SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

AUTONOMNA ROBOTSKA MONTAŽA PRIMJENOM STROJNOG VIDA

MAGISTARSKI RAD

MILJENKO HRMAN

ZAGREB, 2002.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

AUTONOMNA ROBOTSKA MONTAŽA PRIMJENOM STROJNOG VIDA

MAGISTARSKI RAD

Mentor:
Prof. dr.sc. Bojan JERBIĆ

Miljenko HRMAN, dipl.inž

ZAGREB, 2002.

PODACI ZA BIBLIOGRAFSKU KARTICU

UDK: 621.757:007.52:681.5:681.327.12

Ključne riječi: robotska montaža, automatska montaža, strojni vid, vizijski sustavi, prepoznavanje objekata, nesređena okolina robota, upravljanje informacijama

Znanstveno područje: TEHNIČKE ZNANOSTI

Znanstveno polje: STROJARSTVO

Institucija u kojoj je rad izrađen: Fakultet strojarstva i brodogradnje, Sveučilište u Zagrebu

Mentor rada: Prof. dr.sc. Bojan JERBIĆ

Broj stranica: 136

Broj slika: 78

Broj tablica: 3

Broj korištenih bibliografskih jedinica: 51

Datum obrane: 18. Listopad 2002.

Povjerenstvo: Prof. dr.sc. Božo Vranješ

Prof. dr.sc. Bojan Jerbić

Prof. dr.sc. Slobodan Ribarić

Institucija u kojoj je rad pohranjen: Fakultet strojarstva i brodogradnje, Zagreb

ZAHVALA

Zahvaljujem mentoru, prof. dr.sc. Bojanu Jerbiću na savjetima, korisnim raspravama i potpori tijekom izrade ovog rada.

Zahvaljujem na pomoći i ostalim djelatnicima Katedre za projektiranje izradbenih i montažnih sustava, posebno prof. dr.sc. Boži Vranješu.

Na kraju, posebno zahvaljujem svojoj obitelji na podrazumijevanju i potpori.

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

PREDGOVOR

U ovom je radu predstavljen dio istraživačkog rada posvećenog unapređivanju tehnologije montaže i montažnih procesa, te je prikaz iskustva autora u primjeni suvremenih inženjerskih alata i metoda pri razvoju automatskih montažnih sustava.

Rad je dio cjelokupnih istraživanja unutar projekta "Inteligentno strojno sklapanje" financiranog od strane Ministarstva znanosti i tehnologije Republike Hrvatske, kojemu je cilj ostvarenje inteligentnog fleksibilnog automatskog montažnog sustava, uz potpunu integraciju oblikovanja proizvoda, projektiranja montažnoga sustava i izvođenja sklapanja. Ostvarenje sustava uključuje iznalaženje učinkovitih pristupa i metoda, te njihovu formalizaciju i računalnu implementaciju, a za automatsko oblikovanje montažnoga sustava, inteligentno planiranje i izvođenje sklapanja.

SAŽETAK

U ovom je radu predstavljen robotski montažni sustav kojeg odlikuje sposobnost autonomnog djelovanja u nesređenim radnim uvjetima. Osnovni preduvjet ostvarivanja autonomnosti, percepcija radne okoline, ispunjen je primjenom strojnog vida. Strojni vid je tehnologija kojom tehnički sustavi ostvaruju vizualnu percepciju svoje okoline. Sustav strojnog vida tvore elektroničke i optičke komponente te odgovarajući algoritmi za obradu vizualnih informacija.

U prvom dijelu rada je opisana struktura sustava strojnog vida kao i pojedini elementi sustava.

Drugi dio rada predstavlja koncept fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj radnoj okolini. Sustav je integriran na osnovi višestrukog korištenja inženjerskih znanja, strukturiranih u vidu CAD modela proizvoda. Ta se znanja nadopunjuju informacijama iz sustava strojnog vida, čime se upotpunjuje ukupna spoznaja o proizvodu i procesu, omogućavajući autonomno izvođenje montaže.

Sukladno predstavljenom konceptu fleksibilnog integriranog sustava, realizirana je robotska stanica i odgovarajuća programska podrška sustavu strojnog vida te upravljački program robota.

Primjena i verifikacija sustava je prikazana na montaži jednostavnog sklopa.

Ključne riječi:

robotska montaža, automatska montaža, strojni vid, vizijski sustavi, prepoznavanje objekata, nesređena okolina robota, upravljanje informacijama.

SUMMARY

This thesis presents an autonomous robotic assembly system capable of operating in disordered working environment. The fundamental prerequisite for the autonomous robot behavior, that is the perception of work environment, is accomplished by means of machine vision system. The machine vision is a technology that provides the visual perception of the technical systems. This technology consists of the electronic and optical hardware components and the corresponding algorithms for visual information analysis.

The first part of the thesis describes the structure of machine vision system and its components.

The model of flexible integrated assembly system for automatic assembly process in disordered environment is presented in the second part of the thesis. The basic idea of the concept is to maximize the use of the CAD data and the integration of the accompanying process. CAD model embodies the description of the product geometry, which drives the parameters of the production and assembly processes. The rest of the information required for autonomous execution of assembly process is complemented by the machine vision system.

In accordance to the presented model, the robotic assembly cell is realized and the required machine vision software and robot control program are developed.

The robotic system and the developed software are verified by assembling a simple product.

Key words:

robotic assembly, automatic assembly, machine vision, vision system, object recognition, disordered robot environment, product data management.

1 UVOD

1.1. Motivacija

Strojni vid je tehnologija čija je zadaća ostvarivanje vizualne percepcije tehničkih sustava radi unapređenja njihovih funkcionalnih sposobnosti, u prvom redu povećavanja radne autonomije strojeva. Pomoću strojnoga vida, ali i zahvaljujući drugim dostignućima visokih tehnologija (npr. informatika, umjetna inteligencija, novi materijali i postupci, nanotehnologija itd.), tehničke tvorevine sve češće imaju osobine donekle usporedive s ljudskim sposobnostima.

Ova uzbudljiva tehnologija doprinosi novoj kvaliteti u proizvodnim djelatnostima. Naime, roboti predstavljaju uobičajenu opremu proizvodnih sustava još od 1980-tih godina. No, oni su ograničeni na obavljanje repetitivnih zadaća u strogo kontroliranim industrijskim uvjetima. Strojni vid je tehnologija koja omogućuje da roboti djeluju u uvjetima u kakvima je to ranije bilo nemoguće, u uvjetima gdje nije potpuno kontroliran i određen svaki element sustava, u uvjetima u kojima smo sami navikli djelovati.

U domeni proizvodnih djelatnosti montaža se javlja kao izrazito složena djelatnost, jer iziskuje primjenu složenih motoričkih operacija u sprezi s inteligencijom i adaptivnošću. Dakle, iziskuje radne sposobnosti imanentne prvenstveno ljudima, a ne strojevima.

Funkcionalnost automatskih montažnih sustava, čije upravljanje je izvedeno na uobičajen ili klasičan način, je direktno ovisno o obliku proizvoda. Čak male promjene oblika proizvoda uvjetuju velike zahvate u sustavu. Nadalje, automatski montažni sustavi kao i ostali klasični automatski/robotski sustavi zahtijevaju strogo kontroliranu radnu okolinu. To najčešće podrazumijeva da se pozicije i orijentacije kako ugradbenih elementa, tako i elemenata sustava fizički ograničavaju.

Usprkos svim poteškoćama, automatiziranje montaže predstavlja tehnološki imperativ budući da u suvremenim društvima postoji potreba za ukidanjem repetitivnih i monotonih poslova karakterističnih za ručnu montažu. Osim toga, troškovi montaže uobičajeno predstavljaju znatni udio u ukupnim troškovima proizvoda.

Strojnim vidom se pomoću vizualne percepcije dinamički određuje stanje sustava te se tako ukida nužnost fizičkog određivanja, odnosno ograničavanja pozicija ugradbenih elemenata. Time se u velikoj mjeri doprinosi fleksibilnosti sustava.

Stoga je cilj ovoga rada razvoj modela automatskog/robotskog sustava za sklapanje u nesređenoj okolini s posebnim naglaskom na realizaciju i primjenu sustava strojnog vida kojim će se ostvarivati vizualna percepcija radne okoline sustava. Vizualna percepcija radne okoline u ovom slučaju uključuje određivanje pozicija i orijentacija ugradbenih dijelova koji se trebaju sklopiti prema planu montaže. Budući da se pozicije i orijentacije ugradbenih dijelova određuju vizualnom percepcijom, nema potrebe za uređajima poput vibracijskih dodavača ili magazina s prethodno sređenim dijelovima. Takav sustav omogućuje da industrijski robot, kao jedini izvršni član sustava, može izvršiti (gotovo) sve montažne operacije.

Naravno, sustavom strojnog vida moguće je dobiti i čitav niz drugih informacija relevantnih za automatski/robotski montažni sustav. Položaji montažnih uređaja u sustavu, odnosno trenutna konfiguracija sustava, su svakako zanimljiva informacija pomoću koje se mogu izbjeći kolizije u sustavu ili čak eventualne ozljede operatera. Ipak sustav strojnog vida, prikazan u ovome radu, je ograničen na određivanje pozicija i orijentacija ugradbenih dijelova.

Robotski sustav za sklapanje u nesređenoj okolini treba razviti u duhu koncepta simultanog inženjerstva. Simultano inženjerstvo je sistematski pristup (informacijski) integriranom, istovremenom oblikovanju proizvoda, projektiranju procesa izrade i izrade proizvoda. U ovom slučaju to znači da moraju biti omogućena informacijska sučelja prema sustavima kojima se realiziraju ostale faze razvoja proizvoda. Najvažnije komunikacije su usmjerene prema sustavu za oblikovanje proizvoda i sustavu za projektiranje procesa sklapanja.

Oblikovanjem proizvoda se definiraju i strukturiraju informacije o oblikovnim značajkama koje može iskoristiti sustav strojnog vida za prepoznavanje ugradbenih elemenata.

Projektiranjem procesa sklapanja definiraju se operacije sklapanja. Operacije sklapanja se ostvaruju gibanjima montažnih uređaja. U slučaju robotskog sustava za sklapanje u nesređenoj okolini to znači da su parametri upravljačkog programa robota određeni operacijama sklapanja.

1.2. Struktura rada

U drugom poglavlju je ukratko opisana montažna tehnologija, i to ručna i automatizirana montaža te povezanost strojnog vida i inteligentne montaže.

Osnovni elementi i struktura strojnog vida su opisani u trećem poglavlju. Naime, sustavi strojnog vida su vrlo složeni i sastoje se od optičkih i elektroničkih uređaja kojima se upravlja nizom metoda/algoritama koji na različitim razinama analize vizualnih informacija doprinose konačnom rješenju.

U četvrtom je poglavlju opisana poluvodička kamera. Kamera je prvi opisani element sustava strojnog vida budući da se njome kreira digitalna slika koja je primarni nositelj vizualnih informacija.

U petom poglavlju dan je kratak pregled algoritama sustava strojnog vida. Osnovna podjela je podjela na primarne algoritme i algoritme visoke razine.

Jedan od zadataka algoritama primarne razine, posebno interesantan kod primjene strojnog vida u montaži, je segmentacija. Segmentacija je postupak kojim se slika, kao nositelj vizualnih informacija, dijeli na područja koja bi mogla odgovarati objektima, a rezultati podjele se mogu opisati analitički odnosno matematički. Algoritmi za segmentaciju su opisani u šestom poglavlju.

Prepoznavanje objekata je zadatak sustava strojnog vida koji se povezuje s algoritmima visoke razine. Na temelju rezultata algoritama primarne razine i znanja o objektima na slici identificiraju se prisutni objekti i određuju njihove pozicije. Algoritmi za prepoznavanje su opisani u sedmom poglavlju.

U osmom poglavlju opisan je sustav za automatsku montažu primjenom strojnog vida čiji su osnovni elementi šest-osni industrijski robot i sustav strojnog vida. Razvijeni sustav je dio predstavljenog koncepta fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj okolini. Osnovna ideja predstavljenog koncepta je minimaliziranje informacijske entropije na temelju višestrukog korištenja znanja o proizvodu, strukturiranog, konceptualiziranog i operacionaliziranog u formi CAD modela. Informacije prikupljene vizualnom percepcijom nadopunjuju se s informacijama dobivenim iz CAD modela. Na temelju tih informacija upravlja se sustavom. Razvijena je i prezentirana odgovarajuća programska podrška za upravljanje robotom kao i programska podrška strojnog vida.

U devetom poglavlju prikazana je primjena i verifikacija razvijenog sustava na primjeru sklapanja jednog jednostavnog sklopa. Opisana je cjelokupna procedura koja uključuje pripremu parametara programske podrške i izvođenje operacija sklapanja.

