11% vrednosti modre barve (uteži so lahko tudi drugačne, izbrali pa smo tipične).



Slika 2.1: Barvna in sivinska slika.

Intenziteta

$$I(t) = w_r * red(t) + w_g * green(t) + w_b * blue(t) \tag{2.1}$$

kjer so

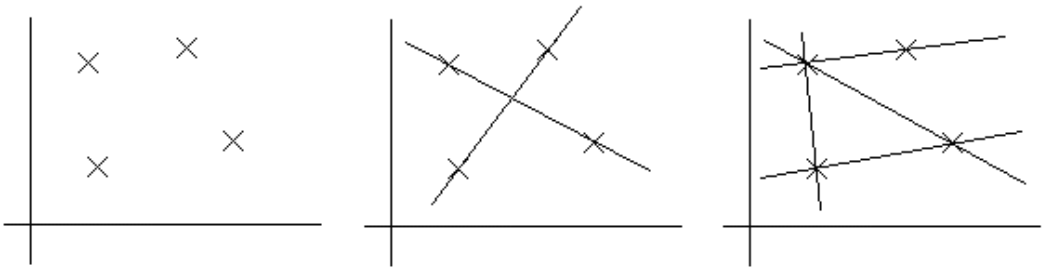
$$\mathrm{w}_r = 0.3,\ \mathrm{w}_g = 0.59,\ \mathrm{w}_b = 0.11$$

predstavlja svetilnost v točki t.

2.2 Houghova transformacija

Hough je našel način za iskanje sledov atomskih in subatomskih delcev na fotografskem filmu. Takšni delci zapustijo ravne pikčaste sledi na filmu, če ni magnetnega polja. Če pogledamo te sledi, vidimo, da so enostavno prepoznavne, vendar je zaradi velike količine slik ta postopek potrebno avtomatizirati.

Houghova ideja je, da uporabimo prostor vrstic v ravnini tako, da povežemo posamezne točke (slika 2.2).



Slika 2.2: Točke v 2D prostoru in morebitne združitve.

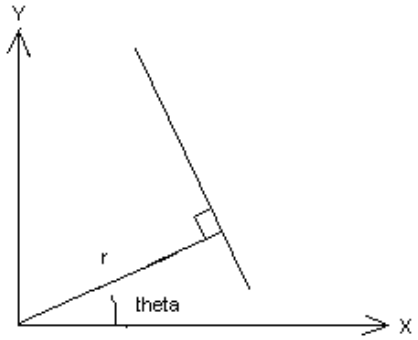
Premice v 2D prostoru lahko opišemo kot:

$$ax + by + c = 0 \tag{2.2}$$

Iz tega sledi, da premico lahko opišemo s s(a,b,c). V parametrični obliki to lahko zapišemo kot:

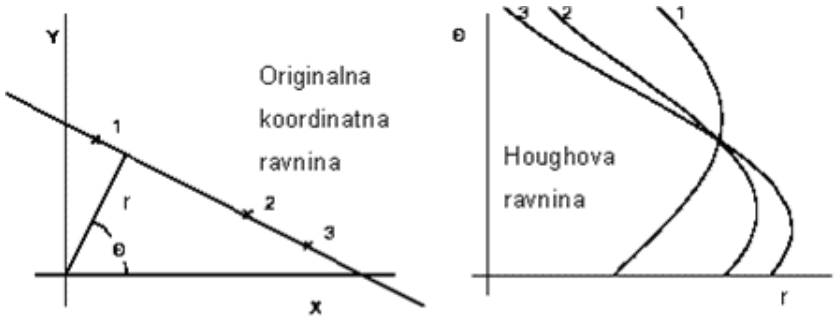
$$x * cos(\varphi) + y * sin(\varphi) = r \tag{2.3}$$

kjer je r dolžina normale od izhodišča, φ pa je usmerjenost glede na os x (slika 2.3). Za vsako točko na tej premici (x,y) sta r in φ konstantna in enaka.



Slika 2.3: Parametrična oblika prikaza ravnih črt v 2D.

V okviru analize slik, so točke konture objekta v 2D prostoru (x_i, y_i) poznane in se lahko smatrajo kot konstante, potrebno pa je najti ustrezne spremenljivke r in φ. Če narišemo možne vrednosti (r, φ) za dane (x_i, y_i) dobimo sinusoidne krivulje v polarnem Houghovem parametričnem prostoru (slika 2.4)).

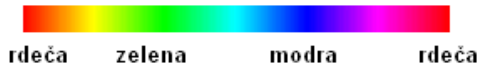


Slika 2.4: Kartezijski in Houghov prostor.

Za točke na ravni črti, se v Houghovem prostoru dobiva približne (zaradi šuma) preseke krivulj. Problem iskanja črte v prostoru R^2 se zoži na iskanje točke kopičenja v Houghovem prostoru vrstic. Če obstaja nekaj črt, dobimo ustrezno število stekališč.
Postopek lahko posplošimo za iskanje drugih karakteristik kot so krožnice ali elipse. V tem primeru se kompleksnost preračunavanja poveča, ker imamo v primeru krožnice tridimenzionalni prostor, v primeru elipse pa kar petdimenzionalen.

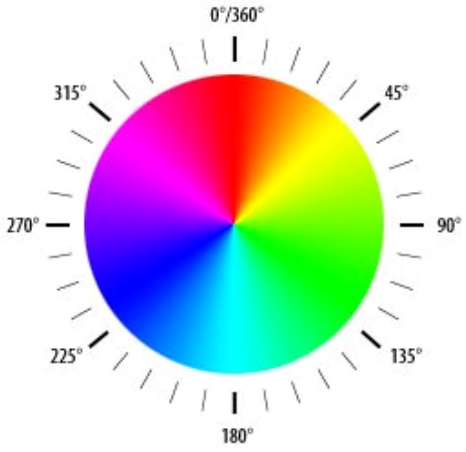
2.3 HSL barvni model

HSL je še en način za opis barve s tremi parametri. Zaslon računalnika uporablja RGB, vendar taka predstavitev ni preveč intuitivna. HSL način opisa barv je bolj intuitiven, vendar ga je potrebno pretvoriti v RGB preden lahko z njim narišemo točko (ang. pixel). Najlepša uporaba tega barvnega modela je, da lahko z njim preprosto ustvarimo mavrični razpon ali spremenimo barvo, svetlost ali nasičenost slike. HSL ima tri parametre H, S in L, ki predstavljajo odtenek (ang. Hue), nasičenost (ang. Saturation) in svetlost (ang. Lightness). Odtenek označuje občutek barve za svetlobo, z drugimi besedami, če je barva rdeča, rumena, zelena, modra,... Ta predstavitev izgleda skoraj enako kot vidni spekter svetlobe, razen da je sedaj na desni magenta (kombinacija rdeče in modre), namesto vijolične svetlobe s frekvenco višjo od modre (slika 2.5).



Slika 2.5: Barvni odtenki.

Odtenek deluje krožno, tako da ga lahko predstavimo na krogu (slika 2.6). Odtenek 360° izgleda enako kot odtenek 0°.



Slika 2.6: Odtenki predstavljeni na krogu [10].

Nasičenost označuje stopnjo, do katere se odtenek razlikuje od nevtralno sive. Vrednostni razpon je od 0 do 1, kjer 0 pomeni brez barve, 1 pa največjo zasičenost danega odtenka pri danem odstotku osvetlitve. Večji kot je spekter

svetlobe, koncentriran okoli ene valovne dolžine, bolj bo barva nasičena (slika 2.7).

Slika 2.7: Nasičenost.

Svetilnost predstavlja osvetlitev barve. Pri 0% je barva popolnoma črna, pri 50% je barva čista in pri 100% postane bela (slika 2.8). V HSL barvnem modelu je barva z maksimalno osvetlenostjo (L=255) vedno bela, ne glede na vrednost komponent odtenka ali nasičenosti. Svetilnost S je definirana kot :

$$S = (maxColor + minColor)/2 \tag{2.4}$$

kjer je maxColor R, G ali B komponenta z največjo vrednostjo in minColor tista z minimalno vrednostjo.

Slika 2.8: Svetilnost.

Poglavje 3

Segmentacija slike

V tej diplomski nalogi bo zajemanje slike potekalo preko spletne kamere. Prvi korak pri sestavljanju je segmentacija ali razčlenitev vhodne slike na posamezne kose. Cilj segmentacije je izločitev vsakega kosa iz digitalne slike, na kateri imamo vse kose sestavljanke. Posamezen kos lahko določimo s štirimi točkami vogalov in tipom roba za vsako stranico (raven, vbočen, izbočen).

Uspešnost segmentacije je odvisna od kakovosti zajete slike kosov sestavljanke. Če so kosi in podlaga preveč podobnih intenzitet, potem imamo lahko težave pri iskanju robov posameznih kosov. Idealno bi bilo zajemanje slike na dveh različnih podlagah, saj bi tako lažje določili robove tistih delov kosa, ki so barvno podobni podlagi. Ker pa to ni mogoče, je potrebno poiskati podlago, ki je kontrastno čimbolj različna od same sestavljanke. Težavo lahko predstavljajo tudi odboji svetlobe, predvsem kadar je vir svetlobe preblizu mesta zajemanja slike (npr. namizna svetilka). Težavo predstavljajo tudi sence ob robovih kosov, predvsem kadar imamo sestavljanko, ki je debelejša. Kosi se prav tako ne smejo prekrivati ali ležati neposredno drug poleg drugega, saj bo program v tem primeru dva ali več kosov obravnaval kot enega in bo tako določanje kosa neuspešno.

3.1 Iskanje robov na sliki

Najprej moramo na vhodni sliki poiskati robove. Robovi so definirani kot področja, kjer je sprememba v intenziteti slike najbolj izrazita, saj so spremembe v intenziteti največje na prehodu iz enega dela objekta v drug del, oziroma iz enega objekta v drug objekt, torej ravno na robovih.