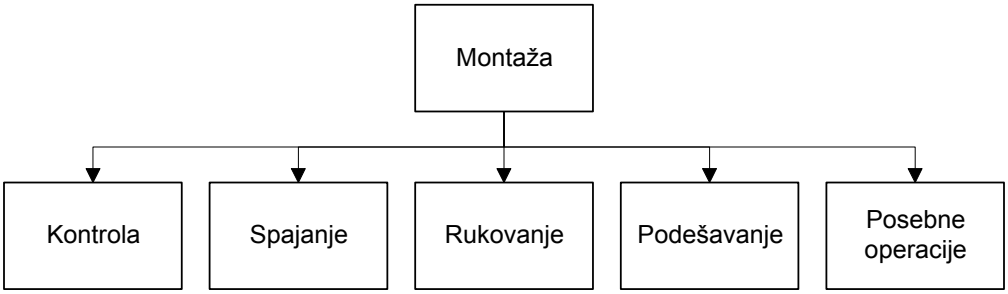
Rad završava sa desetim, zaključnim, poglavljem u kojem se analiziraju rezultati rada te predlažu pravci daljnjih istraživanja.

2 MONTAŽA

Montaža, ili sklapanje, jest svaka djelatnost kojoj je cilj spajanje dvaju ili više objekata u cjelinu, određene namjene. Zastupljena je u svim ljudskim djelatnostima, od industrije, do kućanstva [1]. Industrijski proizvodi u pravilu se sastoje od više ugradbenih dijelova, izrađenih u različitim vremenskim periodima različitim tehnologijama. Zadaća je montaže da se od ugradbenih dijelova izgradi proizvod višeg stupnja složenosti s unaprijed određenom funkcijom [2].

Sukladno gornjim definicijama, montaža je jedna od najstarijih tehnologija. Sjekira od prije nekoliko desetaka tisuća godina, primjer je montaže u kojemu je spoj drvenog drška s kamenom izveden utiskivanjem drška u pogodnu rupu u kamenu ili pak vezivanjem drška uz kamen nitima biljnog ili životinjskog porijekla [3]. Montaža i montažni postupci razvijali su se sukladno s razvojem civilizacije. Postupci sastavljanja metalnih tvorevina na primjer, razvijali su se paralelno s razvojem prerade metala. Pri proizvodnji oružja u srednjem vijeku koristili su se gotovo svi danas poznati postupci spajanja preoblikovanjem. Razvojem manufakture, krajem srednjeg vijeka, postavljeni su temelji industrijskoj montaži [1]. Zajedno sa svakom velikom industrijskom revolucijom razvijala se i montaža, pa tako i s najnovijom revolucijom potaknutom informatičkom tehnologijom.

Montažni proces jest odvijanje djelatnosti potrebnih za sklapanje proizvoda, prema određenim zakonitostima (ekonomskim, tehnološkim, sociološkim, ekološkim...). Montažni proces je slijed uzastopnih i/ili usporedno povezanih djelatnosti - operacija, koje izvode ljudi i/ili automati, sa svrhom izrade tvorevine unaprijed definirane strukture [1]. U svakom montažnom procesu može se uočiti nekoliko osnovnih operacija. DIN 8593 (Sl.-1) analizira montažni proces i opisuje operacije koje on obuhvaća [4].



Sl.-1. Operacije montažnog procesa.

Spajanje je postupak kojim se ugradbeni elementi dovode u međusobni odnos i osiguravaju od rastavljanja. Odnos - spoj, ostvaruje se preko ploha spajanja. Svaki spoj definiran je sa dva odnosa: geometrijskim i energetskim. Geometrijski odnos definira prostorni raspored ugradbenih elemenata, a energetski odnos određuje opterećenje pod kojim je osigurana funkcija spoja. Važna značajka za opisivanje postupka spajanja jest gibanje pri spajanju. Gibanje se ostvaruje primjenom sile ili momenta, ljudskom rukom ili alatom.

Gibanju pri spajanju prethodi postupak rukovanja kojim se ugradbeni elementi dovode u položaj za spajanje. Sve djelatnosti kojima se pripremaju ugradbeni elementi spadaju u operaciju rukovanja koja je podfunkcija toka materijala na radnom mjestu. Rukovanje završava, a spajanje otpočinje u trenutku kada ugradbeni element izgubi najmanje jedan stupanj slobode gibanja. Nakon uspostave dodira između ugradbenih elemenata, gibanje je određeno oblikom i geometrijskim rasporedom ploha spajanja ugradbenih elemenata.

Kontrola je operacija kojom se provjerava stanje, svojstava, kakvoća i funkcionalnosti ugradbenih elemenata te ispravnosti prethodno izvršenih operacija[1].

Podešavanje obuhvaća djelatnosti za otklanjanje izradbeno-tehničkih odstupanja s ciljem da se postigne zadana funkcionalna tolerancija sklopa[2].

Posebne su operacije sve funkcije koje osiguravaju postizanje funkcionalnosti proizvoda[1].

Svako unapređenje montažnog procesa je izvor značajnih smanjenja proizvodnih troškova. Vremenski udio montaže u proizvodnom procesu iznosi od 40 do 60%, dok je troškovni udio od 20 do 50% [5][6].

U osnovi razlikujemo dvije vrste montaže, ručnu i automatsku montažu. Montaža je jedna od rijetkih tehnologija današnjice u kojoj ručni rad ima tako značajnu ulogu. Razlog tome je činjenica da montažni proces bitno ovisi o proizvodu (broj dijelova, veličini, obliku). Rješenje automatizacije jednog montažnog procesa vrlo teško može iskoristiti kod drugog proizvoda. Uz to, montažne su operacije višeg stupnja kompleksnosti od izradbenih, poglavito u smislu zamjene složenog ljudskog rada. Ove okolnosti dodatno su otežane danas iznimno promjenjivim zahtjevima tržišta (porast varijanti i promjenjivost količina proizvoda, skraćeni rokovi isporuka)[3].

[1] Budući da je montaža završna aktivnost u proizvodnji, svi propusti, pogreške i nedostaci prethodnih faza proizvodnje kumuliraju se u njoj. Zbog toga je kontrola vrlo važna i vrlo zastupljena operacija u montažnom procesu.

[2] Operacija podešavanja ne postoji kod automatskih procesa.

2.1. Automatska montaža

Usprkos svim poteškoćama, automatizacija montaže predstavlja tehnološki imperativ, bez obzira na već zastarjeli stav da je montaža dio proizvodnog procesa bez posebnog značaja. Taj stav često proizlazi iz činjenice da se i najkompleksniji proizvodi mogu sklopiti ručno, odnosno uporabom jednostavnih alata. Već spomenuti udio troškova montaže u troškovima proizvoda dovoljan je razlog za razmatranje automatizacije. Trend rasta cijena ljudskog rada, a time i udjela troškova ručne montaže, dodatno naglašava prednosti automatskih montažnih sustava.

U suvremenim društvima postoji potreba za ukidanjem repetitivnih i monotonih poslova karakterističnih za ručne montažne linije. Izvođenje montaže u nerazvijenim zemljama s niskom cijenom ljudskog rada nije trajno rješenje problema.

Kada se govori o prednostima automatskih montažnih sustava mora se spomenuti i povećanje kvalitete proizvoda. Uvjet je automatizacije sklapanja visoka kakvoća dijelova. Smatra se da je automatizacija montaže moguća samo ako dijelovi imaju do 2% škarta [7].

Automatizacija uvijek zahtijeva kapitalne investicije koje se moraju amortizirati uštedama pri sklapanju. Da bi se uvođenje automatizacije moglo opravdati potreban je veliki obujam proizvodnje. No, je li potreban i veliki broj identičnih proizvoda? Zahvaljujući fleksibilnoj automatizaciji, nije nužno da proizvodi budu identični. Naime, postoje dva osnovna tipa automatskih sustava, kruti i fleksibilni [8].

Pod krutom se automatizacijom podrazumijeva klasična odnosno mehanička automatizacija, tipična za 40-te i 50-te godine 20. stoljeća. U montažnim sustavima strojevi u kojima je automatizacija ostvarena mehanički, krivuljnom pločom, zovu se jednonamjenski montažni automati. Iako predstavljaju stariji model automatizacije, oni još uvijek nisu zastarjeli. Montažni sustavi opremljeni jednonamjenskim montažnim automatima imaju mnogo veću proizvodnost od fleksibilnih montažnih sustava. Naime, ako je potrebno sklopiti veliku količinu jednog proizvoda, oni su još uvijek pravo rješenje. Njihova je mana nefleksibilnost. Vrlo male varijacije oblika ili strukture proizvoda zahtijevaju velike promjene uređaja za sklapanje. U novije vrijeme upravljanje krivuljnim pločama se zamjenjuje PLC[3]-om čime se donekle povećava fleksibilnost jednonamjenskih montažnih automata.

Osnovna značajka fleksibilne automatizacije jest da je programibilna, a stoga i reprogramibilna. Fleksibilnost montažnog sustava se očituje u:

- primjenjivosti sustava za različite proizvode,
- sposobnosti sustava da se nosi s nepredviđenim okolnostima.

Robot, odnosno industrijski robot je osnovni element fleksibilne automatizacije. Postoji mnogo definicija kojima se nastoji opisati robot. Definicija *Robot Institute of America* ignorira značajke robota vezane uz inteligenciju i fleksibilnost, već robot opisuje kao manipulator. Definicija glasi:

[3] PLC (Eng.) Programmable Logical Controller – Programibilni Logički Upravljač

Robot je programibilni, višefunkcijski manipulator oblikovan da pomiče materijal, dijelove, alate ili specijalnu opremu kroz različite programirane pokrete u izvedbi različitih zadaća [9].

Neki autori u znatno većoj mjeri uvažavaju inteligentne i fleksibilne značajke (upravljanja) robota. Tony Owen daje sljedeću definiciju robota:

Robot je reprogramibilni manipulator opremljen s integralnim osjetilnim perceptorima koji mu omogućavaju da otkrije promjene u radnom prostoru ili uvjetima rada i sa svojom vlastitom sposobnošću donošenja odluka, odluči o narednim aktivnostima [6].

Većina robota instaliranih u proizvodnim sustavima ne posjeduje obilježja opisane u drugoj definiciji robota, nego zahtijevaju potpuno kontroliranu, deterministički određenu okolinu. Postavlja se pitanje je li budućnost primjene robota u daljnjem prilagođivanju svijeta njihovim potrebama? Izvjesnijom se čini opcija primjene, odnosno razvoja inteligentnih i fleksibilnih robota sposobnih da uspješno djeluju u uvjetima kontroliranog kaosa na koji smo navikli.

2.2. Inteligentno sklapanje i strojni vid

Roboti korišteni u montažnim sustavima također (uglavnom) zahtijevaju potpuno sređenu okolinu. Sređena okolina je potrebna jer upravljački program pretpostavlja unaprijed određene pozicije i orijentacije ugradbenih elemenata i sklopa. Osim toga, potrebno je unaprijed točno opisati okolinu robota kako ne bi došlo do prostornih kolizija (sudara) u sustavu. Glavni nedostaci ovakvog pristupa su:

- neupotrebljivost upravljačkog programa za slične montažne zadaće,
- osjetljivost sustava na poremećaje,
- potrebni dodatni uređaji za osiguranje sređene okoline (cijena sustava).

Ukoliko nije moguće postići sređeno stanje okoline, ili je to ekonomski neopravdano, nužno se provodi ručna montaža.

Naravno, konstantni je cilj unapređenja kako proizvodnje i tehnologije, tako i napretka ljudske civilizacije uopće, zamjena ljudskog rada radom strojeva. Čovjekovo izravno djelovanje/upravljanje je zasada nezamjenjivo u nepredvidivim (nedeterminističkim) situacijama. Njegova inteligencija, sposobnost učenja, ljudski vid te koordinacija i vještina pokreta (i djelovanja općenito) nadmašuje sve postojeće tehničke sustave.

Nadomještanje i nadopunjavanje ljudskog rada u nepredvidivim radnim uvjetima iziskuje tehničke sustave u kojima su implementirane ljudske sposobnosti. Ljudska sposobnost vizualne percepcije je jedna od presudnih ljudskih sposobnosti. Ona je mnogo više od sposobnosti identifikacije i lociranja objekata u prostoru. Vid je jedan od glavnih mehanizama kojim stječemo (dinamičku i statičku) percepciju prostora koji nas okružuje.

U nastojanjima unaprjeđivanja fleksibilnih montažnih sustava čovjek se logično nameće kao model. Da bi se roboti mogli koristiti za zadaće u uvjetima nesređene okoline svakako moraju imati barem nekakvu percepciju prostora koji ih okružuje. U ovom radu predstavit će se tehnologija koja čini vizualne informacije dostupne strojevima. Tehnička implementacija vizualne percepcije naziva se strojni vid.

Ljudi pri montaži uzimaju/dobivaju mnogo više informacija vidom od postojećih sustava koji imaju implementiranu tehnologiju strojnog vida. Na temelju vizualnih informacija rješavaju više različitih složenih zadaća, od lociranja ugradbenih elemenata, preko njihovog prenošenja do sklopa te na kraju i samog određivanja putanje umetanja u sklop. Čovjek je sposoban služeći se samo vizualnim informacija kreirati plan montaže i izvršiti montažu.

Potrebno je stoga istražiti na koji način iskoristiti vizualne informacije kako bi se povećala radna inteligencija montažnih sustava. Time bi se postigle veće autonomije automatskih montažnih sustava i veće fleksibilnosti njihovih upravljačkih programa.

3 STROJNI VID

Strojni vid je tehnologija koja koristi optičke uređaje za automatsko dobivanje i interpretiranje slike prizora[4] kojom se dobivaju informacije i/ili upravljaju strojevi ili procesi.[10].

Tehnologija strojnog vida je usko povezana s sljedećim znanstvenim disciplinama:

- obrada slike,
- računalna (kompjutorska) grafika,
- umjetna inteligencija,
- prepoznavanje uzoraka (pattern recognition),
- psihofizika.

Obrada slike je znanstvena disciplina koja se bavi problemima transformiranja slike s ciljem povećanja njihove kvalitete po zadanom kriteriju. Dakle, rezultati algoritama za obradu slike su također slike. Interpretacija slika, odnosno ekstrakcija relevantnih informacija je područje algoritama strojnog vida.

Računalna grafika je znanstvena disciplina koja se bavi tehnologijom kreiranja digitalnih slika. Pri tome se pod digitalnim slikama podrazumijeva svaka slikovna interpretacija podataka. Na računalnom monitoru se svi podaci prikazuju tehnologijom računalne grafike bez obzira radilo se o prozorima operacijskog sustava, matematičkim fraktalima ili virtualnim CAD[5]

[4] Prizor – Objekt i pozadina u najjednostavnijem smislu. Dio prostora obuhvaćen osjetilom vizijskog sustava zbog mjerenja ili analize [14].
[5] CAD – eng. Computer Aided Design – Računalom podržano oblikovanje.

svjetovima. Rad na području računalne grafike, točnije na razvoju CAD sustava rezultirao je matematičkim aparatom te algoritmima za opis ploha i tijela u prostoru. Prema gornjoj definiciji, problem računalne grafika je vizualizacija tih virtualnih ploha i tijela. Strojni vid se bavi inverznim problemom. Na temelju slike nastoje se rekonstruirati geometrijska tijela te njihove pozicije i orijentacije u prostoru. Pri tome se koriste tehnike reprezentacije krivulja i ploha razvijene u području računalne grafike.

Ne postoji jedinstvena definicija umjetne inteligencije. Ipak, najšire je prihvaćena definicija koju je postavio Marvin L. Minsky, pionir područja umjetne inteligencije. Umjetna inteligencija je znanost kojom se postiže da strojevi čine stvari za koje bi bila potrebna inteligencija ako bi ih činili ljudi [11]. Zadatak umjetne inteligencije u sustavima strojnog vida je analiziranje vizualnih informacija. Pod analiziranjem vizualnih informacija se podrazumijeva izračunavanje simboličkih reprezentacija slika, odnosno otkrivanje i određivanje svojstava objekata na slikama. Slike se u pravilu prije analiziranja obrađuju algoritmima za obradu slike.

Prepoznavanje uzoraka je područje umjetne inteligencije koje se bavi klasificiranjem numeričkih i simboličkih podataka. Razvijene su mnoge statističke i sintaktičke tehnike za klasifikaciju uzoraka koje se koriste pri identifikaciji vizualnih ili zvučnih uzoraka i obilježja pomoću računala. Osim u sustavima strojnog vida, tehnologija prepoznavanja uzoraka se koristi kod sustava za prepoznavanje glasa i rukopisa [12][13].

Tehnologija strojnog vida mnogo duguje istraživanjima s područja psihofizike i kognitivnih znanosti. Rezultati istraživanja mehanizama ljudskog vida inspirirala su razvoj umjetnih vizijskih sustava, odnosno kreirala model za oblikovanje sustava strojnog vida.

Strojni vid više nije isključivo područje rada znanstvenika, već se ta tehnologija uspješno primjenjuje u praksi, prvenstveno u industriji i sigurnosnim djelatnostima. U tehnološkom smislu, strojni vid je multidisciplinarna tehnologija. Pod tim se podrazumijeva da je za izvedbu sustava strojnog vida potrebno poznavati i koristiti čitav niz tehnologija. Najvažnije od njih su [10]:

- tehnologija rasvjete,
- optika,
- senzorika,
- rukovanje materijalom,
- elektronika (analogna, digitalna, video),
- arhitektura digitalnih sistema,
- softver,
- regulacija,
- ....

Razlikuju se dvije osnovne arhitekture sustava strojnog vida, arhitekture [15]:

- signali → simboli → signali (SSS),
- signali → monolitski (iz)račun → signali (SMS).

Obje arhitekture imaju isti tip ulaza i izlaza. Ulaz u sustav su signali iz optičkog osjetila, a izlaz su upravljački signali za proces kojim se upravlja.

Sustavi strojnog vida arhitekture SSS nastoje hijerarhijski opisati sadržaj prizora na temelju njegove slike. Pod prizorom se pri tome smatra objekt i pozadina u najjednostavnijem smislu, odnosno dio prostora obuhvaćen osjetilom vizijskog sustava zbog mjerenja ili analize [14].

Proces obrade je najčešće podijeljen na tri razine. Prva, najniža razina obrade bavi se samo sa slikom. Slika se unaprjeđuje te se pronalaze i analiziraju njezina obilježja. Rezultat su svojstva točaka i pojedinih obilježja. Središnji proces nastoji matematički opisati identificirana obilježja slike. Gornja razina se koristi konceptualnim znanjem o sadržaju prizora i cilj joj je dodijeliti eksplicitna pojmovna tumačenja obilježja, odnosno, povezati ih s objektima u prizoru. Te informacije se koriste za generiranje upravljačkih signala kojima se djeluje na objekte u promatranom prizoru. Opisana arhitektura je usko povezana s područjem umjetne inteligencije.

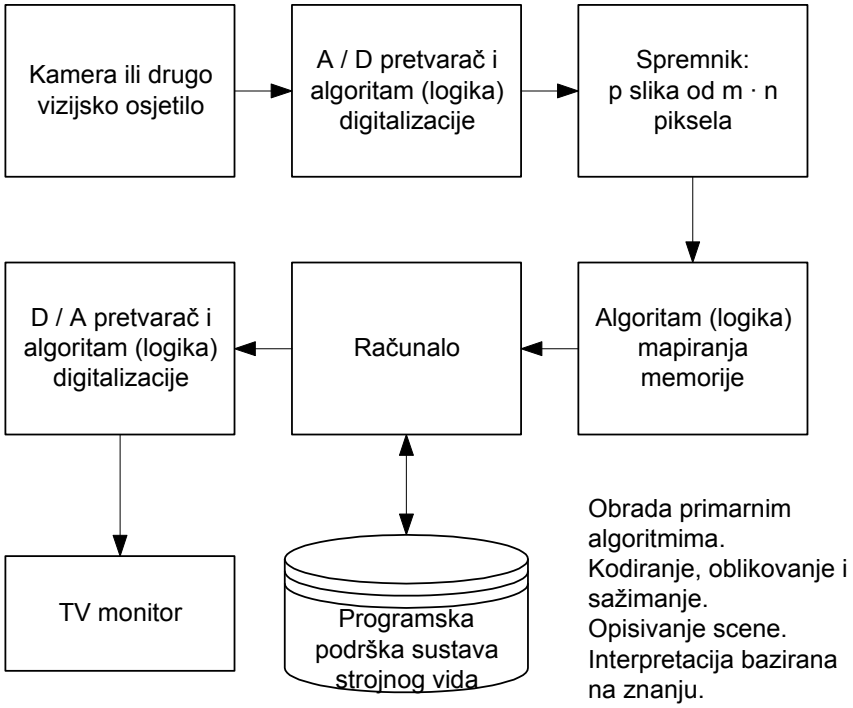
Osnovna ideja kod sustava arhitekture SMS oslanja se na globalnu mjeru, funkciju ili skup jednadžbi koji predstavljaju vezu između slike prizora i modela prizora. Ovakav pristup uključuje teoriju signala, statističku teoriju odluka i neuronske mreže. Upravljački signali se generiraju na temelju modela prizora.

Ovaj rad je ograničen na razmatranje sustava arhitekture SSS. U daljnjem tekstu se pod sustavima strojnog vida razumijevaju sustavi te arhitekture.

3.1. Struktura sustava strojnog vida

Sustav strojnog vida se sastoji od niza hardverskih i softverskih komponenata. U osnovi, sustav se sastoji od vizijskog osjetila, elektroničkog računala te programske podrške za obradu signala iz kamere odnosno programske podrške strojnog vida.

Elektroničko računalo je centralna komponenta sustava. Zaduženo je za prihvaćanje signala s vizijskog osjetila te formatiranje u oblik, prikladan za programsku podršku strojnog vida koja se također izvodi na računalu. Rezultate obrade i analize slike računalo prosljeđuje upravljačkim programima elemenata sustava kojima se upravlja strojnim vidom. Komponente sustava strojnog vida prikazane su slikom Sl.-2.

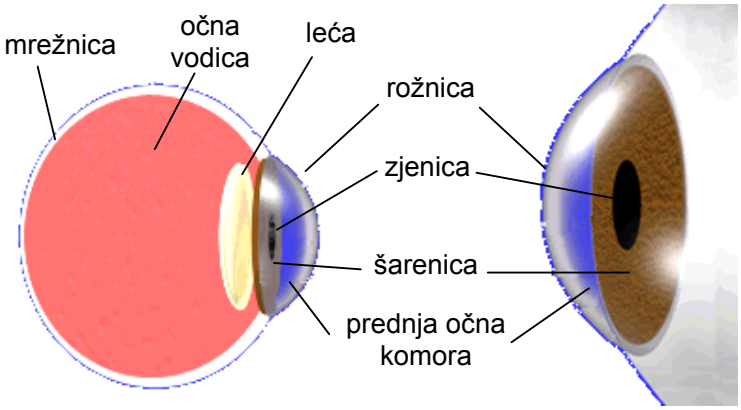


Sl.-2. Komponente sustava strojnog vida.

4 KAMERA

U sustavu strojnog vida kamera ima funkciju proizvesti sliku promatranog prizora. "Sirova slika" se mora obraditi i analizirati u sustavu za prepoznavanje kako bi se došlo do informacija u pogodnom obliku.

Zanimljiva je analogija u stupnju složenosti biološkog (ljudskog) vida i strojnog vida. Ljudsko oko je "razmjerno jednostavan" organ čija je anatomija (Sl.-3) i funkcija razjašnjena još u 16. stoljeću uvelike zahvaljujući Andreasu Vesaliusu (1514-1564) [16]. Oko (retina) je preko optičkog živca direktno povezano s mozgom, odnosno njegov je dio. Čak je 50 % ljudskog mozga povezano s vizualnom percepcijom [17].



Sl.-3. Anatomija ljudskog oka.

Poluvodička kamera, odnosno optičko osjetilo je, poput oka, relativno jednostavan uređaj u sklopu složenog sustava strojnog vida, čijoj složenosti najviše doprinose sustavi i algoritmi zaduženi za interpretaciju, odnosno percepciju slike.

Povijest digitalnih kamera započinje četrdesetih godina prošlog stoljeća s televizijom. Do 60-tih godina tehnologija je bila isključivo analogna. Jedan od usputnih proizvoda svemirskog programa Apollo je i digitalizacija signala kamera. Signali su se morali digitalizirati kako bi se neutralizirao utjecaj prirodne radijacije u svemiru.

Prvi sustav u kojem je bila umjetna vizualna percepcija je onaj iz 1930-tih kad je kompanija iz New Jeresy-a, Electronic Sorting Company, ponudila uređaje za sortiranje hrane koji su koristili specifične filtre i fotomultiplikatore kao detektore. Taj uređaj svakako nije model ljudskog vida, ali je ipak prvi uređaj koji je u upravljanju koristio optičko osjetilo [18]. Razvoj kognitivnih znanosti sredinom dvadesetog stoljeća dovodi do razvoja pravog sustava strojnog vida. Prvi pravi sustav strojnog vida razvio je 1965. L.G. Roberts i bio je sposoban u nekoj mjeri prepoznati 3D primitive u promatranom prizoru [19].

4.1. Poluvodička kamera

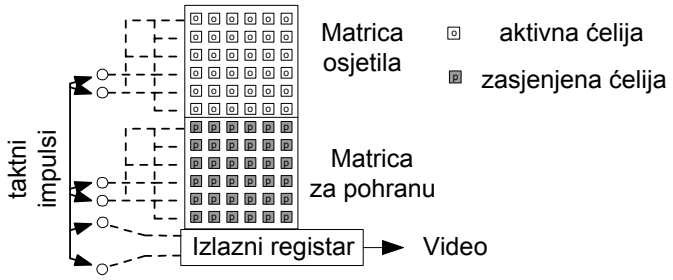
Poluvodička kamera je uređaj koja fokusira svjetlo na fotoosjetljivo poluvodičko osjetilo. Poluvodičko osjetilo je najčešće sastavljeno od pravokutne matrice ili samo jedne linije ekvidistantnih, diskretnih, fotoosjetljivih dioda – fotoćelija. Svaka fotoćelija djeluje kao opto-električni pretvarač, koji se nabija električnim nabojem proporcionalno intenzitetu svjetla kojim je bila osvjetljena tijekom vremena. Jedan ciklus nabijanja foto dioda naziva se integracijski ciklus. Danas postoje dva tipa poluvodičkih kamera, poluvodičke kamere sa CCD i CMOS poluvodičkim osjetilima.