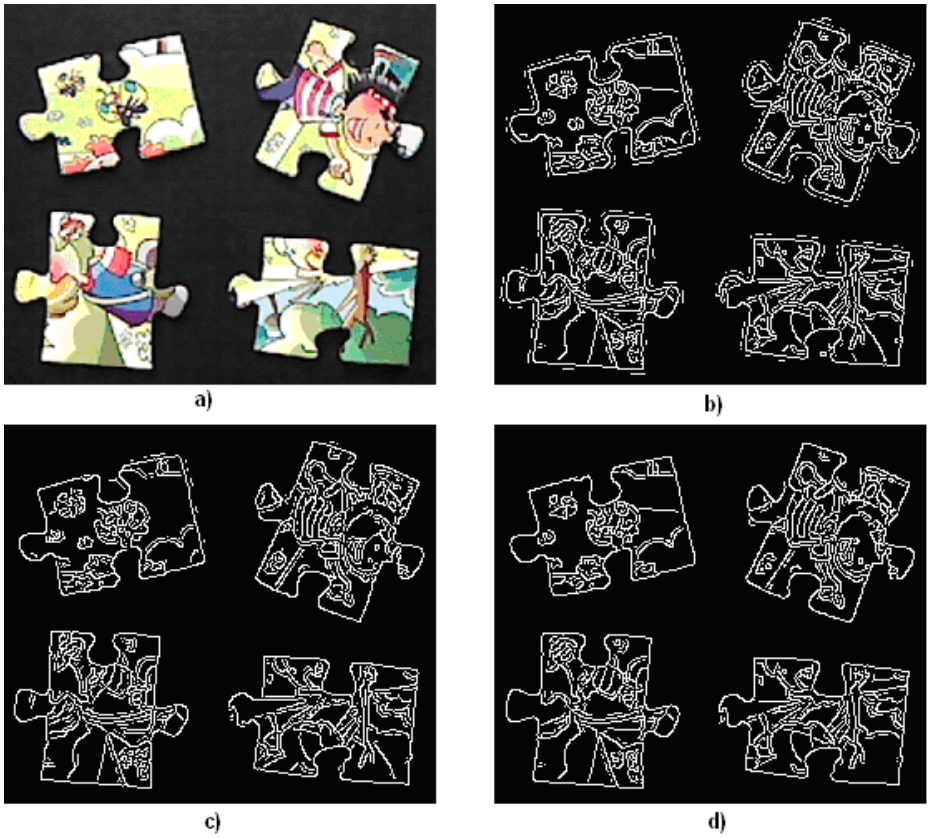
Robove na sliki bomo iskali s pomočjo algoritma, ki ga je leta 1986 razvil John F. Canny [8, 9]. Kljub temu, da je algoritem že precej star, je postal ena

standardnih metod za odkrivanje robov in se še vedno uporablja v raziskavah. Cannyjev algoritem za iskanje robov (angl. Canny edge detection algorithm), je poznan pod imenom "optimalni detektor robov". Njegov namen je bil razvoj algoritma, ki je optimalen glede na sledeče kriterije:

- Detekcija (angl. detection) - obstajati mora visoka verjetnost da smo zaznali resnične robove, hkrati pa nizka verjetnost zaznave napačnih robnih točk.
- Lokalizacija (angl. localization) - zaznani robovi morajo biti čim bližje pravim robovom na sliki.
- Minimalni odziv (angl. only one response to a single edge) - en resničen rob naj ne predstavlja več zaznanih robov ter v primeru, da slika vsebuje šum, le-ta ne sme biti označena kot robna točka.

Neizogibno je, da bodo slike zajete s kamero vsebovale določeno količino šuma, zato Cannyjev algoritem iskanja robov najprej zgladi sliko in s tem zmanjša prisoten šum ter tako prepreči zamenjavo šuma za robove. Nadalje poišče gradiente slike z uporabo Sobelovega operatorja, ki računa spremembo (gradient) intenzitete v vsaki točki in kot rezultat daje smer največjega možnega povečanja ter razmerje spremembe v tej smeri. Matematično gledano filter deluje tako, da vsako slikovno točko z njenimi sosedami pomnoži z matriko velikosti 3x3. Vrsta gradientov se v naslednjem koraku skrči z uporabo histereze, ki se uporablja z namenom sledenja vzdolž ostalih točk, ki niso bile izločene. Histereza uporablja dve meji. Če je magnituda pod prvo mejo, se ta postavi na točko nič (ne-rob). V primeru, da je magnituda nad višjo mejo, pa se ta postavi na točko ena (rob). Če je magnituda med nižjo in višjo mejo, je prav tako postavljena na točko nič (ne-rob), razen v primeru, če obstaja pot od te točke do točke z gradientom nad višjim pragom.

Cannyjev operator je določen z dvema parametroma, spodnjo in zgornjo mejo. Za dobre rezultate se zgornja meja ponavadi postavi precej visoko, spodnja meja pa precej nizko. Postavitev previsoke spodnje meje povzroči razpad šumnih robnih točk. Postavitev prenizke zgornje meje pa poveča število lažnih nezaželenih robnih točk (slika 3.1).

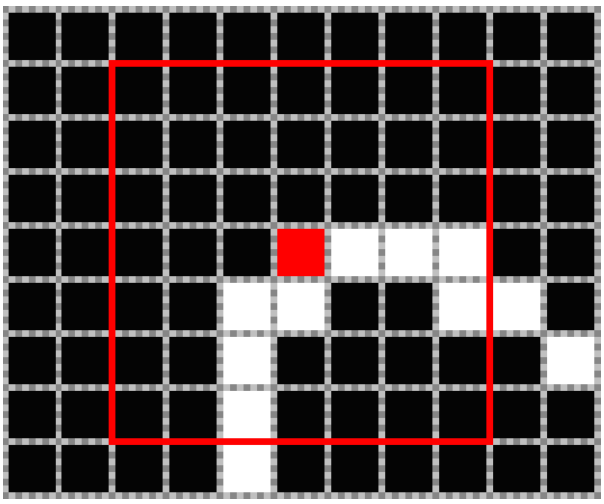


Slika 3.1: (a) vhodna slika, s pomočjo katere bomo pokazali razlike pri iskanju robov s spreminjanjem zgornje meje Cannyjevega operatorja, (b-d) Cannyjeva detekcija robov s spodnjim pragom T1=25 in zgornjim pragom (b) T2=100, (c) T2=200, (d) T2=300.

3.2 Izločanje posameznih kosov

Cilj izločanja je imeti vsak kos v svoji sliki, kar nam bo pri kasnejši obdelavi omogočalo obdelavo vsakega kosa posebej. Kos iz slike izločimo tako, da na binarni sliki poiščemo prvo belo točko in jo prekopiramo v izhodno sliko. Nato pogledamo sosedne točke v območju velikosti 7x7 točk, kjer je trenutna točka v sredini (od obravnavane točke, 3 točke v vsako smer) (slika 3.2) in vsako najdeno belo točko prekopiramo v izhodno sliko, ter postopek ponovimo za vsako izmed njih. Končamo, ko belih točk zmanjka, v izhodni sliki pa imamo kopijo robov kosa. Če je dobljen rezultat vsaj velikosti 40x40 točk, smatramo da je dobljen obris pravilen kos. V nasprotnem primeru pa tak obris zavržemo,

ker predstavlja šum ali pa obris nečesa, kar ne spada v sestavljanko. Težava nastane, če sta dva kosa tako blizu, da med njima ni vsaj treh točk razmaka, ker program v takem primeru dva kosa obravnava kot enega.



Slika 3.2: Območje iskanja sosednjih belih točk.

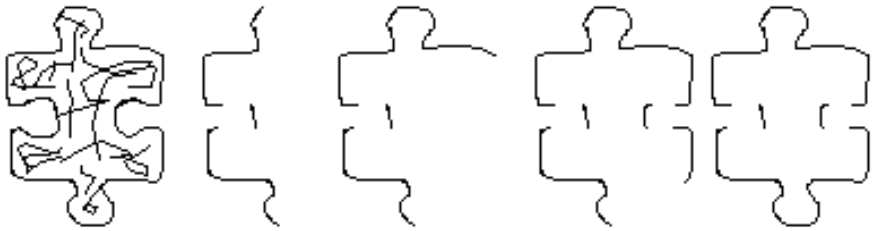
Poglavje 4

Določanje lastnosti kosa

Za vsak kos sestavljanke je potrebno določiti njegove lastnosti s pomočjo katerih bomo v nadaljevanju kose primerjali in jih združevali v izhodno sliko. Določiti je potrebno štiri vogalne točke kosa in tipe vseh štirih stranic, ki omejujejo kos. Natančnost določanja je zelo pomembna saj v primeru napake pri določanju tipa stranice sestavljanke ne bo mogoče zložiti, slabo določen vogal pa povzroči večje napake pri primerjanju kosov.

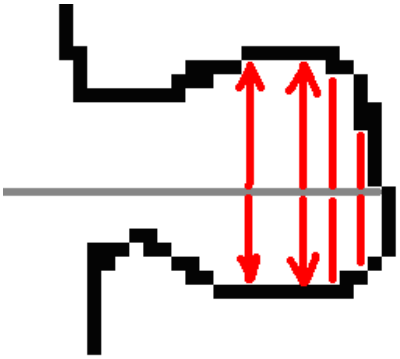
4.1 Odstranjevanje notranjih robov

Preden lahko izvedemo Houghovo transformacijo je potrebno odstraniti robove, ki so znotraj kosa. To želimo storiti zato, da nam bo Houghova transformacija res vrnila premice robov kosa in ne kake premice, ki leži na črti iz notranjosti kosa. Rezultat dobimo tako, da v vsaki vrstici poiščemo prvo belo točko, ki je najbližja levemu robu slike in jo zapišemo v izhodno sliko, preostale točke v tej vrstici pa preskočimo in z izvajanjem nadaljujemo v naslednji vrstici. Na ta način dobimo levi obris kosa, kot je prikazan na drugem mestu na sliki 4.1. Nato ponovimo postopek, le da tokrat iščemo najbližjo belo točko desnemu robu slike, ter tako dobimo še obris desnega roba kosa. Podobno storimo še v navpični smeri in tako dobimo še zgornji in spodnji rob kosa.



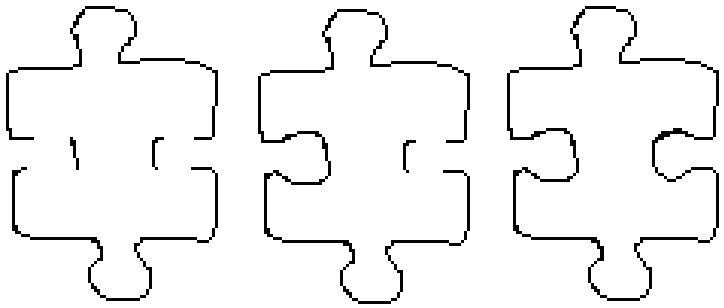
Slika 4.1: Potek odstranjevanja notranjih robov.

Opazimo, da udrtine na stranicah, ki imajo konveksne robove, nimajo celotnega obrisa, ker so robovi le teh "zakriti "z zunanjimi robovi kosa. Problem rešimo tako, da poiščemo sredino te udrtine in iščemo prvo črno točko v obeh smereh pravokotno na prejšno smer iskanja od najglobje točke udrtine do roba slike (slika 4.2).



Slika 4.2: Prikaz iskanja robov v udrtini.

Tako dobimo končen in popoln obris kosa, ki je primeren za nadaljno obravnavo (slika 4.3). Obris je debel eno točko, ni pa nujno, da je sklenjen. Kjer imamo vrzeli v obrisu s tem postopkom dobimo tudi točke v notranjosti samega kosa, ki sem ne sodijo, odstranimo pa jih z zapolnjevanjem vrzeli v obrisu, kar je naslednji korak.



Slika 4.3: Določanje robov pri udrtinah.

4.2 Zapolnjevanje vrzeli v obrisu kosa

Notranjost vsakega kosa želimo napolniti s črno barvo, kar potrebujemo pri postopku primerjanja oblike dveh kosov med seboj. Preden pa to lahko storimo, je potrebno zapolniti vrzeli v obrisih kosov. Program najprej poišče prvo črno točko na vhodni sliki in jo prekopira v izhodno sliko, nato poišče njej najbližjo črno točko ter ju v primeru, da je razdalja med njima več kot eno točko, poveže s črto. Postopek se ponovi za vsako točko, dokler ne pridemo nazaj v izhodišče. Takrat se izvajanje prekine, v izhodni sliki pa imamo obris kosa brez vrzeli. S tem postopkom smo odstranili tudi točke iz notranjosti kosa, ki so ob vrzelih ostale po prejšnjem postopku, če le te niso preveč blizu roba. Če so preblizu roba, postanejo del obrisa. Tak obris sedaj lahko uporabimo v naslednjem koraku, kjer določimo vogalne točke kosa.

4.3 Določanje vogalov kosov

Določanje vogalnih točk je ena od pomembnejših stvari, ki jih je potrebno pripraviti, preden začnemo kose primerjati, saj napačno določen vogal predstavlja veliko težavo pri računanju ujemanja posameznih kosov med seboj. Slabo določen vogal pomeni večjo verjetnost, da program na določeno mesto postavi napačen kos, povzroči pa tudi prekrivanje ali razmak med kosi na izhodni sliki.
Določanje vogalov poteka v treh korakih:

- Iskanje premic na robovih kosa in določanje njihovih presečišč

- Iskanje točke na robovih kosa
- Računanje povprečne oddaljenosti obrisa kosa od roba slike

4.3.1 Iskanje premic na robovih kosa in določanje njihovih presečišč

Pri določanju vogalov kosa najprej izvedemo Houghovo transformacijo, s pomočjo katere dobimo več daljic na robovih kosa. Nato izračunamo pod kakšnim kotom ležijo dobljene daljice tako, da z enačbama (4.2) in (4.3) izračunamo α_i. Med daljicami, katerih kot se od navpičnice ali vodoravnice razlikuje največ 10°, poiščemo najdaljšo tako, da z enačbo (4.1) izračunamo razdaljo daljice R_i, ki jo določata točki (X_1, Y_1) in (X_2, Y_2).

Izračun dolžine daljice R_i, koeficienta k_i in kota med premicami α_i:

$$R = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2} \tag{4.1}$$

$$k = \frac{(Y_2 - Y_1)}{(X_2 - X_1)} \tag{4.2}$$

$$\alpha = ctg \left| \frac{k1 - k2}{1 + k1 * k2} \right| \tag{4.3}$$

Tako smo dobili prvo daljico D_1 na robu kosa, ki nam bo služila kot osnova za določanje ostalih treh. Najprej poiščemo najdaljšo daljico, ki je vzporedna (kot med njima je največ 10°) D_1 in je vsaj polovico dolžine kosa oddaljena od D_1. Razdaljo med daljicama dobimo tako, da si izberemo točko T na daljici D_1 in z uporabo enačbe (4.4) izračunamo razdaljo med točko in daljico, ki jo določata točki (X_1, Y_1) in (X_2, Y_2). Izračun razdalje med točko in daljico:

$$k = \frac{((Y_2 - Y_1) * (T_x - X_1) + (X_2 - X_1) * (T_y - Y_1))^2}{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \tag{4.4}$$

Kjer je: (T_x, T_y) točka na daljici D_1, (X_1, Y_1) in (X_2, Y_2) pa točki, ki določata drugo daljico.

Sedaj imamo dve daljici, ki predstavljata levi in desni ali pa zgornji in spodnji rob kosa, določiti je potrebno še preostali dve. Med daljicami, ki so pravokotne na D_1 (kot med njimi je med 80° in 100°) poiščemo najdaljšo ter tako dobimo tretjo daljico D_3. Zadnja daljica mora biti pravokotna na D_1 in vzporedna z

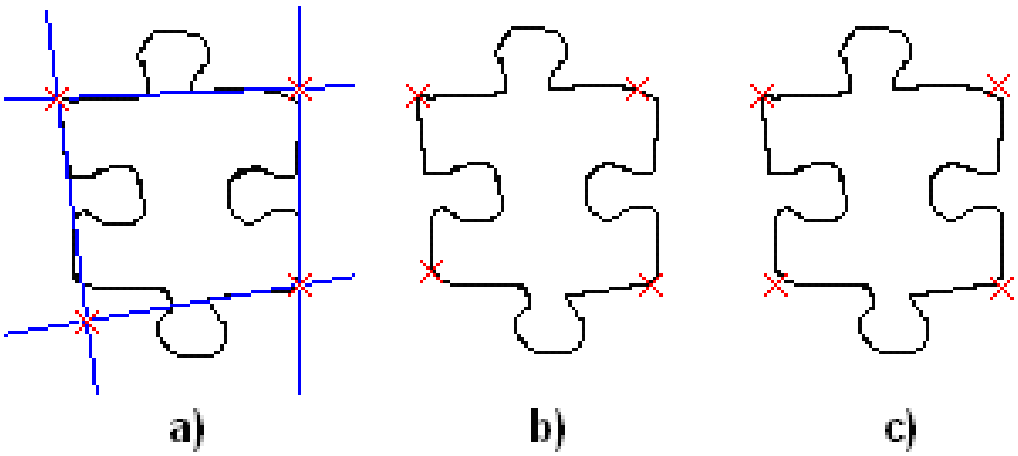
D_3 ter od nje oddaljena vsaj polovico dolžine kosa.

Določiti želimo presečišča premic, ki jih določajo daljice D_1...D_4, zato najprej določimo enačbe premic v eksplicitni obliki y=kx+n. Izračunati je potrebno k_i z enačbo (4.2) ter n_i z enačbo n=y-kx. Tako dobimo enačbe štirih premic P_1...P_4, ki potekajo skozi daljice D_1...D_4. Včasih na kateri od stranic (zaradi slabe detekcije robov, senc, ...) s Houghovo transformacijo ne dobimo nobene daljice, zato v takem primeru na tisti stranici določimo kar premico, ki poteka po robu slike kosa. Enačbe teh premic so x=0, y=0, x=w in y=h, kjer je w širina slike kosa, h pa višina slike kosa.

Točke presečišč premic določimo z rešitvijo sistema enačb y=f(x) in y=g(x) med premicami P_1 in P_3, P_1 in P_4, P_2 in P_3 ter P_2 in P_4. Potrebno je le še ugotoviti, katera točka predstavlja kateri vogal kosa, kar storimo tako, da za vsako točko izračunamo razdaljo do vseh štirih vogalov slike kosa. Vogal, ki je točki najbližji, predstavlja tudi vogal kosa. Tako dobimo zgornji levi TL(top left), zgornji desni TR(top right), spodnji levi BL(bottom left) in spodnji desni BR(bottom right) vogal kosa.

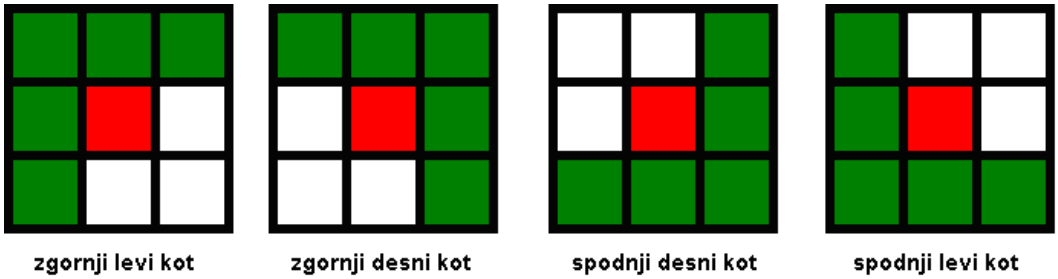
4.3.2 Iskanje vogalne točke na obrisih kosa

Točke, ki smo jih dobili s pomočjo presečišč premic večinoma ne predstavljajo točnih vogalov kosov (slika 4.4a), ampak le njihove približke, katere je potrebno popraviti ter tako dobiti točke, ki bolje predstavljajo dejanske vogale kosa, zato poskušamo s pomočjo presečišč premic dobiti točke na obrisu kosa (slika 4.4b).



Slika 4.4: Iskanje vogalnih točk.

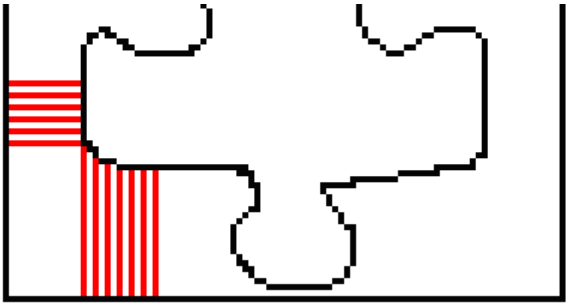
Za vsakega izmed štirih vogalov najprej poiščemo točko na obrisu kosa. Do te točke pridemo tako, da iz presečišč premic iščemo prvo črno točko po diagonali od točke presečišča premic proti središču kosa (če se presečišče nahaja zunaj robov kosa) ali proti zunanjemu robu slike (če se presečišče nahaja v notranjosti kosa). Nato pa se po obrisu kosa premikamo v dovoljenih smereh (slika 4.5), dokler se lahko. Ko pridemo do točke iz katere se ni več moč premakniti v nobeno izmed dovoljenih smeri, nam ta točka predstavlja novi vogal kosa.