Kod CCD (eng. Charge-Coupled Device) poluvodičkih osjetila naboji koji se stvore na pojedinim foto-diodama se međusobno povezuju tako da se mogu transportirati iz matrice direktno ili pretvoriti u analogni video signal.

Prema načinu na koji se akumulirani naboji odvode iz matrice fotoćelija razlikujemo dva tipa CCD čipova, s matričnim i s linijskim prijenosom naboja[6].

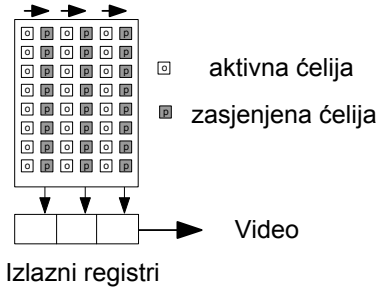
Najjednostavniji CCD s matričnim prijenosom se sastoji od izlaznog registra i CCD čipa kojemu je jedna polovica površine izložena svjetlu. Fotoćelije na izloženoj polovici nazivaju se aktivnim fotoćelijama. Druga polovica površine je zasjenjena neprozirnim materijalom. Naboji stvoreni u aktivnim ćelijama, tijekom integracijskog ciklusa, prebacuju se u zasjenjene ćelije. Nakon toga se u aktivnim ćelijama akumuliraju novi naboji, a oni u zasjenjenoj matrici se redak po redak prebacuju u izlazni registar iz kojeg se generira video signal. CCD s matričnim prijenosom naboja je shematski prikazan slikom Sl.-4 [20].

[6] Eng. frame-transfer i interline-transfer CCD.



Sl.-4. CCD poluvodičko osjetilo s matričnim prijenosom naboja.

CCD s linijskim prijenosom naboja (Sl.-5) se sastoji od dvodimenzionalne matrice fotoćelija organizirane u nizove parova aktivnih i zasjenjenih ćelija. Svi nizovi su paralelno spojeni s registrom, a svaka aktivna ćelija je povezana s odgovarajućom susjednom ćelijom za pohranu naboja. Iz zasjenjenih ćelija za pohranu, naboji se prebacuju u izlazni registar kao i kod CCD-a s matričnim prijenosom [21]. Budući da se aktivna ćelija nalazi odmah uz ćeliju za pohranu moguće je postići mnogo veće brzine prijenosa podataka. Prednost CCD s matričnim prijenosom je veća razlučivost slike po jedinici osvjetljene površine čipa, jer se kod CCD-a s linijskim prijenosom zasjenjene ćelije za pohranu nalaze u osvijetljenom području.



Sl.-5. CCD poluvodičko osjetilo s linijskim prijenosom naboja.

CMOS (eng. Complementary Metal Oxide Semiconductor) poluvodička osjetila očitavaju i pojačavaju naboj iz fotoćelije pomoću poluvodičkih elemenata smještenih na čipu odmah uz svaku fotoćeliju.

CCD poluvodička osjetila daju kvalitetniju "sliku" zbog većeg broja foto-dioda po jedinici površine čipa. CMOS poluvodička osjetila su jeftinija jer se mogu proizvesti standardnom tehnologijom za izradu čipova.

Poluvodička osjetila sastavljena od samo jedne linije fotoćelija nazivaju se linijska osjetila. Koriste se za snimanje rotacijskih objekata[7] ili objekata na konvejeru.

[7] Rotacijski predmet se zarotira za 360° ispred senzora i rezultat je slika njegovog razmotanog plašta.

Poluvodička osjetila su razvijana prvenstveno za potrebe zabavne industrije (televizije i filma). Stoga su poluvodičke kamere oblikovane na način da se mogu direktno spojiti na televizijski monitor. Za stvaranje iluzije pokreta na monitoru, zbog tromosti ljudskog oka, frekvencija izmjene statičnih slika mora biti veća od 30 Hz. Kako bi se što više smanjio efekt titranja slike, razvijena je tehnologija sparivanja[8] kojom se umjetno povećava frekvencija signala s poluvodičke kamere.

Set podataka s osjetila dobiven u jednom integracijskom ciklusu naziva se okvir[9]. Okvir se sastoji od dva polja[10]. Polje čini skup podataka sa svih parnih ili svih neparnih linija osjetila. Kod tehnologije sparivanja polja parnih i neparnih linije s poluvodičkog osjetila se sukcesivno prikazuju na televizijskom monitoru[11].

Kod sustava strojnog vida takav format podataka nije povoljan jer je za daljnju analizu potreban čitav okvir. Do okvira se dolazi postupkom integracije.

2:1 integracija sparivanjem okvira jednostavno zbraja dva uzastopna polja. Takva integracija je nepovoljna kod snimanja pokretnih objekata jer su dva polja nastala u različitim trenutcima [21].

Integracija se može vršiti i bez sparivanja uzastopnih okvira. Postoje dvije inačice postupka integracije bez sparivanja, integracija okvirom i integracija poljem.

Kod integracije okvirom s osjetila se očitavaju ili samo parne ili samo neparne linije. Rezultat je okvir kojeg čini samo jedno polje. Polovica informacija s osjetila se zanemaruje. Kod integracije poljem parovi uzastopnih linija na osjetilu se očitavaju istovremeno koristeći na taj način sve linije osjetila. Na primjer, linije 1 i 2 se očitavaju kao linija 1, linije 2 i 3 kao linija 2 i tako dalje dok se ne konstruira čitavi okvir. Postupak integracije iziskuje određeno vrijeme, a ovisno o primijenjenoj metodi odričemo se informacija ili je rezultat zamućena slika.

Kamere s progresivnim skeniranjem djeluju tako da odmah prenose sve linije sa senzora bez da se vrši sparivanje. Prednost takvog pristupa je oštra slika kod snimanja pomičnih prizora. Jedini nedostatak je inkompatibilnost sa standardnim televizijskim monitorima. To su najpogodnije kamere za upotrebu u sustavima strojnog vida jer omogućuju direktan izlaz digitalne slike te eliminiraju nepotrebne i dugotrajne obrade signala povezane s televizijskim standardom [20].

[8] Eng. interlacing.
[9] Eng. frame.
[10] Eng field.
[11] Tehnologija sparivanja je bila nužna zbog problema s fosforom na prvim televizijskim ekranima. Ionizirani fosforni sloj nije blijedio dovoljno brzo da bi se mogao prikazati čitav okvir svakih 1/60 sekundi.

4.2. Reprezentacija digitalne slike

4.2.1. Monokromatska digitalna slika

Kamera, odnosno vizijsko osjetilo pohranjuje prizor na dvodimenzionalnu digitalnu sliku [26].

Monokromatska digitalna slika je opisana matricom (poljem) F prema izrazu (1), pri čemu su vrijednosti funkcije $f(i,j)$ prirodni brojevi. Izrazima (2) i (3) definirana je kodmena funkcije $f(i,j)$.

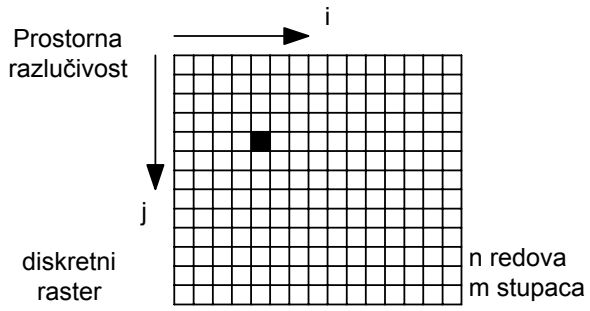
$$F = \begin{vmatrix} f(1,1) & f(1,2) & \dots & f(1,m) \\ f(2,1) & f(2,2) & \dots & f(2,m) \\ \dots & \dots & \dots & \dots \\ f(n,1) & f(n,2) & \dots & f(n,m) \end{vmatrix} \quad (1)$$

$$0 < f(i,j) < W \quad (2)$$

$$f(i,j) \in N_0 \quad (3)$$

Element matrice $f(i,j)$ naziva se piksel ili pel[12]. Intenzitet piksela se određuje na temelju signala s odgovarajuće fotoćelije poluvodičkog osjetila i ono je mjera osvjetljenosti piksela. Naravno, iznos akumuliranog naboja na samom čipu je analogna veličina koja se analogno/digitalnim pretvaračem digitalizira.

Intenzitet piksela $f(i,j)$, odnosno elementi matrice F označavaju intenzitete svjetlosti unutar malih pravokutnih regija stvarne (optičke) slike. Pri tome je $f(i,j) = 0$ potpuni nedostatak svjetlosti, odnosno crna boja, dok $f(i,j)=W$ označava bijelu boju. Budući da je $f(i,j)$ element skupa N_0 veličina W definira razlučivost signala. Notacija digitalne slike je ilustrirana slikom Sl.-6.



Sl.-6. Notacija digitalnih slika.

[12] Eng. picture element

f(i,j) je vrijednost intenziteta svjetlosti u točki. No, ako je odgovarajuća pravokutna regija dovoljno mala, aproksimacija je dovoljno dobra za većinu aplikacija. Matrica *F* sadrži *m·n* elemenata, a ta se veličina naziva *prostorna razlučivost.*

Svaki piksel zahtjeva $log_2(1+W)$ bita za pohranu u memoriji računala. Uobičajeno je da se za pohranu monokromatskih slika koristi rezolucija signala od jednog bajta, odnosno:

$$W = 2^8 - 1 = 255 \qquad (4)$$

Slika Sl.-7 prikazuje monokromatsku digitalnu sliku prostorne razlučivosti 64·64 piksela te razlučivosti signala *W = 255*. Za pohranu te slike potrebno je *64·64·8= 32768* bita, odnosno *4096* bajta.

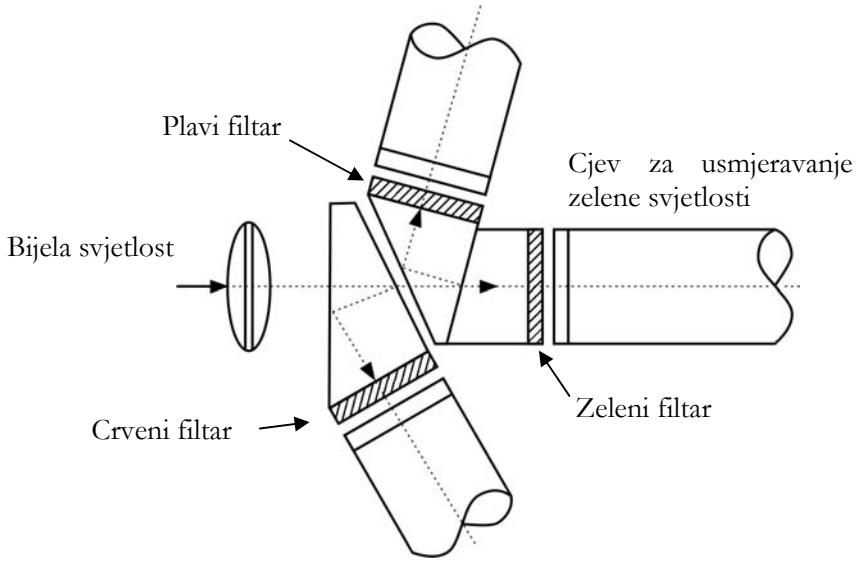
Sl.-7. Monokromatska digitalna slika prostorne razlučivosti 64·64 piksela te razlučivosti signala *W* = 255.

Binarne slike su slike kod kojih su moguća samo dva intenziteta svjetlosti *f(i,j)* u matrici *F*, intenziteti 0 i 1. Razlučivost signala binarnih slika *W* je 1. Stoga je potrebno osam puta manje memorije za njihovu pohranu u memoriji (1 bit po pikselu) (Sl.-8).

Sl.-8. Binarna slika.

4.2.2. Slika u boji

Osjećaj boje se može ostvariti u oku superponiranjem tri ili četiri zasebne slike. Poluvodičke kamere u boji djeluju na sličan način. U tipičnoj konstrukciji kamere u boji, shematski prikazanoj na slici Sl.-9, zraka svjetlosti se najprije sustavom filtra i leća dijeli na tri kompone, crvenu, zelenu i plavu svjetlost. Svaka se zraka fokusira na zasebno poluvodičko osjetilo na kojem se formira digitalna slika na način kako je to opisano u prethodnom poglavlju.



Sl.-9. Razdioba svjetlosti na crvenu, zelenu i plavu komponentu.