Slika 4.5: Dovoljene smeri za posamezne vogale.

4.3.3 Računanje povprečne oddaljenosti obrisa kosa od roba slike

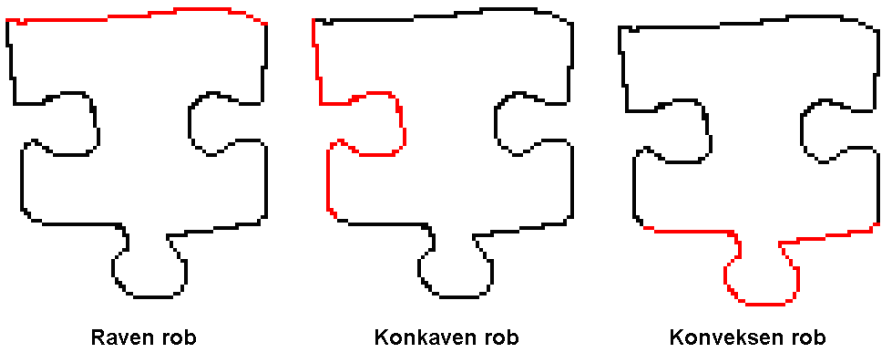
Ker pri detekciji robov lahko nastanejo napake (sence, majhna kontrastna razlika med kosom in podlago), vogalna točka, ki leži na obrisu kosa, pogosto ni najboljša, saj v vsaj eni smeri leži preveč v notranjosti ali preveč zunaj kosa. Zato je potrebno to točko malce premakniti in sicer tako, da v vodoravni in navpični smeri izračunamo povprečno oddaljenost obrisa kosa od roba slike (slika 4.6), ter tako dobimo x in y koordinate točke, ki kar najbolje predstavlja vogalno točko kosa (slika 4.4c).
S tem postopkom smo dobili vse štiri vogalne kose, kar je predpogoj za uspešno primerjanje kosov med seboj ter končno zlaganje posameznih kosov v sestavljeno sliko.



Slika 4.6: Iskanje povprečne oddaljenosti obrisa kosa od roba slike.

4.4 Določanje stranic kosa

Vsakemu kosu je potrebno določiti kakšne so njegove stranice. Možnosti so tri: ravna stranica, stranica s konveksno krivuljo in stranica s konkavno krivuljo (slika 4.7). Konveksne stranice preprosto določimo tako, da pogledamo koordinate vogalnih točk, ki omejujeta rob. Če je vsaj ena od roba slike oddaljena več kot 10 odstotkov višine ali širine (odvisno kateri rob določamo), potem ima taka stranica konveksno krivuljo. Za določanje konkavne stranice najprej poiščemo prvi ravni del stranice kosa, nato pa iščemo vsaj pet zaporednih robnih točk, ki so od roba slike oddaljene več kot 20 odstotkov širine ali višine kosa. Če najdemo pet takih točk potem spet iščemo vsaj pet točk, ki so od roba slike oddaljene manj kot 15 odstotkov širine ali višine slike kosa ter tako najdemo drugi ravni del stranice. Kadar so vsi ti pogoji izpolnjeni, takrat lahko obravnavano stranico določimo kot konkavno stranico. Preostale stranice, ki niso ne konkavne ne konveksne pa so stranice z ravnim robom, torej so to robni ali vogalni kosi sestavljanke.



Slika 4.7: Vrste robov kosa.

Tip kosa preprosto določimo tako, da preštejemo število ravnih stranic, ki jih najdemo pri obravnavanem kosu:

0 ravnih stranic - Notranji kos
1 ravna stranica - Robni kos
2 ravni stranici - Vogalni kos

Če pri kosu najdemo tri ali štiri ravne stranice potem je prišlo do napake pri detekciji robov, zato z izvajanjem programa prekinemo, ker tak kos pri običajnih sestavljankah ne obstaja in ga zato ni mogoče zložiti poleg ostalih. Prav tako pride do napake, kadar ugotovimo, da sta ravni leva in desna ali zgornja in spodnja stranica. Vse ostale kombinacije stranic so dovoljene in predstavljajo običajne kose sestavljanke kot so predstavljeni v tabeli 5.1 (stran 28).

Poglavje 5

Primerjava kosov

Za sestavljanko z N kosi ob upoštevanju vseh možnih ujemanj z drugimi kosi imamo $4 * 4 * (N - 1)$ primerjav. Vsak kos ima na splošno štiri robove. Tako si lahko predstavljamo, kako veliko je skupno število primerjanj med vsemi kosi sestavljanke. V problemu zlaganja sestavljank obstaja "eksplozija kombinacij". Da bi omilili problem "eksplozije kombinacij", smo uvedli modele kosov za sestavljanke, katerih kosi imajo štiri robove. Obstajajo štiri vogalne točke, robno krivuljo pa lahko razdelimo na štiri robove, ki povezujejo posamezne vogalne točke. Vsak od štirih robov spada v enega izmed naslednjih treh vzorcev:

- Raven rob označen z "L"
- Rob z konkavno krivuljo označen z "C"
- Rob z konveksno krivuljo označen z "V"

Te tri vrste robov so značilne za vse sestavljanke s štirimi robovi in jih uporabljamo za razvrstitev kosov v razrede. Če vsak kos pogledamo podrobno, vidimo, da lahko vse kose sestavljanke razvrstimo v tri kategorije: vogalni kos, robni kos in notranji kos. Vogalni kos označen z R, se nahaja v kotu sestavljanke. Vedno imamo natanko štiri take kose v sestavljanki s štirimi robovi, ne glede na število kosov, ki so del sestavljanke. Robne kose nadalje delimo na štiri kategorije, kot so prikazane v vrstici 1 do 4 v prvem stolpcu tabele 5.1, poimenovani so R_0, R_1, R_2 in R_3. Konfiguracije robov R_i (i=0,1,2,3) so "CCLL", "VCLL", "CVLL"in "VVLL" v vrstnem redu kot so podani v drugem stolpcu od 1 do 4 vrstice (tabela 5.1). Enaka kot kombinacija "CCLL" je tudi "CLLC" saj je "CLLC" mogoče dobiti z vrtenjem "CCLL" ali s spremembo začetne vogalne točke. To velja tudi za druge kose. V tabeli 5.1 je začetna vogalna točka

vseh kosov sestavljanke v spodnjem levem kotu, in robovi so obravnavani v smeri urinega kazalca.

Robne kose označene z E, najdemo na štirih robovih celotne sestavljanke. Število robnih kosov se razlikuje glede na velikost sestavljanke. Ob upoštevanju vseh možnih kombinacij roba vzorcev dobimo osem vrst robnih kosov, ki so prikazani v vrsticah od 5 do 11 v prvem stolpcu tabele 5.1, in so poimenovani E0 do E7. Konfiguracije robov so "CCCL", "CVCL", ..., "VVVL" v vrstem redu kot so podani v drugem stolpcu od 5. do 11. vrstice.

Notranje kose označene z I, najdemo v notranjem območju sestavljanke. Število notranjih kosov je prav tako odvisno od velikosti sestavljanke in teh kosov (razen pri majhnih sestavljankah) je precej več kot robnih kosov. Obstaja šest vrst notranjih kosov prikazane so v prvem stolpcu od 13 do 18 vrstice (tabela 5.1) in poimenovane z I_0, I_1, ..., I_5. Konfiguracije robov so "CCCC", "VCCC", ..., "VVVV" v vrstem redu kot so podani v drugem solpcu.

Tako ugotovimo, da je sestavljanke s štirimi robovi mogoče razvrstiti v 18 kategorij, ki so označene z S_M kjer je S_M=$\{\{R_0,...,R_3\},\{E_0,...,E_7\},\{I_0,...,I_5\}\}$. Slike teh vzorcev so prikazane v prvem stolpcu tabele 5.1.

Uvedimo relacijo "PT:ET", ki izraža združljivost robov med kosi sestavljanke, kjer PT(Piece Type) pomeni tip sosednjega kosa (npr. R, E ali I), ET pa tip sosednjega robu (L, C ali V). Združljivost med temi 18 tipi kosov so povzete v stolpcih 3 do 6 tabele 5.1. 3., 4., 5., in 6. stolpec prikazujejo potreben tip sosednjega kosa in potreben tip sosenjega robu za vsakega izmed štirih robov kosa. Za vogalne kose obstaja relacija E:V in E:C, kot je prikazano v 3. in 4. stolpcu (tabela 5.1). E:V pomeni, da je potreben sosednji kos robni kos ter ima konveksnen rob. Podobno velja za robne kose, kjer obstaja šest vrst povezav - E:V, E:C, R:V, R:C, I:V in I:C. In za notranje kose obstajajo štiri vrste povezav - E:V, E:C, I:V in I:C. "None"v 5. in 6. stolpcu pomeni, da sosednji kos ne obstaja.

Vsebina te tabele igra pomembno vlogo pri reševanju problema zlaganja sestavljank, saj lahko z njeno uporabo močno skrajšamo čas, potreben za rešitev problema. Prihranek časa pri primerjanju kosov je različen za vsako sestavljanko, ker različne oblike kosov zahtevajo različno število primerjav.

	Tip kosa	Opis robov	Rob 0	Rob 1	Rob 2	Rob 3
R	 R_0	CCLL	E.V	E.C	None	None
	R_1	VCLL	E.C	E.V	None	None
	R_2	CVLL	E.V	E.C	None	None
	R_3	VVLL	E.C	E.C	None	None
E	 E_0	CCCL	E.V, R.V	I.V	E.V, R.V	None
	E_1	CVCL	E.V, R.V	I.C	E.V, R.V	None
	E_2	VCCL	E.C, R.C	I.V	E.V, R.V	None
	E_3	CCVL	E.C, R.V	I.V	E.C, R.C	None
	E_4	VVCL	E.C, R.C	I.C	E.V, R.V	None
	E_5	CVVL	E.V, R.V	I.C	E.C, R.C	None
	E_6	VCVL	E.C, R.C	I.V	E.C, R.C	None
	E_7	VVVL	E.C, R.C	I.C	E.C, R.C	None
I	 I_0	CCCC	E.V, I.V	E.V, I.V	E.V, I.V	E.V, I.V
	I_1	VCCC	E.C, I.C	E.V, I.V	E.V, I.V	E.V, I.V
	I_2	VCVC	E.C, I.C	E.V, I.V	E.C, I.C	E.V, I.V
	I_3	CVVC	E.V, I.V	E.C, I.C	E.C, I.C	E.V, I.V
	I_4	VVVC	E.C, I.C	E.C, I.C	E.C, I.C	E.V, I.V
	I_5	VVVV	E.C, I.C	E.C, I.C	E.C, I.C	E.C, I.C

Tabela 5.1: Model kosov sestavljanke.