Signal iz poluvodičke kamere u boji, odnosno slika u boji se opisuje trima matricama prema izrazima (5), (6) i (7) pri čemu svaka matrica opisuje sliku sa jednog poluvodičkog osjetila.

$$R = \{r(i,j)\} \quad (5)$$

$$G = \{g(i,j)\} \quad (6)$$

$$B = \{b(i,j)\} \quad (7)$$

Matrica R označava intenzitete svjetlosti nakon što je svjetlost propuštena kroz crveni filtar, matrica G kroz zeleni a matrica B kroz plavi. Vektor $\{r(i,j), g(i,j), b(i,j)\}$ opisuje intenzitet i boju u točki (i,j) multispektralne slike. Dakle, multispektralne slike se opisuju pomoću tri monokromatske slike. Ukupna količina potrebne memorije M je dana izrazom (8) pri čemu je r broj monokromatskih slika.

$$M = m \cdot n \cdot r \cdot \log2(1+W) \text{ [bit]} \quad (8)$$

Opisani su samo osnovni oblici reprezentacija slika, nužni za razumijevanje jednostavnih funkcija odnosno algoritama za obradu slike. Postoje mnogi alternativni te napredniji oblici reprezentiranja slike, no oni neće biti opisivani u ovome radu [27].

5 ALGORITMI VIZIJSKIH SUSTAVA

Sustav strojnog vida je sačinjen od niza algoritama kojima se na temelju digitalne slike prizora dobivaju informacije o stanju u prizoru. Algoritmi od kojih je sačinjena programska podrška nazivaju se vizijski algoritmi, a programska podrška vizijski sustav.

Pojam *vizija, vizijski* dolazi od latinske riječi *videre* što znači vidjeti. Njome se opisuje spoznaja i to u smislu slike koju stvara mašta, s ili bez uporišta u stvarnom svijetu, ali i čin i sposobnost gledanja. Vizijski sustav sačinjen od vizijskih algoritama omogućuje gledanje i spoznaju.

U tehničkim sustavima spoznaja se ostvaruje strukturiranjem informacija. Vizijski sustav proizvodi informacije koje se koriste za upravljanje procesa. Kako se strojni vid najčešće može pronaći u sustavima s automatskom kontrolom, navigacijom i/ili sklapanjem, vizijski sustav proizvodi informacije kojima se:

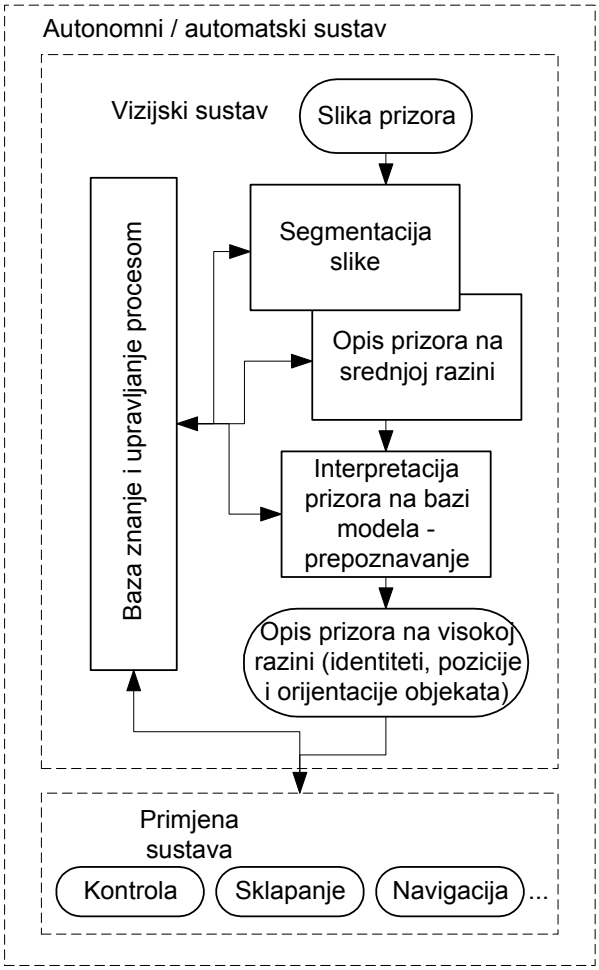
- identificiraju objekti,
- određuju dimenzije objekata,
- određuju pozicije i orijentacije objekata.

Struktura vizijskog sustava je prikazana slikom Sl.-10. Vizijski sustav je prikazan u kontekstu njegove primjene, budući da je vizijski sustav obično tek jedan od sustava koji sudjeluju u upravljanju. Ulaz u vizijski sustav je slika prizora, a izlaz su pozicije i orijentacije identificiranih objekata ili opis prizora.

Proces obrade i analize sastoji se od dvije osnovne faze. U prvoj fazi se slika segmentira na dijelove za koje se očekuje da odgovaraju objektima. Rezultati segmentacije se opisuju matematički, simbolički ili analitički.

Druga faza je interpretacija prizora na bazi modela. U ovoj fazi se nužno uključuje kako znanja o sadržaju prizora i geometrijskim značajkama objekata na prizoru te njihovim međusobnim odnosima, tako i sva ostala znanja o kontekstu sustava koja mogu pridonijeti procesu prepoznavanja.

Operacije i algoritmi vizijskih sustava uobičajeno se dijele na tri razine, na primarne vizijske algoritme, algoritme srednje razine i algoritme visoke razine. Ova podjela je utemeljena na količini znanja/informacija o objektima na prizoru koju zahtijevaju pojedini algoritmi.



Sl.-10. Struktura sustava za obradu i analizu slike – Vizijskog sustava

Primarni algoritmi vizije zaduženi su za pronalaženje deskriptora na slikama koji se također uglavnom reprezentiraju kao slike. Pod deskriptorima se razumijevaju obilježja poput naglih promjena intenziteta susjednih piksela što bi moglo odgovarati signifikantnim granicama objekata u prizoru. Proces najčešće ne posjeduje nikakva konceptualna znanja o sceni niti o

prostornom odnosu kamere i prizora. Rezultat primarnih algoritama vizije je skup neovisnih deskriptora kao što su bridovi, linije, točke, regije i slično.

U 6. poglavlju opisati će se najvažniji primarni algoritmi vizije u kontekstu problema segmentacije slike budući da je segmentacija problem pri čijem rješavanju se upošljavaju algoritmi ove razine.

Zadaća vizijskih algoritama srednje razine je matematičko/analitičko opisivanje slikovnih rezultata algoritama primarne razine. Na taj se način opisuju oblici i položaji pojedinih područja na slici. Rezultati vizijskih algoritma srednje razine su značajke slika.

Algoritmi ove razine ne koriste znanja o objektima na prizoru, ali mogu koristiti znanja o procesu kreiranja slike, odnosno o načinu na koji se oblik manifestira na slici [30].

Kao i primarni vizijski algoritmi, algoritmi srednje razine se uglavnom povezuju s problemom segmentacije, stoga će se neki od njih opisati u 6. poglavlju.

Vizijskim algoritmima visoke razine se interpretira prizor. Interpretacija se vrši na temelju znanja o objektima na prizoru i njihovim međusobnim odnosima. Znanja o objektima su strukturirana simbolički i analitički kako bi se mogla koristiti u kombinaciji s rezultatima algoritama srednje razine. Tipični rezultati ovih algoritama su lokacije i imenovanja objekata prisutnih na prizoru, identifikacija objekata koji zadovoljavaju određenu funkciju, opis gibanja ili čak sažetih opisa prizora.

Vizijski algoritmi visoke razine se uglavnom povezuju s problemom "prepoznavanja".

6 SEGMENTACIJA

Jedan od najvažnijih zadataka vizijskih sustava jest identificiranje regija u slici koje predstavljaju objekte. Postupak particioniranja slike u regije se naziva segmentacija. U idealnom slučaju, regija na slici odgovara objektu ili dijelu objekta. Proces segmentacije se vrši primarnim vizijskim algoritmima pa se regije isprva tretiraju kao kandidati za objekte odnosno njihove dijelove. No, što to čini regiju na slici ako je promatramo na razini piksela? Osnovna karakteristika regije je da pikseli koji čine regiju zadovoljavaju nekakav uvjet ili funkciju, za razliku od svih ostalih piksela na slici. Funkcija po kojoj se vrši postupak segmentacije određuje kvalitetu rješenja.

Regiju čini skupina povezanih piksela sa sličnim svojstvima [12]. U prizoru se može nalaziti više objekata od kojih je svaki prikazan zasebnom regijom. Osim toga, ovisno o algoritmu kojim se vrši segmentacija te stupnju složenosti geometrije objekata u prizoru, jedan objekt može biti prikazan s više regija.

Zbog prirode algoritama za segmentaciju potrebno je znanje o sadržaju prizora da bi se mogla postaviti ispravna korelacija između pronađenih regija te pozicija i orijentacija objekata u prizoru.

Algoritmi za segmentaciju mogu se podijeliti na dvije skupine.

- Algoritmi za segmentaciju bazirani na regijama.
- Algoritmi za segmentaciju bazirani na bridovima.

Kod segmentacije bazirane na regijama, označava se skupina piksela koja odgovara objektu. Pikseli se dodjeljuju u skupinu na temelju kriterija koji ih razlikuje od ostalih piksela na slici. Kriterij opisuje ideju da će se objekt projicirati na piksele koji su blizu jedan drugome i to sa

sličnim intenzitetima. Ta pretpostavka često nije zadovoljena te se na temelju znanja o sadržaju prizora regije moraju grupirati ili dijeliti da bi se objekti ispravno opisali.

Alternativni pristup particioniranju slike jest particioniranje konturama. Konture reprezentiraju granice regija na slici. Identificiraju se na temelju "velike" razlike intenziteta osvjetljenosti susjednih piksela.

Zadaća i jednih i drugih algoritama za segmentaciju je ista te se očekuju istovjetni rezultati. Na temelju definicije bridova jednostavnim se algoritmima za popunjavanje kontura mogu dobiti regije koje bi trebale odgovarati objektu. Jednako tako, algoritmi za praćenje ruba produciraju prikaz regija konturama. Iako su rješenja kompatibilna rijetko će stvarni rezultati biti jednaki. Uzrok tome su naravno nesavršeni algoritmi koji sami po sebi nisu sposobni anulirati sve smetnje.

6.1. Segmentacija bazirana na regijama

Dana je sljedeća definicija problema segmentacije.

Ako je:

- I skup piksela koji predstavljaju objekte
- $P(\cdot)$ uvjet homogenosti
- R_i identificirana regija

Za pronađenih n regija moraju biti zadovoljeni izrazi (9), (10) i (11).

$$\bigcup_{i=1}^{n} R_i = I \qquad (9)$$

$$P(R_i) = Istina \qquad (10)$$

$$P(R_i \cup R_j) = Neistina \qquad (11)$$

Uvjet homogenosti $P(\cdot)$ definira usklađenost piksela koji pripadaju regiji. Funkcija uvjeta homogenosti određuje uspješnost postupka segmentacije. Oblik te funkcije je uvjetovan specifičnim prizorom odnosno slikom, a kod naprednih algoritama se određuje automatski.

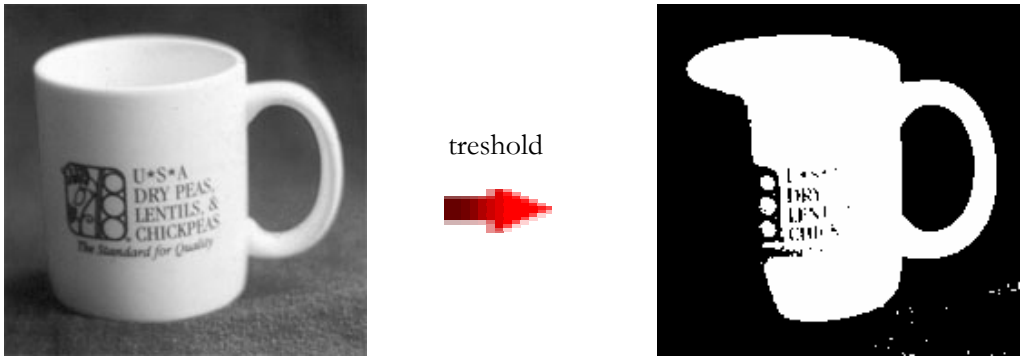
6.1.1. Treshold algoritmi

Najjednostavniji oblik segmentacije je segmentacija upotrebom treshold[13] algoritma. Treshold algoritmi pretvaraju monokromatske ili multispektralne slike u binarne i na taj način odjeljuju objekte od pozadine. Kako bi treshold algoritmi bili učinkoviti kontrast između objekata u sceni i pozadine mora biti dovoljno velik. Osim toga, moraju biti poznati intenziteti[14] pozadine ili objekata. Treshold algoritam se može opisati izrazom (12) pri čemu je F_T slika dobivena treshold algoritmom. Skup Z opisuje intenzitete piksela koji odgovaraju intenzitetima piksela objekata, a $f(i,j)$ je intenzitet piksela (i,j) slike koja se analizira.

$$F_T(i,j) = \begin{cases} 1 \;\; ako \; f(i,j) \in Z \\ 0 \end{cases} \qquad (12)$$

Rezultat treshold algoritma je binarna slika $B(i,j)$, pri čemu je:

$$B(i,j) = F_T(i,j) \qquad (13)$$



Sl.-11. Segmentacija slike jednostavnim treshold algoritmom.