Tabela 5.2 prikazuje prihranek časa za štiri različne sestavljanke. V prvi vrstici je število kosov, ki tvorijo sestavljanko. Druga vrstica prikazuje število vseh primerjav, če bi vsak kos primerjali z vsako stranico vseh ostalih kosov, ki še niso sestavljeni. Tretja vrstica prikazuje število primerjav, ob upoštevanju pravil iz tabele 5.1, četrta vrstica pa prikazuje število primerjav, kjer kot kandidate za primerjanje izločimo tudi tiste kose, katerih dolžine stranic niso enake dolžini stranice kosa, s katerim primerjamo (razlikujejo se za več kot 10%). Zadnja vrstica kaže prihranek števila primerjav in opazimo lahko, da je v nekaterih primerih le-ta tudi več kot 90 odstoten.

Št. kosov	12	16	20	25
Št. vseh možnih primerjav $N * (N - 1) * 2$	264	480	760	1200
Št. primerjav ob upoštevanju pravil iz tabele 5.1	48	84	132	210
Št. primerjav ob upoštevanju pravil iz tabele 5.1 in dolžin stranic	26	84	70	210
Zmanjšanje števila primerjav v %	90.15	82.50	90.79	82.50

Tabela 5.2: Število primerjav med kosi pri različnih pogojih.

Posamezne kose med seboj primerjamo tako, da ju postavimo drug ob drugega in izračunamo njuno ujemanje. Primerjamo lahko desno stran prvega kosa z levo stranjo drugega, ali pa spodnjo stran prvega z zgornjo stranjo drugega kosa. Ostali dve možnosti (leva stran prvega z desno drugega, ali zgornja stran prvega z spodnjo drugega) nista potrebni ker v tem primeru samo zamenjamo kosa in lahko uporabimo prvi dve metodi. Za vsak par kosov, ki ju primerjamo dobimo neko število, ki predstavlja ujemanje. Manjše kot je število, boljše je ujemanje. Ujemanje je sestavljeno iz petih atributov:

- Ujemanje oblike kosov
- Ujemanje barvnih (RGB) komponent točk ob robu kosov
- Ujemanje odtenkov točk ob robu kosov
- Ujemanje intenzivnosti točk ob robu kosov
- Ujemanje svetilnosti točk ob robu kosov

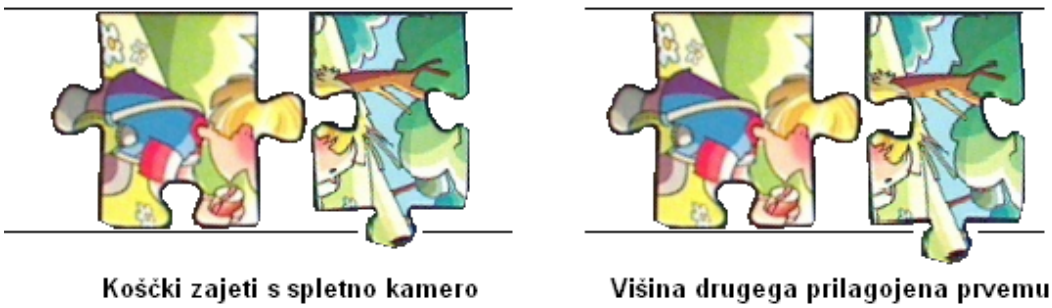
Ocena ujemanja je obtežena vsota teh petih atributov.

5.1 Ujemanje oblike kosov

Ujemanje oblike dobimo tako, da kosa postavimo drug ob drugega in izračunamo površino prekrivanja kosov in površino praznega prostora med njima. Preden lahko kosa postavimo drug ob drugega, ju moramo poravnati in enega ustrezno raztegniti/skrčiti tako, da se vogali čimbolj ujemajo

5.1.1 Raztezanje/skrčevanje kosa

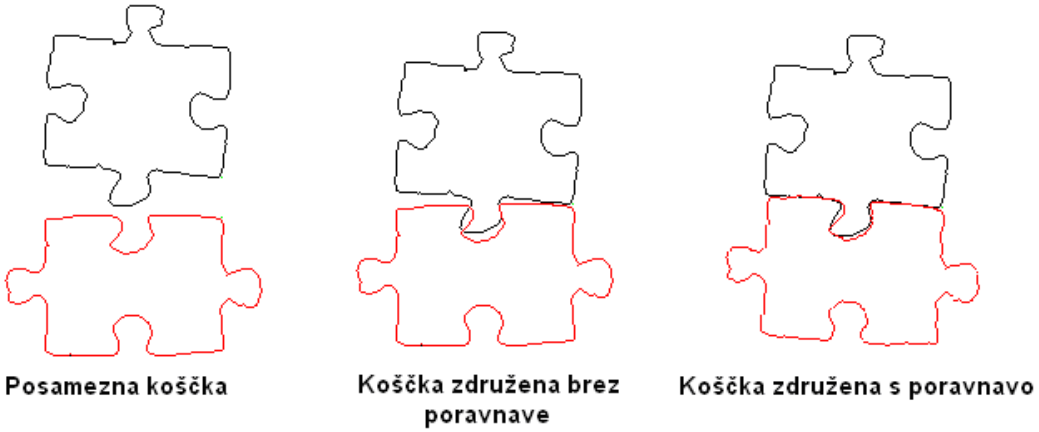
Ko postavimo dva kosa drug poleg drugega ugotovimo, da se njuni višini ali širini lahko precej razlikujeta (slika 5.1). Do tega pride, ker kose zajemamo preko spletne kamere in kot pod katerim kamera zajame določen kos, je odvisen od pozicije le tega na podlagi. Kosi na robu "vidnega"polja kamere so zaradi perspektivnega popačenja videti manjši kot tisti na sredini, čeprav so enake velikosti. Primerjava oblik kosov, ki sta zaradi perspektivnega popačenja različnih velikosti, nam da napačno informacijo o ujemanju kosov, zato je potrebno dimenzije enega kosa prilagoditi drugemu. To storimo tako, da izračunamo razdalje med vogali obeh kosov, iz teh pa dobimo količnik, za katerega moramo drugi kos raztegniti ali skrčiti. Če je količnik razdalj med vogali kosov večji kot ena, kos raztegnemo, sicer ga skrčimo.



Slika 5.1: Prilagajanje višine kosov.

5.1.2 Poravnava naklona kosov

Kose imamo poravnane približno vodoravno, vendar lahko zaradi napak pri določanju robov naklon kosa odstopa za nekaj stopinj. V primeru, da je en kos nagnjen v eno smer drug pa v nasprotno, potem je razlika med njima že precejšnja in brez poravnave bo površina prekrivanja ali praznega prostora med njima velika in posledično bo ocena ujemanja slabša, kot bi morala biti. Kose lahko poravnamo v navpični smeri glede na spodnjo in zgornjo stranico kot prikazuje slika 5.2 ali pa v vodoravni smeri glede na levo in desno stranico.



Slika 5.2: Združevanje kosov.

Naklon stranice kosa glede na vodoravnico dobimo tako, da izračunamo kot α_i daljice ki jo določata vogala stranice z enačbo:

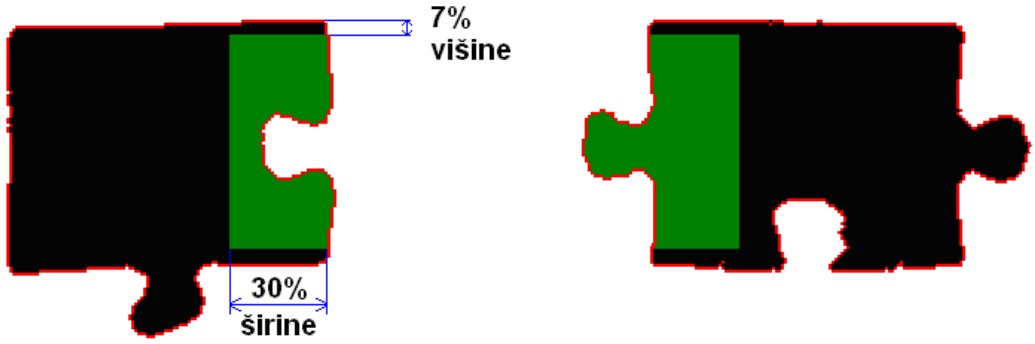
$$\alpha_i = arctan(\frac{x_2 - x_1}{y_2 - y_1}) \tag{5.1}$$

kjer sta točki T(x_1,y_1) in T(x_2,y_2) vogala kosa.
Kot za katerega je potrebno kos zavrteti, je razlika med kotom naklona stranice prvega kosa in kotom stranice drugega kosa. Izhodišče vrtenja postavimo v vogalno točko preko katere bomo kosa postavili drug ob drugega.

5.1.3 Izračun ujemanja

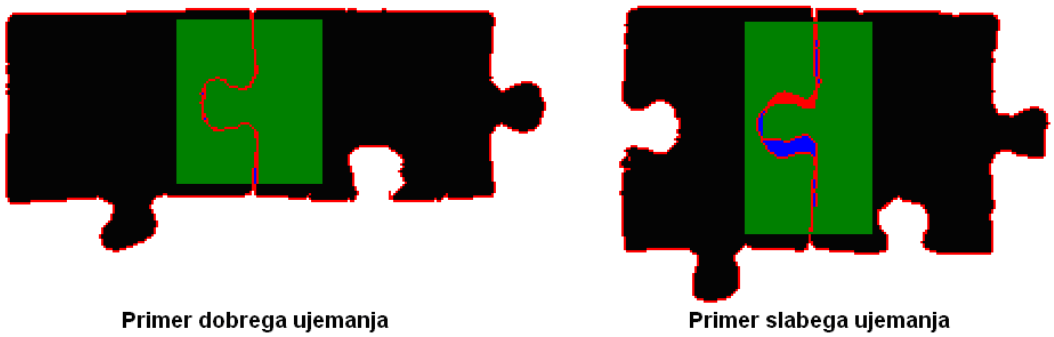
Ocena ujemanja kosov, je vsota površine prekrivanja in površine praznega prostora med njima. Najprej za vsak kos posebej določimo območje, na katerem bomo prešteli število črnih pik (slika 5.3). To območje je veliko 30 odstotkov širine kosa od stranice h kateri želimo postaviti drugi kos in 86 odstotkov višine kosa, ker na vsaki strani območje odmaknemo od roba za 7 odstotkov ter tako iz izračuna izločimo območje blizu vogalov, ki zaradi pogostih napak pri določanju robov kosov lahko daje zavajujoče rezultate pri računanju ujemanja oblike. Na tem območju preštejemo število črnih pik, kar nam predstavlja osnovno površino, s pomočjo katere kasneje izračunamo prekrivanje kosov.