Rezultat segmentacije jednostavnim treshold algoritmom je prikazan slikom Sl.-11, pri čemu je uzeto da je $Z = [128,255]$. Razlučivost signala analizirane slike W iznosi 255. Dobiveno rješenje oslikava niz problema koji se javljaju prilikom particioniranja. Iako se radi o svijetlom objektu na relativno tamnoj podlozi, rezultanta regija ne obuhvaća čitav objekt. Uzrok tome je sjena i/ili nepravilno definiran skup Z.

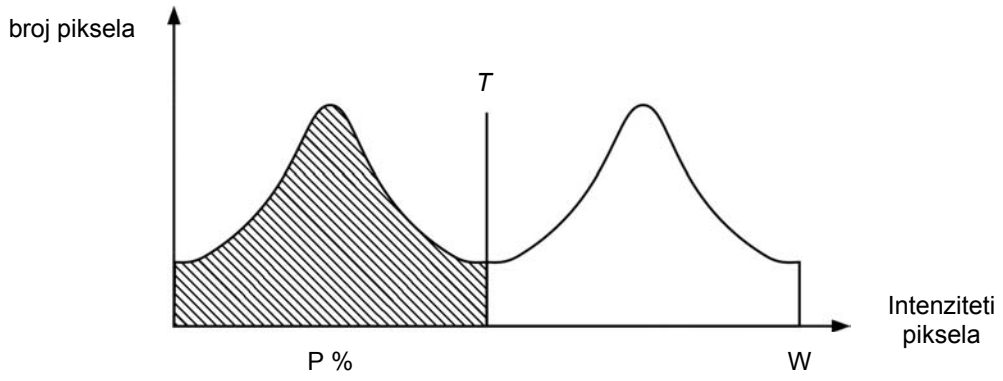
Kako bi postupak segmentacije bio što robusniji, uvjet homogenosti mora biti određen automatski. Za automatsko određivanje uvjeta homogenosti potrebna su znanja o objektima u prizoru, funkciji sustava, namjeni sustava strojnog vida te ostalim relevantnim uvjetima u prizoru. To su znanja poput:

[13] Treshold eng.- Prag.
[14] Pod intenzitetima se razumijevaju intenziteti piksela slike koja se analizira.

- karakteristike intenziteta osvjetljenosti objekata,
- veličine objekata,
- očekivani udio površine objekata u površini slike,
- broj različitih objekata.

Navedena znanja se opisuju na način općenitiji od fiksnog iznosa praga treshold algoritma. Automatski treshold algoritmi analiziraju distribuciju intenziteta svjetlosti na slici te na temelju znanja određuju prag, odnosno pragove.

P – tile algoritam je jednostavan automatski treshold algoritam. Pretpostavlja se da je intenzitet osvijetljenosti objekta "dovoljno" različit od intenziteta pozadine te da je poznat udio površine slike ***P*** koji pripada objektima. Ukoliko su objekti tamniji od pozadine, uzima se da je prag ***T*** jednak intenzitetu najsvjetlijeg od ***P%*** najtamnijih piksela. Taj postupak je opisan slikom Sl.-12. Osjenčana površina je ***P%*** svih piksela.



Sl.-12. Određivanje praga treshold algoritma P-tile metodom na temelju histograma slike. Na apscisi se nalaze diskretne vrijednosti intenziteta piksela od *0* do *W*, a na ordinati je broj piksela.

Ovaj jednostavni algoritam ima samo ograničenu upotrebu. Rijetke su aplikacije gdje se može definirati ***P*** za općeniti slučaj.

"*Mode method*" algoritam, kao i *P-tile* algoritam određuje vrijednost praga ***P*** na temelju histograma. Uz pretpostavku da su objekti i pozadina na slici prikazani različitim intenzitetima te da je slika degradirana šumom, histogram slike je zbroj dvije normalne distribucije. Histogram sadrži dva peak-a (vrha) i jednu udubinu između njih. Minimum udubine, odnosno, intenzitet s najmanjom frekvencijom je vrijednost praga ***T***. Ova metoda se može generalizirati za slučaj više različitih objekata s različitim frekvencijama. Lokalni minimum svake identificirane doline histograma predstavlja jedan prag treshold algoritma.

Opisani algoritmi se mogu koristiti samo u slučaju jednolikog osvjetljenja prizora. Na nepravilno/nejednoliko osvjetljenom prizoru pojavljuju se sjene koje postaju dominantna obilježja slike. Kako bi se poništio njihov utjecaj razvijeni su adaptivni i varijabilni treshold algoritmi.

"Adaptivni treshold" algoritmi rješavaju taj problem tako da se slika najprije podjeli na nekoliko manjih segmenata. Time se smanjuje utjecaj sjene na pojedinom segmentu i moguće je

upotrijebiti jedan od prije opisanih treshold algoritama. Konačno rješenje adaptivnog algoritma je zbroj identificiranih regija na svim segmentima slike.

"Varijabilni treshold" algoritmi nastoje nejednoliko osvjetljenje prizora opisati funkcijom, najčešće linearnom ili bikvadratnom funkcijom. Funkcija se dobiva na temelju distribucije intenziteta osvjetljenja pozadine. Kada je nepravilnost osvjetljenosti opisana funkcijom, njezin utjecaj na sliku se može kompenzirati. Ta se tehnika naziva i "normalizacija pozadine".

"Dvostruki treshold" algoritam se primjenjuje pri segmentaciji slike gdje se dijelovi objekata koje nastojimo identificirati dovoljno razlikuju od pozadine, no neki njihovi dijelovi su pak vrlo slične nijanse kao i pozadina. U takvom slučaju u prvom se koraku identificiraju regije koje sigurno pripadaju objektima. U daljnjim se koracima te regije proširuju s pikselima kandidatima na temelju dodatnih kriterija. Dodatni kriteriji uključuju promjenu veličine praga, udaljenost prethodno pronađenih regija i slično. Izbor dodatnih kriterija ovisi o aplikaciji.

Treshold algoritmi za segmentaciju slike su brzi i robusni algoritmi koji često ispunjavaju zahtjeve aplikacija sustava strojnog vida u sređenoj okolini. Da bi treshold algoritam ispravno identificirao regije mora biti zadovoljen sljedeći uvjet: cjelokupna površina objekta mora biti "dovoljno" različita od čitave površine pozadine. Ova ograničenje proizlaze iz glavnog nedostatka algoritama za segmentaciju baziranih na histogramu slike. Histogram ne sadrži nikakve geometrijske informacije. Posljedica toga je da se jedno od najvažnijih obilježje regija, blizina piksela, ne koristi prilikom segmentacije.

6.1.2. Algoritmi za dijeljenje i spajanje regija

Pod algoritmima za dijeljenje i spajanje regija razumijevaju se algoritmi koji nizovima operacija spajanja susjednih regija te podjelama postojećih regija doprinose točnosti segmentacije. Pri tome se najčešće unapređuje segmentirana slika dobivena treshold algoritmom. Algoritmi se koriste konceptualnim znanjem o prizoru i/ili znanjem o postupku kreiranja slike.

Operacije spajanja i dijeljenja regija eliminiraju pogrešne granice te spajaju susjedne regije koje pripadaju istom objektu i dodaju granice odnosno dijele regije koje sadrže dijelove različitih objekata. Algoritmi su temeljeni na sljedećim pristupima:

- spajanje susjednih regija sa sličnim karakteristikama,
- uklanjanje nepotrebnih granica,
- redefiniranje regija na temelju topoloških svojstava regija,
- redefiniranje regija na temelju informacija o obliku objekata na prizoru,
- redefiniranje regija na temelju semantičkih informacija o prizoru.

Prva tri pristupa omogućuju kreiranje fleksibilnih i potpuno automatskih algoritama jer ne koriste informacije specifične za određenu domenu. U nastavku će biti kratko opisani neki važniji algoritmi za segmentaciju bazirani na dijeljenju i spajanju regija.

6.1.2.1. Algoritmi za spajanje regija

Njihov je zadatak spajanje regija za koje se pretpostavlja da pripadaju istom objektu. Jasno je da je jedini zadatak ili problem koji se ovdje pojavljuje utvrđivanje sličnosti među regijama.

Općenito, o sličnostima regija se zaključuje na temelju:

- karakteristika intenziteta piksela unutar regija,
- svojstava bridova između promatranih regija,
- prostornog odnosa promatranih regija.

Prostorni odnos regija se koristi samo kao dodatni uvjet prilikom analize sličnosti regija na temelju prva dva uvjeta.

Algoritmi za spajanje bazirani na karakteristikama intenziteta piksela

Dva su pristupa za ocjenu sličnosti promatranih susjednih regija:

- ocjena sličnosti na temelju prosječnog intenziteta piksela,
- statistička ocjena sličnosti.

Kod prvog pristupa sličnost regija se utvrđuje na temelju razlike aritmetičkih sredina intenziteta piksela promatranih regija. Ako je razlika manja od neke predodređene vrijednosti, pretpostavlja se da regije pripadaju istom objektu te se spajaju.

Kod statističkog pristupa ocjeni sličnosti nastoji se utvrditi da li se promatrane regije, odnosno intenziteti njihovih piksela mogu opisati jednom statističkom distribucijom. Ukoliko se ne mogu, pretpostavlja se da regije pripadaju različitim objektima.

Algoritmi za spajanje regija bazirani na karakteristikama granica

Algoritmi za spajanje regija bazirani na karakteristikama granica nastoje identificirati slabe granice i spojiti regije koji oni odjeljuju. Pod slabim granicama se razumijevaju granice između susjednih regije za koje se pretpostavlja da odgovaraju istom objektu.

Granice se proglašavaju slabima na temelju tri kriterija:

- producira li uklanjanje granice regiju u kojoj se intenziteti piksela mijenjaju "dovoljno malo",
- razlika između intenziteta piksela s obje strane promatrane granice mora biti manja od nekog unaprijed definiranog iznosa,
- udio dužine granice čije uklanjanje se razmatra mora činiti značajni udio granice regija kandidata za spajanje.

6.1.2.2. Algoritmi za dijeljenje regija

Algoritmi za dijeljenje pronalaze nehomogene regije te ih dijele na nekoliko homogenih. Pretpostavka je da regije čija svojstva ne zadovoljavaju uvjet homogenosti sadrže dijelove više od jednog objekta.

Homogenosti regija može se utvrditi

- varijancom intenziteta piksela,

- funkcijom aproksimiranom intenzitetima piksela.

Varijanca intenziteta piksela je mjera njihove "konstantnosti". Ako pak se homogenost regije utvrđuje aproksimiranom funkcijom, kao mjera homogenosti se koristi iznos odstupanja intenziteta piksela od funkcije.

Problem određivanja načina na koji će se regija podijeliti je znatno složeniji od utvrđivanja njezine homogenosti.

Najjednostavnije metode za dijeljenje regija su one koje podijele regiju na fiksni broj regija jednake veličine. Takve metode se nazivaju regularne dekompozicijske metode.

Učinkovita regularna dekompozicijska metoda je ona koja sukcesivno dijeli regije na nekoliko manjih (obično četiri) sve dok se ne dobiju homogene regije. Da bi se završio proces segmentacije, nju mora slijediti postupak spajanja regija.

6.1.2.3. Algoritmi za segmentaciju bazirani na dijeljenju i spajanju regija

Vrlo su učinkoviti algoritmi za segmentaciju koji vrše i diobe i spajanja regija. Takvim algoritmima se može uspješno izvršiti segmentacija slike bez prethodne inicijalne segmentacije treshold algoritmom.

Slijedi jedan općeniti algoritam za segmentaciju baziran na spajanju i dijeljenju regija [22].

1. Izvršiti inicijalnu segmentaciju i odrediti uvjet homogenosti.
2. Ako regija *R* nije homogena, dijeli se na četiri regije jednake veličine. Dvije ili tri nove regije se međusobno spajaju ukoliko to dozvoljava uvjet homogenosti. Ova operacija se odvija sve dok postoje regije koje ne zadovoljavaju uvjet homogenosti.
3. Traže se i spajaju parovi susjednih regija prema uvjetu homogenosti. Ova operacija se nastavlja dok postoje parovi susjednih regija koji se mogu spojiti.
4. Ako je potrebno ukloniti "male" regije one se spajaju s najsličnijom susjednom regijom.

6.1.3. Algoritmi za segmentaciju bazirani na rastu regija

Algoritmi za segmentaciju bazirani na rastu regija inspirirani su idejom da se vrijednosti intenziteta piksela ispravno definirane regije mogu modelirati jednostavnom matematičkom funkcijom.

Intenziteti piksela se najčešće aproksimiraju jednostavnim linearnim i kvadratnim funkcijama, iako se kao uvjet homogenosti može koristiti bilo koje svojstvo regije (prosječni intenzitet piksela, varijanca, boja, tekstura, ...).

Algoritam započinje inicijalnom segmentacijom koja može biti neka od regularnih dekompozicijskih metoda ili treshold metoda. Ipak, najčešće se započinje s pravokutnim regijama dimenzija 5x5 ili 7x7 [12].

Za inicijalne regije se kreiraju matematički modeli i neke od njih se proglašavaju sjemenim regijama. Ovo je vrlo osjetljiv dio postupka s obzirom da nije moguće unaprijed znati kako definirati sjemene regije, jer to je u stvari sam problem segmentacije. Tu se uključuje konceptualno znanje o domeni i znanje o postupku kreiranja slike.