Nato kosa postavimo skupaj in na območjih obeh kosov zopet preštejemo število črnih pik. Prvi del ocene ujemanja je tako razlika med vsoto črnih pik prvega in drugega kosa ter številom črnih pik združenih kosov. Ker je število



Slika 5.3: Območje računanja ujemanja kosov.

črnih pik pri združenih kosih vedno manjše ali enako vsoti črnih pik posameznih kosov, na ta način dobimo neko pozitivno število, ki nam pove kakšno površino prvega kosa prekriva drugi kos (Na sliki 5.4 označeno z rdečo). Drugi del ocene pa dobimo tako, da na istem območju na sliki združenih kosov preštejemo bele pike (Na sliki 5.4 označene z modro), katere predstavljajo območje, kjer se kosa ne stikata popolnoma in je med njima ostal prazen prostor



Slika 5.4: Primera dobrega in slabega ujemanja.

5.2 Ujemanje barvnih (RGB) komponent točk ob robu kosov

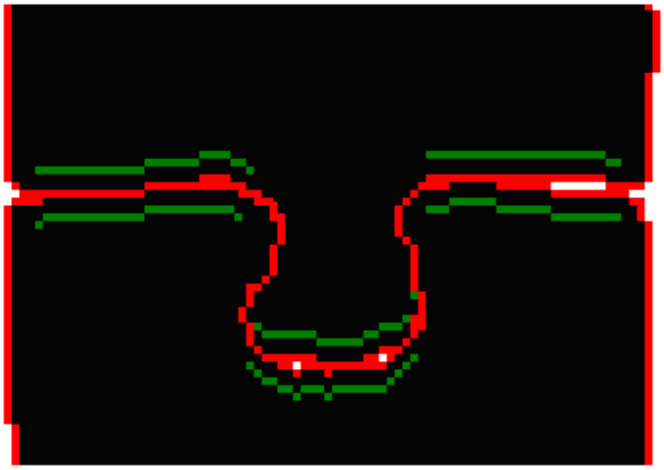
Kose želimo med seboj primerjati tudi po barvi, ker so si posamezni kosi včasih po obliki zelo podobni. Zato samo preverjanje oblike ne zadostuje in

potrebujemo še kakšno dodatno informacijo o ujemanju. Barvno ujemanje kosov izračunamo tako, da kosa postavimo skupaj nato pa v vsaki vrstici iz vsakega kosa preberemo točko (ang. pixel), ki je tri točke oddaljen od roba kosa (na sliki 5.5 označeno z zeleno). Lahko bi vzeli tudi točke neposredno na robu, vendar te točke pogosto ne nosijo najboljše informacije o barvi na tistem delu kosa, ker rob ni popolnoma natančno določen in lahko predstavlja sence, ki jih kos meče na podlago. Oceno ujemanja v teh točkah dobimo kot vsoto absolutnih razlik posameznih barvnih komponent po formuli:

$$U_{rgb} = |red(T_1) - red(T_2)| + |green(T_1) - green(T_2)| + |blue(T_1) - blue(T_2)| \tag{5.2}$$

kjer je T1 točka iz prvega kosa, T2 pa točka iz drugega kosa.

Ocena ujemanja je povprečna vrednost vseh U_{rgb}, ki so izračunane za vsak stolpec po vsej širini stranice. V primeru, da primerjamo kosa vodoravno, oceno izračunamo enako, le da tokrat točke beremo v vsaki vrstici po višini stranice.



Slika 5.5: Primerjava točk po barvah.

5.3 Ujemanje svetilnosti točk ob robu kosov

Ujemanje svetilnosti izračunamo iz istih točk (slika 5.5), kot smo računali ujemanje RGB barvnih komponent. Ujemanje svetilnosti dveh točk je absolutna

razlika svetilnosti posameznih točk. Svetilnost izračunamo iz RGB s pomočjo formule (5.3) [11]:
Vrednosti rdeče, zelene in modre komponente, ki obsegajo 0-255 preračunamo v rang od 0 do 1. Poiščemo minimalno in maksimalno vrednost med RGB. Svetilnost L je polovica vsote minimalne in maksimalne komponente RGB:

$$L = \frac{minColor + maxColor}{2} \tag{5.3}$$

Ujemanje svetilnosti U_L:

$$U_L = \frac{\sum_{i=0}^{N} |L1_i - L2_i|}{N} \tag{5.4}$$

Kjer je N število parov točk, ki jih primerjamo, L1 svetilnost točke v prvem kosu, L2 pa svetilnost točke v drugem kosu.

5.4 Ujemanje intenzivnosti točk ob robu kosov

Ujemanje intenzivnosti enako kot prej izračunamo iz istih točk.
Intenzivnost S v točki izračunamo s formulama (5.5) in (5.6) [11]:
kadar je L > 0.5

$$S = \frac{maxColor - minColor}{maxColor + minColor} \tag{5.5}$$

sicer

$$S = \frac{maxColor - minColor}{2 - maxColor - minColor} \tag{5.6}$$

Ujemanje intenzivnosti U_S je povprečna vrednost absolutnih razlik intenzivnosti točk iz prvega in drugega kosa.

$$U_S = \frac{\sum_{i=0}^{N} |S1_i - S2_i|}{N} \tag{5.7}$$

Kjer je N število parov točk, ki jih primerjamo, S1 intenzivnost točke v prvem kosu, S2 pa intenzivnost točke v drugem kosu.

5.5 Ujemanje odtenkov točk ob robu kosov

Odtenek H v točki izračunamo z eno od formul (5.8), (5.9) ali (5.10) [11]:
če je R=maxColor:

$$H = \frac{G - B}{maxColor - minColor} \tag{5.8}$$

če je G=maxColor:

$$H = \frac{2 + B - R}{maxColor - minColor} \tag{5.9}$$

če pa je B=maxColor:

$$H = \frac{4 + R - G}{maxColor - minColor} \tag{5.10}$$

Ujemanje odtenka U_H je povprečna vrednost absolutnih razlik odtenkov točk iz prvega in drugega kosa.

$$U_H = \frac{\sum_{i=0}^{N} |H1_i - H2_i|}{N} \tag{5.11}$$

Kjer je N število parov točk, ki jih primerjamo, H1 odtenek točke v prvem kosu, H2 pa odtenek točke v drugem kosu.

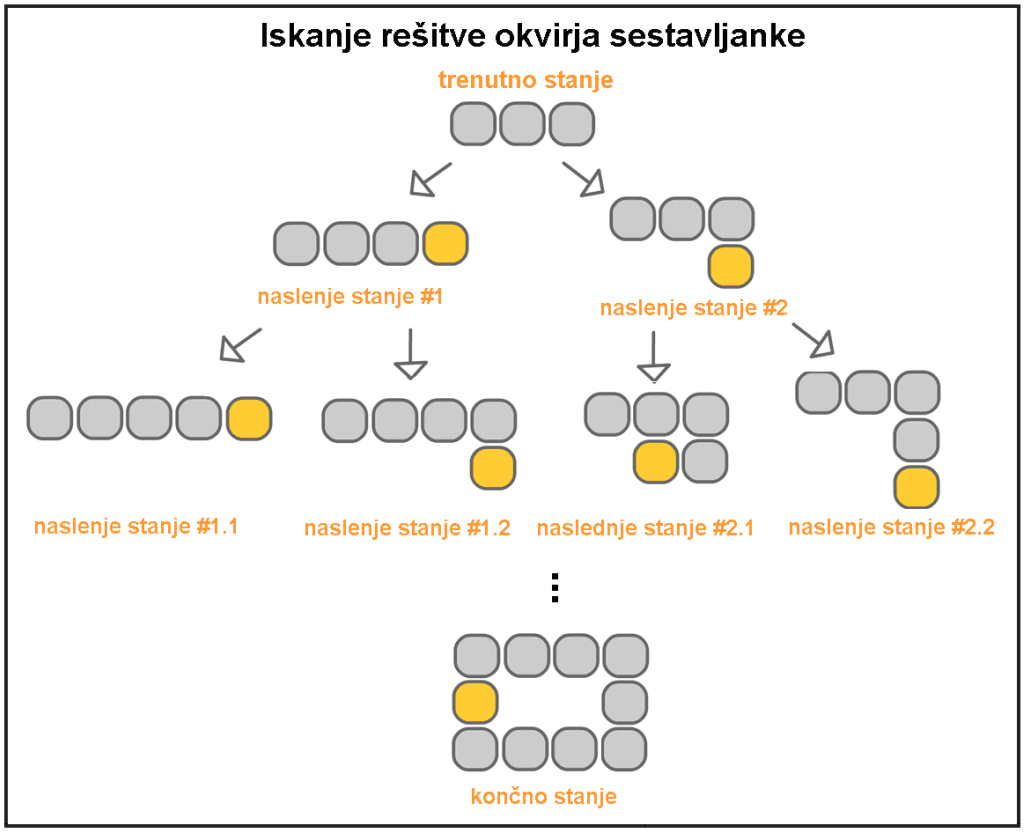
Poglavje 6

Zlaganje kosov

Zlaganje sestavljanke poteka v dveh korakih. Najprej zložimo okvir slike v smeri urinega kazalca, nato pa notranji del po vrsticah od zgoraj navzdol.