Regije rastu tako da se u njih uključuju susjedni pikseli koje zadovoljavaju model regije. Nakon toga se definira novi model proširene regije i ponovno se nastoji ekstrapolirati na susjedne piksele. Postupak se nastavlja sve dok svi pikseli nisu uključeni u neku od regija.

6.1.4. Opisivanje regija

Podaci i algoritam su dvije osnovne komponente svakog računalskog programa. Organizacija podataka može značajno utjecati na odabir i implementaciju algoritma za segmentaciju. Pojedini algoritmi za segmentaciju djeluju samo u sprezi s točno određenom strukturom podataka za opisivanje regija. Stoga će se ukratko opisati najvažnije među njima.

Kao i ostale elemente sustava strojnog vida, aplikacija uvjetuje koja struktura za opisivanje regija će se koristiti. Općenito, strukture se mogu podijeliti na:

- strukture za opisivanje matricom,
- strukture za simbolički opis,
- hijerarhijske strukture,
- topološke strukture.
- relacijske strukture

Metode koje opisuju regije matricom koriste matricu dimenzija jednakih matrici originalne slike. Svaki element matrice odgovara jednom pikselu i definira kojoj regiji taj piksel pripada. Valja spomenuti i metode opisivanja bit-maskama gdje je svaka regija opisana s jednom matricom. Preklapanjem bit-maski dobivaju se sve regije na prizoru. Opisivanje bit-maskama omogućuje da se pojedini pikseli dodjele u više od jedne regije.

Regija se može opisati i simbolički. Najčešći oblici simboličkog opisivanja regija su opisivanje:

- opisanim pravokutnikom,
- težištem,
- momentom,
- eulerovim brojem[15].

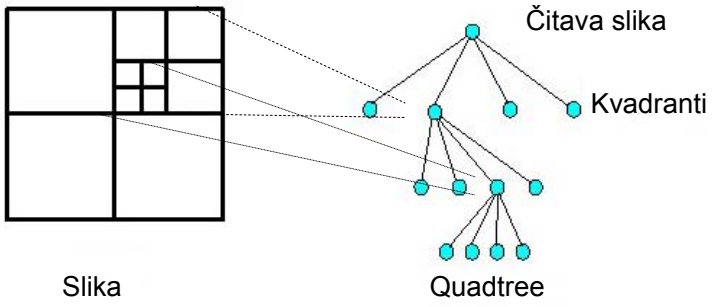
Osim geometrijskih svojstava regije često je zanimljiva i distribucija intenziteta piksela unutar regija. Aritmetička sredina i varijanca intenziteta su česte simboličke značajke regija.

[15] Eulerov broj je svojstvo regije. Definiran je kao broj odvojenih područja na slici koja čine jednu regiju umanjen za broj "rupa" unutar tih područja.

Hijerarhijske strukture podataka omogućavaju upotrebu algoritama koji određuju strategiju obrade na bazi relativno malo podataka. Maksimalna razlučivost podataka se koristi samo u dijelovima slike gdje je to zaista potrebno.

Najčešća hijerarhijska struktura podataka za opisivanje regija je "quadtree" struktura. Najveću primjenu ima kod regularnih dekompozicijskih metoda segmentacije. Quadtree strukture u prvom koraku pokriva planarnu interesnu zonu pravokutnikom. Zatim se pravokutnik rekurzivno dijeli sve dok se ne postigne stanje u kojem svaki pravokutnik sadrži prikladan uniformni skup podataka [24].

Binarne slike se dijele dok se ne postigne da su svi pikseli u pravokutnicima ili crni ili bijeli. Monokromatske slike se dijele sve dok se ne postigne određeni iznos varijance intenziteta, iako su moguće i druge mjere sličnosti intenziteta piksela. "Quadtree" struktura podataka je prikazana na slici Sl.-13.



Sl.-13. . Quadtree struktura podataka za opisivanje regije.

Topološke strukture podataka opisuju sliku kao skup elemenata i njihovih relacija. Relacije se često reprezentiraju grafovima.

Graf G je algebarska struktura koja se sastoji od skupa čvorova V i lukova E.

$$G = (V,E) \tag{14}$$

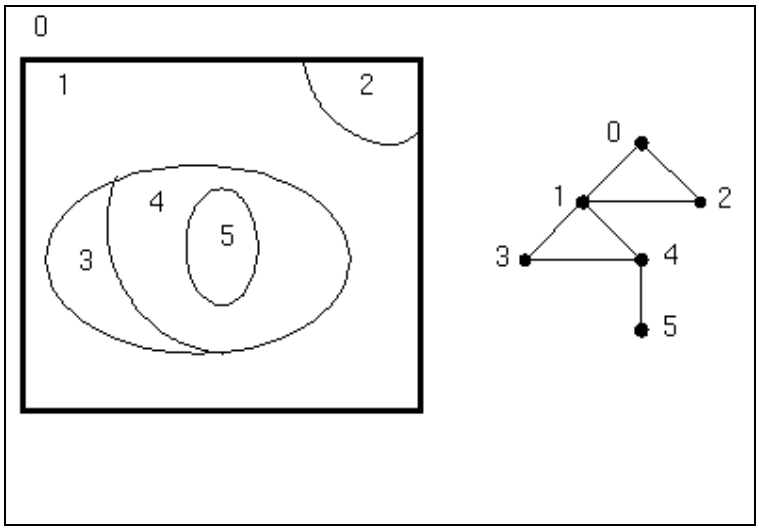
$$V = \{v_1, v_2, \ldots, v_n\} \tag{15}$$

$$E = \{e_1, e_2, \ldots, e_m\} \tag{16}$$

Svaki luk e_k povezuje par čvorova $\{v_i, v_j\}$. Stupanj čvora je jednaka broju lukova povezanih s njim. Lukovima i čvorovima se mogu dodijeliti vrijednosti koje opisuju neka svojstva objekta. Grafovi s dodijeljenim vrijednostima se nazivaju vrednovani grafovi.

Grafovi susjednih regija (eng. region adjacency graph**)** RAG su tipičan predstavnik ove klase struktura podataka. Čvorovi odgovaraju regijama, a susjedne regije[16] su povezane lukovima.

Segmentiranu slika na kojoj su regije označene brojevima te odgovarajući graf susjednih regija, prikazana je slikom Sl.-14. Nula predstavlja piksele izvan slike. Čvor nula se koristi kako bi se ukazalo na regije koje dodiruju granice slike [23].



Sl.-14. Segmentirana slika i odgovarajući graf susjednih regija.

RAG posjeduje nekoliko korisnih karakteristika:

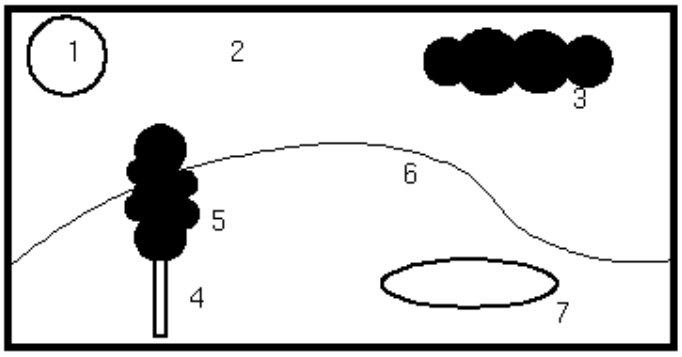
- Ako regija obuhvaća druge regije, dio grafa koji odgovara obuhvaćenim regijama se može lako izdvojiti.
- Čvorovi prvog stupnja odgovaraju jednostavnim "rupama".
- RAG se može koristiti za usporedbu s pohranjenom RAG strukturom za prepoznavanje objekata.

Za opisivanje regija se koriste i relacijske baze podataka. Sve informacije koncentriraju se u relacije između semantički važnih dijelova slike – objekata – koji su rezultat segmentacije.

Relacije se pohranjuju u tabličnom obliku. Slika Sl.-15 prikazuje identificirane i označene objekte na slici. Tablica 1 je relacijska tablica sa semantičkim i numeričkim karakteristikama objekata. Pojedini objekti su povezani sa svojim imenima, dakle prepoznati su. Njihova pozicija je određena pozicijom gornjeg lijevog piksela objekta[17].

[16] Regije su susjedne ako imaju zajedničku granicu.

[17] Ishodište slike je u gornjem lijevom uglu, pa se redci piksela broje od gore prema dolje, a stupci s lijeva na desno.



Sl.-15. Slika sa identificiranim objektima

Tablica 1. Tablica sa semantičkim i numeričkim karakteristikama objekata

Broj	Ime objekta	Boja	Prvi redak	Prvi stupac
1	Sunce	Bijelo	5	40
2	Nebo	Plavo	0	0
3	Oblak	Sivo	20	180
4	Drvo	Smeđe	95	75
5	Krošnja	Zeleno	53	63
6	Brijeg	Svijetlo zeleno	97	0
7	Jezero	Plavo	100	160

Opisivanje relacijskim strukturama podataka je primjereno višim razinama sustava strojnog vida koji su koncentrirani na prepoznavanje objekata.

6.2. Segmentacija bazirana na konturama

Sedamdesetih godina dvadesetog stoljeća na temelju neuropsiholoških eksperimenata proizišla je Marr-ova teorija. Teza teorije je da su granice objekata najvažnije informacije koje povezuju intenzitete svjetlosti s interpretacijom slike [25].

Brid je signifikantna lokalna promjena slike i očekuje se da će lokalne promjene intenziteta biti najveće na granicama objekata. Segmentacija bazirana na konturama je skupina metoda baziranih na informacijama o lokalnim promjenama intenziteta. To je jedan od najranijih pristupa segmentaciji, a metoda je i danas vrlo aktualna.

Segmentacija bazirana na bridovima se sastoji od dva koraka:

- detekcija bridova,
- segmentacija konturama.

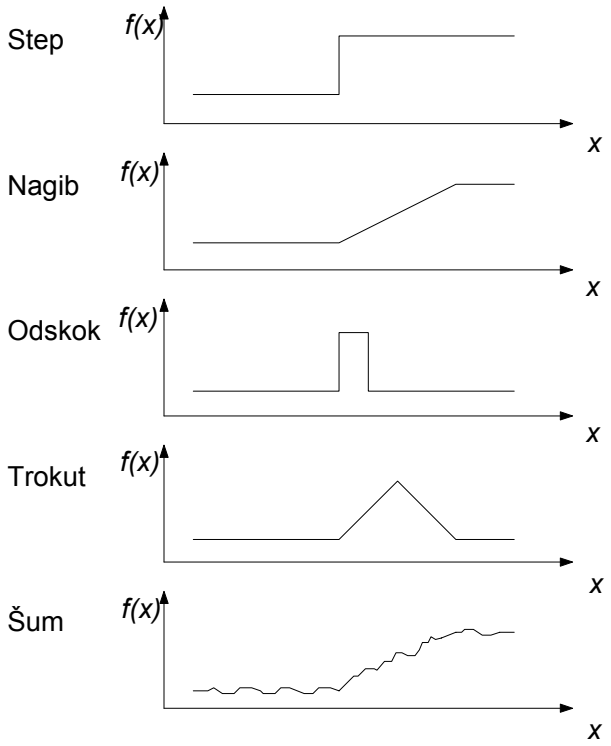
Bridovi su značajke piksela. Algoritmi kojima se detektiraju ne uključuju konceptualna znanja o sadržaju prizora. Rezultat procesa detekcije bridova nije ništa drugo nego popis piksela kojima je dodijeljen atribut brida.

Postupak segmentacije se sastoji u kreiranju kontura. Konture su reprezentacije granica regija, a generiraju se povezivanjem ili interpoliranjem naglašenih bridova.

Kontura je za razliku od brida značajka slike. Reprezentirane su na "pomalo" simbolički način. Opisuju pozicije i oblik dijelova prizora. Konture se ne pridružuju objektima jer nije uključeno znanje o identitetima objekata. Ipak, ponekad se koristi znanje o oblicima na prizoru i načinu na koji se ti oblici projiciraju na sliku. Zbog toga se ovi algoritmi klasificiraju kao vizijski algoritmi srednje razine [30].

6.2.1. Detekcija bridova

Brid je signifikantna lokalna promjena intenziteta slike. Povezuje se s diskontinuitetom intenziteta slike ili diskontinuitetom prve derivacije intenziteta. Tipični oblici diskontinuiteta intenziteta piksela, po jednoj dimenziji digitalne slike, su ilustrirani slikom Sl.-16. Ovi oblici nazivaju se profili bridova. Algoritmi za detekciju bridova najčešće se podešavaju za određeni profil brida.



Sl.-16. Oblici diskontinuiteta intenziteta piksela po jednoj dimenziji digitalne slike.

Detekcija bridova se sastoji od sljedećih koraka:

- filtriranje,
- kvantiziranje diskontinuiteta,
- odabir bridova,
- lokalizacija.

6.2.1.1. Filtriranje slike

Filtriranje je postupak kojim se unapređuje kvaliteta digitalne slike. Pri tome se na kvalitetu gleda u kontekstu primjene slike.