6.1 Zlaganje okvirja

Pri zlaganju okvirja kose med seboj primerjamo samo na eni stranici, kar nam da v primerjavi s kosi v notranjosti sestavljanke, kjer lahko vsak kos primerjamo vsaj na dveh stranicah, veliko slabšo točnost primerjave in zato večjo verjetnost napake. Veliko teh napak lahko odpravimo tako, da izračunamo najboljšo celotno rešitev okvirja, ne da samo zlagamo kos za kosom glede na trenutno najboljšo izbiro. Za vsak kos okvirja izračunamo ujemanje z vsemi ostalimi kosi, ki mu po obliki ustrezajo, in te ocene shranimo v urejeno tabelo, kjer je kos z najboljšo oceno na prvem mestu. Nato poiščemo rešitev okvirja tako, da ob vsak kos postavimo tistega z najboljšo oceno ujemanja, ki še ni bil uporabljen. Zlaganje začnemo v zgornjem levem kotu sestavljanke. Če nam uspe sestaviti vse kose v pravokotnik, kjer je zadnji kos na poziciji pod začetnim, potem to smatramo kot rešitev okvirja, sicer pa kos pred trenutnim zamenjamo z drugo najboljšo izbiro kosa pred njim. Zamenjave kosov v rekurzivni funkciji ponavljamo, dokler ne pridemo do pravilne rešitve ali pa izčrpamo vse kombinacije in ugotovimo, da je vsaj pri enem kosu prišlo do napake pri ugotavljanju lastnosti kosa (poglavje 4). S to metodo tudi rešimo težave z nesklenjenim okvirjem ali okvirjem v katerega ni mogoče postaviti vseh notranjih kosov, ostanejo pa napake kjer sta zamenjana dva ali več kosov enake oblike.

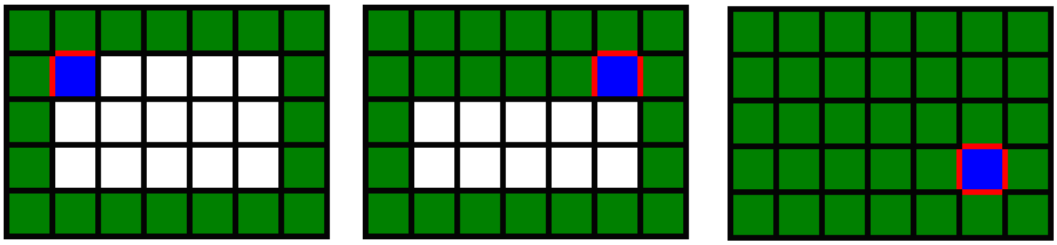


Slika 6.1: Iskanje rešitve okvirja sestavljanke.

6.2 Zlaganje notranjosti sestavljanke

Za zlaganje notranjosti sestavljanke ne iščemo najboljše celotne rešitve, ampak iščemo najbolj ustrezen kos za trenutno pozicijo. Če bi iskali najboljšo celotno rešitev, bi verjetno dobili boljše rezultate, vendar to zaradi časovne kompleksnosti ni najbolj praktično. Možnih kombinacij med notranjimi kosi je veliko več kot med robnimi, saj lahko te obračamo in v najslabšem primeru potrebujemo kar štiri primerjave med dvema kosoma. Najboljši kos za trenutno pozicijo dobimo s primerjanjem vseh kosov z vsemi sosednjimi (že postavljenimi). Kos lahko na določenem mestu primerjamo v večih orientacijah, nekateri po tipih stranic ustrezajo na samo en način, nekateri na dva, nekateri pa celo na vse štiri orientacije. Število primerjav za posamezen kos na iskanem mestu dobimo kot produkt ustreznih orientacij kosa in številom sosednjih že postavljenih kosov (slika 6.2). Ocena ujemanja kosa je vsota pri-

merjav z vsemi sosednjimi. Za vsak kos si je torej potrebno shraniti ocene za vsako možno orientacijo in nato med njimi poiskati tistega z najboljšo oceno ter ga postaviti na trenutno obravnavano mesto. Uporabnik nato s klikom na gumb *Naprej* potrdi, da se kos resnično prilega, program pa začne iskati kos za naslednjo mesto. Kadar je predlagan napačen kos, takrat uporabnik pritisne gumb *Napačen*, program pa predlaga naslednjega z najboljšo oceno prileganja, ki je lahko isti, vendar v drugačni orientaciji ali pa kak drug.



Slika 6.2: Število sosednjih kosov na določeni poziciji.

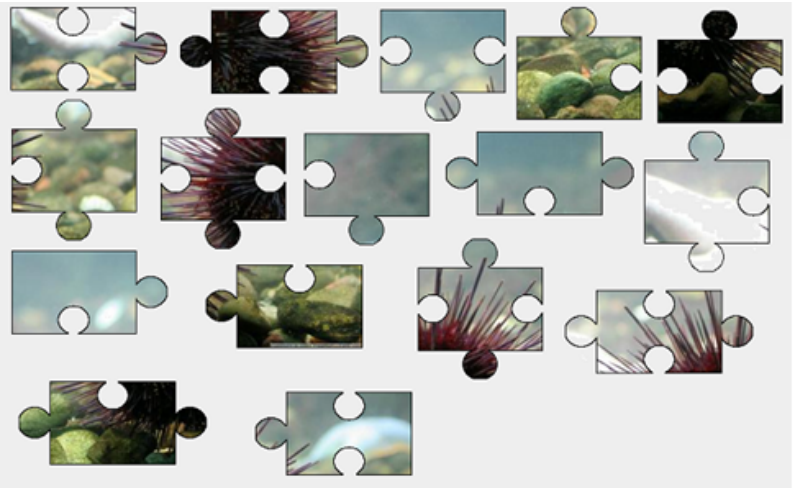
Ko nek kos postavimo, potem na naslednji poziciji izbiramo samo še med preostalimi, zato za vsak naslednji kos potrebujemo nekoliko manj časa za iskanje najustreznejšega. Na zadnjem mestu imamo samo še en kos, vendar moramo še vedno preveriti v kateri orientaciji se najbolj prilega vsem sosedom.

Poglavje 7

Primeri uporabe

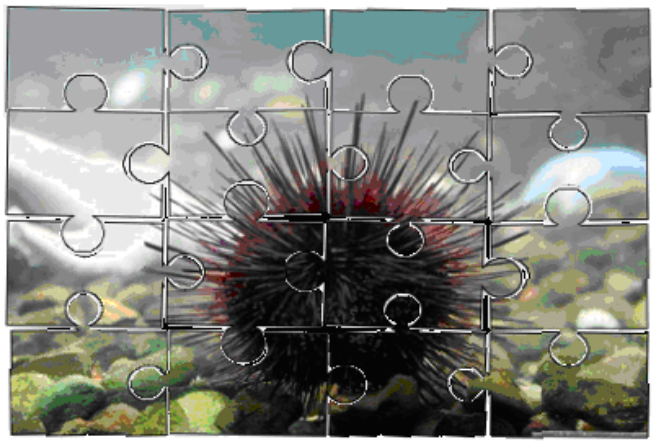
Delovanje algoritma za zlaganje sestavljank smo preizkusili na različnih primerih. Tako smo ugotavljali in beležili različne faktorje, ki vplivajo na uspešnost rešitve. Iz dobljenih rezultatov smo oblikovali priporočila za postavljanje koščkov na podlago, za doseganje optimalnih rezultatov, ki so opisana v poglavju 1 (Omejitve). V tem poglavju je opisanih nekaj primerov sestavljank, kot tudi težave, ki se lahko pojavijo zaradi neustreznih vhodnih slik. Na koncu je prikazana še časovna zahtevnost algoritma za različne sestavljanke.

Slika 7.1 predstavlja idealen vhod. Sestavljanka je generirana s pomočjo računalnika in je enostavna za prepoznavo. Slika je dimenzij 640 x 400 in ima 16 kosov, zložljivih v mrežo polj 4 x 4.



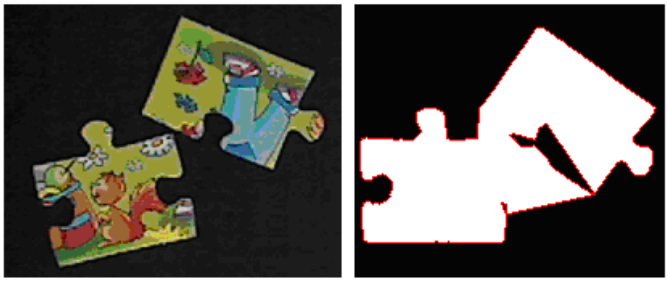
Slika 7.1: Vhodna slika računalniško generirane sestavljanke.

Ker je primer enostaven in brez šuma, pri prepoznavi kosov skoraj ni bilo napak. Vsi kosi so postavljeni vodoravno, zato tudi ni bilo težav pri določanju naklona kosov. Ker imajo kosi dve krajši in dve daljši stranici, smo lahko že na podlagi tega izločili precej kandidatov pri primerjanju kosov. Kosi so podobnih oblik, zato so bile barvne informacije pri računanju ujemanja kosov pomembnejše od ujemanja oblike.



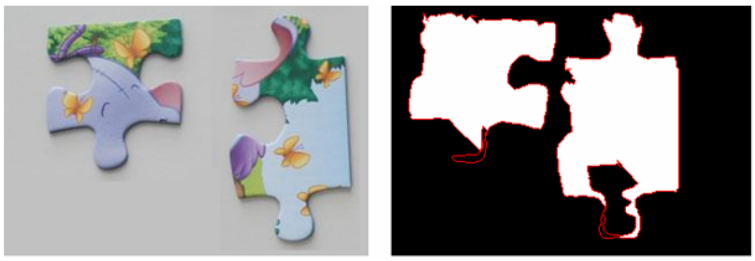
Slika 7.2: Izhodna slika računalniško generirane sestavljanke.

Naslednji primer (slika 7.3) prikazuje rezultat segmentacije slike, kjer sta dva kosa postavljena preblizu eden drugemu. Program ni uspel razločiti meje med njima in ju je obravnaval kot en kos. Na sliki obrisov kosov opazimo belo površino tudi kjer ni nobenega kosa. Do zgoraj opisanih nepredvidljivih rezultatov pride, kadar program poskuša zapolniti vrzeli v obrisu, kjer dva kosa obravnava kot enega.



Slika 7.3: Primer kosov postavljenih preveč skupaj in njun obris.

Naslednji primer (slika 7.4) prikazuje napako pri detekciji robov, kjer ozadje ni dovolj kontrastno glede na kose sestavljanke. Na levi strani obeh kosov rob ni bil dovolj izrazit, zato so bili zaznani napačni robovi, kar posledično onemogoči nadaljevanje zlaganja take sestavljanke. Prav tako so nastale napake na spodnji stranici, kjer je program rob zaznal, vendar preveč nepopolno, da bi bil rezultat uporaben v nadaljnih korakih.



Slika 7.4: Primer premalo kontrastnih kosov glede na podlago.

Naslednji primer (slika 7.5) predstavlja sestavljanko iz dvanajstih kosov, zajeto s spletno kamero. Pri postavljanju kosov smo pazili na prej opisani težavi, da so drug od drugega dovolj oddaljeni, postavljeni pa so na črno podlago, ki se kontrastno precej razlikuje od sestavljanke, katera je svetlejših barv.

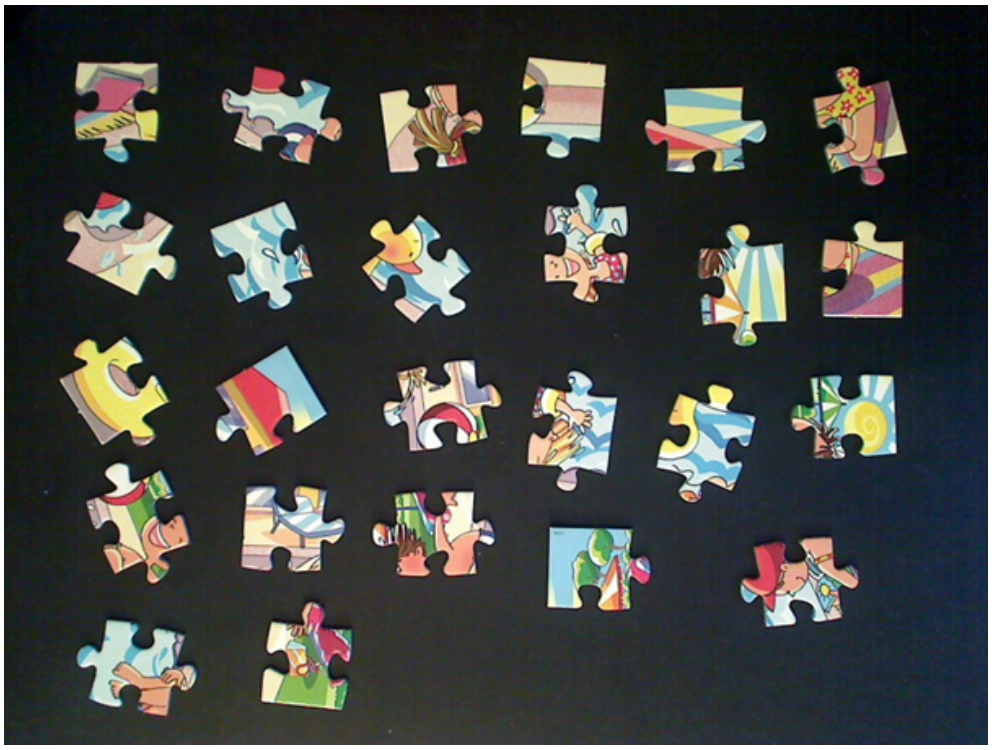
Slika 7.5: Kosi enostavne sestavljanke zajeti s spletno kamero.

Algoritem je uspešno prepoznal vse kose in jih tudi pravilno zložil skupaj (slika 7.6). Do manjše napake je prišlo pri določanju zgornjega desnega vogala kosa, v drugi vrstici drugega stolpca, zato je ta kos postavljen pod nekoliko napačnim kotom, med njim in zgornjim sosednjim kosom pa je nekaj praznega prostora.

Slika 7.6: Izhodna slika enostavne sestavljanke.

Slika 7.8 prikazuje večjo sestavljanko s 25 kosi, od tega jih je 16 v okvirju in 9 v notranjosti sestavljanke. Vhodna slika je bila velikosti 800 x 600, zato je tudi obdelava počasnejša. Vsi kosi so bili pravilno izločeni iz vhodne slike, pri zlaganju pa je program naredil napako pri enem kosu v okvirju (slika 7.7). Po povratni informaciji uporabnika o napačnem kosu, so bili vsi naslednji kosi postavljeni na pravo mesto (slika 7.9).

Slika 7.7: Napačno postavljen kos.



Slika 7.8: Vhodna slika sestavljanke s 25 kosi.

Slika 7.9: Izhodna slika sestavljanke s 25 kosi.

Tabela 7.1 prikazuje izvajalne čase posameznih korakov algoritma za pet sestavljank, z različnim številom kosov. Prvi del tabele vsebuje podatke o vhodnih slikah, katerih resolucija je 640x480 pik. Drugi del pa podatke o slikah z resolucijo 800x600. Opazimo, da se pri 20% povečanju resolucije, časovna zahtevnost poveča za približno 35 do 45 odstotkov. Prav tako je iz tabele razvidno, da večje število kosov ni nujno časovno zahtevnejše. Za zlaganje sestavljanke s 16 kosi, smo porabili več časa kot za sestavljanko z 20 kosi, razlog za to pa je v obliki kosov, saj je imela manjša sestavljanka kvadratne kose in za razliko od večje s pravokotnimi kosi, pri primerjavah kosov ni bilo moč izločiti zgolj na podlagi dolžin stranic.

640x480	12 kosov	16 kosov	20 kosov	25 kosov	35 kosov
Segmentacija	2.20	2.64	2.29	2.32	5.37
Določanje lastnosti	1.76	1.80	1.93	2.46	3.28
Primerjanje kosov	2.71	6.98	5.35	14.62	23.12
Skupaj	6.67	11.42	9.57	19.40	31.77
800x600	12 kosov	16 kosov	20 kosov	25 kosov	35 kosov
Segmentacija	2.89	3.18	3.31	3.15	7.20
Določanje lastnosti	2.54	2.62	2.78	2.81	3.85
Primerjanje kosov	3.79	9.70	7.43	20.34	34.12
Skupaj	9.22	15.50	13.52	26.30	45.17

Tabela 7.1: Izvajalni časi (v sekundah) posameznih korakov algoritma.

Zaključek in nadaljnje delo

To diplomsko delo predstavi računalniško rešitev problema zlaganja sestavljank. Poskusi so bili izvedeni s slikami sestavljank iz resničnega sveta. Rezultati poskusov kažejo, da je naš način uspešen in bi bil lahko uporaben v inteligentnih sistemih za robotsko sestavljanje. Z našo metodo smo kategorizirali kose sestavljank v 18 vzorcev. To je močno skrajšalo čas računanja in poenostavilo postopek. Metoda vključuje ekstrakcijo kosa sestavljanke, določanje vogalnih točk in določanje tipa kosa v prvi fazi ter primerjave oblike robov, primerjave barvnih informacij kosov in združevanje le-teh v sestavljeno sliko v drugi fazi.

Različni avtorji (glej poglavje 1), predlagajo različne metode za računalniško zlaganje sestavljank. Kot vsaka metoda avtomatskega zlaganja sestavljank, ima tudi ta svoje prednosti in slabosti pred ostalimi, saj deluje samo za navadne sestavljanke s kosi, ki imajo štiri stranice, poleg tega pa ni mogoče sestaviti samo dela sestavljanke, ampak mora vhodna slika vsebovati vse kose, ki ji pripadajo.

V nadaljnjem razvoju bi bilo potrebno poenostaviti uporabnikovo interakcijo z vmesnikom, kjer bi program lahko samodejno zaznal da se kos, ki ga je predlagal, ne prilega. Kadar bi uporabnik določen kos vrnil nazaj med nesestavljene kose, bi lahko program preko spletne kamere zaznal, da se kos na predlagano mesto ne prilega. V nasprotnem primeru pa bi predvideval, da kos ustreza in bi nadaljeval z naslednjim. Razmisliti bi veljalo tudi o boljši metodi za določanje robov med kosi sestavljanke in ozadjem. Naša metoda deluje pravilno samo v primeru, da je meja med kosi in ozadjem dokaj jasno določena. Program bi lahko priredili za uporabo na mobilnih napravah (pametni telefoni, tablični računalniki, dlančniki, ...), kjer bi dobil bolj praktično uporabno vrednost saj ima večina takih naprav že vgrajene digitalne kamere s pomočjo katerih bi uporabnik brez težav zajel sliko nezloženih kosov sestavljanke.

Slike

Tabele

Literatura

[1] H. C. da Gama Leitao and J. Stolfi. *Automatic reassembly of irregular fragments*, Tech. Report IC-98-06, Univ. of Campinas, 1998.

[2] H. C. da Gama Leitao. *Information Contents of Fracture Lines*, Tech. Report, Univ. of Campinas, 1999.

[3] (2011) M. Levoy. The digital Forma Urbis Romae project. Dostopno na: http://www.graphics.stanford.edu/projects/forma-urbis/

[4] (2011) M. Levoy. The digital Michelangelo project. Dostopno na: http://www.graphics.stanford.edu/projects/mich/

[5] G.M. Radack and N.I. Badler. *Jigsaw puzzle matching using a boundary-centered polar encoding*, Computer Graphics and Image Processing, str. 1-17, 1982.

[6] H. Wolfson, E. Schonberg, A. Kalvin and Y. Lamdan. *Solving jigsaw puzzles by computer*, Annals of Operations Research, str. 51-64, 1988.

[7] (2011) Graycale. Dostopno na: http://en.wikipedia.org/wiki/Greyscale

[8] J. Canny, *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no.6, pp. 679-698, 1986.

[9] (2011) Canny edge detector. Dostopno na: http://en.wikipedia.org/wiki/Canny_edge_detector

[10] (2011) The HSL Color Model. Dostopno na: http://www.academictutorials.com/graphics/graphics-the-hsl-color-model.asp

[11] RGB to HSL. Dostopno na:
http://130.113.54.154/ monger/hsl-rgb.html

[12] Freeman, H., Garder, L. *Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition*, str. 118-127, 1964.

[13] Webster, R.W., LaFollette, P.S., Stafford, R.L., *Isthmus critical points for solving jigsaw puzzles in computer vision*, str. 1271-1278, 1991.

[14] Kosiba, D.A., Devaux, P.M., Balasubramanian, S., Gandhi, T.L., Kasturi, R., *An automatic jigsaw puzzle solver*, Proc. 12th IAPRInternat. Conf. on Pattern Recognition, Part I, October 9-13, Jerusalem, Israel, 1994.

[15] Yao, F.H., Shao, G.F., Tamaki, A., Kato, K., *Recovery of connection relationships among two-dimensional objects*, str. 746-761, 1997.

[16] Chung, M.G., Fleck, M.M., Forsyth, D., *Jigsaw puzzle solver using shape and color*, str. 12-16, 1998.

[17] Glassner, A., *Putting the pieces together*, Str. 76-86, 2002.

[18] Kennedy, T., *Jigsaw puzzle solver usin computer vision*, str. 1-24, 1998.

[19] Cho, T.S., *Avidan, S., Freeman, W., A probablistic image jigsaw puzzle solver*, 2010.