Ako će sliku gledati ljudi, najčešće je potrebno unaprijediti kontrast ili izoštriti sliku kako bi se što više naglasile razlike intenziteta. Na slici Sl.-17 prikazana je slika s niskim kontrastom te slika s uvećanim kontrastom.

Sl.-17. Slika sa niskim kontrastom i poboljšana slika.

S druge strane, ako će se slika obrađivati u sustavu strojnog vida najveći problem predstavlja šum. Pod šumom se razumijevaju slučajne varijacije intenziteta piksela. Šum je posljedica svih elektromagnetskih smetnji i pogrešaka pri analogno – digitalnim pretvorbama prilikom kreiranja i dobave slike.

Sl.-18. Slika sa šumom.

Problem šuma posebno dolazi do izražaja kod detekcije bridova, jer šum uzrokuje velike lokalne promjene intenziteta koje se mogu pogrešno interpretirati kao bridovi. Slika pokvarena šumom, poput slike Sl.-18, mora se filtrirati kako bi se uklonio šum bez obzira što je čovjeku slika puno preglednija i jasnija od slike s niskim kontrastom (Sl.-17).

U nastavku ovog rada pod filtriranjem slike će se razumijevati operacije uklanjanja šuma, jer filtriranje radi povećanja kontrasta nije toliko zanimljivo u kontekstu strojnog vida.

Konvolucija i konvolucijske maske

Operacije filtriranja su linearne transformacije kojima se izračunavaju vrijednosti intenziteta izlazne slike $h(i,j)$ kao linearne kombinacije intenziteta u susjedstvu piksela $f(i,j)$ originalne, odnosno ulazne slike.

Odziv linearnog sustava pobuđenog impulsnom funkcijom $\delta(x,y)$ je funkcija $g(x,y)$. Ukoliko je funkcija $g(x,y)$ neovisna o položaju impulsne pobude, sustav se može u potpunosti opisati

g(x,y) funkcijom. Izlaz iz takvog linearnog sustava *h(x,y)* je konvolucija ulazne funkcije *f(x,y)* s funkcijom *g(x,y)*. Konvolucija je definirana izrazima (17) i (18).

$$h(x,y) = f(x,y) \otimes g(x,y) \tag{17}$$

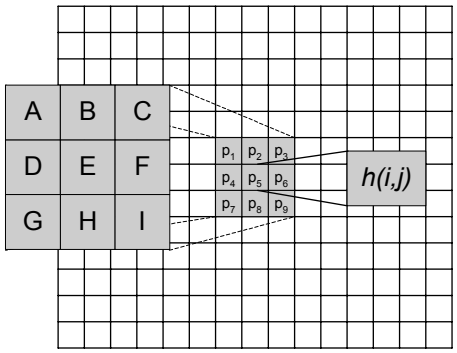
$$h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')g(x-x',y-y')dx'dy' \tag{18}$$

Konvolucija diskretnih funkcija je definirana izrazima (19) i (20).

$$h[i,j] = f[i,j] \otimes g[i,j] \tag{19}$$

$$h[i,j] = \sum_{k=1}^{n}\sum_{l=1}^{m} f[k,l]\, g[i-k, j-l] \tag{20}$$

Ako su diskretne funkcije *f* i *h* slike, konvolucija je ponderiranje i sumiranje piksela u okolici svakog piksela. Diskretna funkcija *g[i,j]* se naziva konvolucijska maska. Vrijednost svakog piksela slike *h* se dobiva tako da se konvolucijska maska "primjeni" na svaki pojedini piksel. Rezultat dobiven primjenom konvolucijske maske na određeni piksel se naziva odgovor. Postupak konvolucije maskom dimenzija 3x3 ilustriran je slikom Sl.-19.



$$h[i,j] = Ap_1 + Bp_2 + Cp_3 + Dp_4 + Ep_5 + Fp_6 + Gp_7 + Hp_8 + Ip_9$$

Sl.-19. Konvolucija digitalne slike

Linearni filtri su svi filtri koji koriste ovaj model za računanje novih vrijednosti piksela. Nelinearni filtri su svi ostali koji koriste druge modele za uprosjećavanje piksela na temelju njihove okolice.

Filtar aritmetičke sredine

Filtar aritmetičke sredine je jednostavan filtar koji računa novu vrijednost piksela kao aritmetičku sredinu njegove okolice. Za primjer okolice veličine 3x3 konvolucijska maska *g* dana je izrazom (21).

$$g = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{21}$$

Ova jednostavna maska se može unaprijediti tako da se poveća utjecaj središnjih piksela. Izrazima (22) i (23) su dane dvije maske dimenzije 3x3 s takvim svojstvima:

$$g = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{22}$$

$$g = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{23}$$

Gausov filtar

Gausov filtar je linearni filtar kod kojih iznosi pondera u konvolucijskoj maski odgovaraju gausovoj funkciji.

Diskretni oblik dvodimenzionalne gausove funkcije je dan izrazom (24), pri čemu σ određuje "širinu" gausove funkcije.

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}} \tag{24}$$

Izrazom (25) je dana gausova maska dimenzije 7x7. Ponderi iz maske su dobiveni uvrštavanjem indeksa $i = [-3,3]$, $j = [-3,3]$ u gornji izraz, pri čemu je uzeto da je $\sigma^2 = 2$.

$$g[i,j] = \frac{1}{1115}\begin{bmatrix} 1 & 4 & 7 & 10 & 7 & 4 & 1 \\ 4 & 12 & 26 & 33 & 26 & 12 & 4 \\ 7 & 26 & 55 & 71 & 55 & 26 & 7 \\ 10 & 33 & 71 & 91 & 71 & 33 & 10 \\ 7 & 26 & 55 & 71 & 55 & 26 & 7 \\ 4 & 12 & 26 & 33 & 26 & 12 & 4 \\ 1 & 4 & 7 & 10 & 7 & 4 & 1 \end{bmatrix} \tag{25}$$

Median filtar

Primjer jednostavnog nelinearnog filtra je median filtar. Median filtar ne izračunava novu vrijednost piksela kao linearnu kombinaciju. Nova vrijednost piksela je median iznosa u lokalnoj okolici.

Rotacijske maske

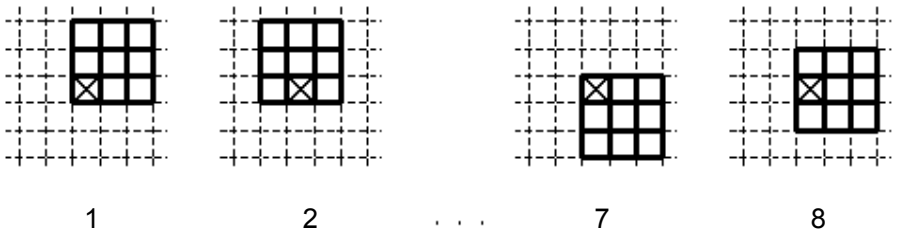
Filtriranje šuma lokalnim uprosjećivanjem rezultira zamućivanjem oštrih granica na slici. To se negativno manifestira prilikom detekcije bridova jer su smanjeni lokalni diskontinuiteti. Razvijene su neke metode koje filtriraju sliku bez zamućivanja rubova. Ti algoritmi se temelje na ideji da se u proces uprosjećivanja uključuju samo pikseli sa sličnim svojstvima. Očekuje se da pikseli sa sličnim svojstvima pripadaju istoj regiji pa uprosjećivanje neće zamutiti granice regija.

Filtriranje rotacijskim maskama je postupak s tim svojstvima. Metoda izbjegava zamućivanje bridova tako da traži homogeni dio pikselovog susjedstva i uprosjećuje samo na temelju tog područja.

Standardna devijacija σ^2 je mjera homogenosti regije. Za regiju R sa n elemenata i ulaznu sliku $f(x,y)$ računa se prema izrazu (26).

$$\sigma^2 = \frac{1}{n}\sum_{(i,j)\in R}\left[f(i,j) - \frac{1}{n}\sum_{(i',j')\in R} f(i',j')\right] \qquad (26)$$

Algoritam za filtriranje rotacijskom maskom rotira konvolucijsku masku oko svakog piksela. Konvolucijska maska će se primijeniti u položaju s minimalnim σ^2. Postupak je prikazan slikom Sl.-20.



Sl.-20. Rotacijske maske.

Filtriranje rotacijskim maskama je vrlo učinkovito jer je rezultat filtriranja u stvari slika s izoštrenim granicama [23].

6.2.1.2. Kvantiziranje diskontinuiteta

Bridovi su signifikantne lokalne promjena intenziteta slike, odnosno područja na slici s izraženim diskontinuitetom intenziteta piksela. Brid je svojstvo svakog pojedinog piksela i kvantificira sljedećim metodama:

- gradijentom,
- laplaceovim operatorom,
- parametarskim modelom slike.

Gradijent

Gradijent je mjera promjene funkcije. To je dvodimenzionalni ekvivalent prve derivacije i definiran je kao vektor prema izrazu (27).

$$\vec{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{27}$$

Zanimljiva su dva svojstva vektora gradijenta, iznos i smjer. Iznos je definiran izrazom (28).

$$G[f(x,y)] = \sqrt{G_x^2 + G_y^2} \tag{28}$$

Očekuje se da lokalni maksimumi gradijenta odgovaraju granicama među regijama na slici. Profili funkcija na mjestima koja odgovaraju bridovima i odgovarajuće prve i druge derivacije su prikazani slikom Sl-21. Pod profilom funkcije slike se podrazumijeva funkcija kojom su opisani intenziteti piksela jednog retka.

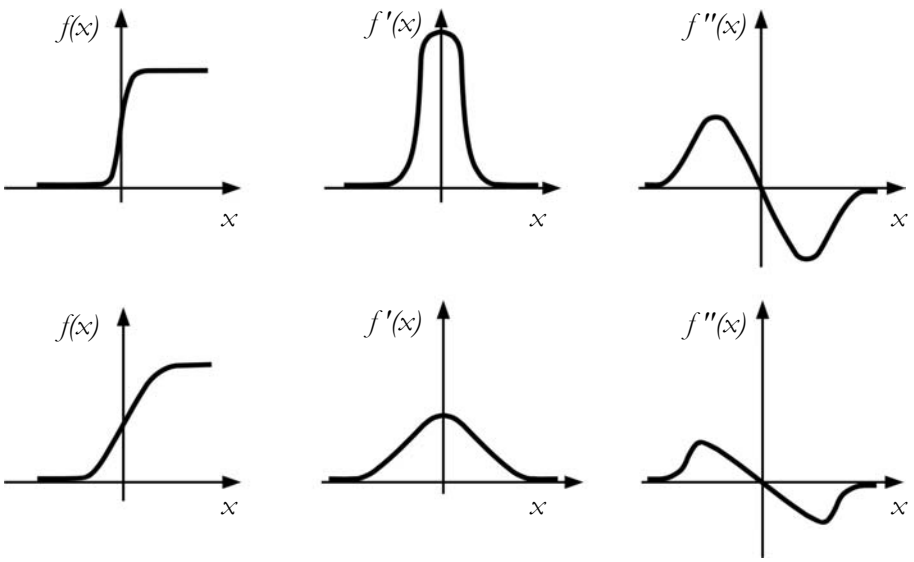
Budući da su slike opisane diskretnim vrijednostima intenziteta piksela, a ne kontinuiranim funkcijom, gradijent se aproksimira diferencijama susjednih piksela. Izrazi za aproksimaciju gradijenta su dani izrazima (29) i (30), a odgovarajuće konvolucijske maske izrazima (31) i (32).

$$G_x[i,j] \cong f[i,j+1] - f[i,j] \tag{29}$$

$$G_y[i,j] \cong f[i,j] - f[i+1,j] \tag{30}$$

$$G_x = \begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array} \tag{31}$$

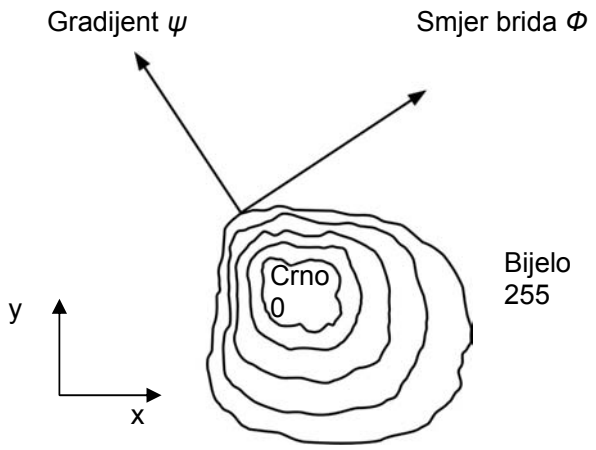
$$G_x = \begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline \end{array} \qquad (32)$$



Sl-21. Profili funkcija slike na mjestima koja odgovaraju bridovima i odgovarajuće prve i druge derivacije.

Smjer gradijenta definiran je kutom ψ koji se mjeri u odnosu na os x. On ukazuje na smjer maksimalnog rasta funkcije i okomit je na smjer brida Φ. Smjer brida i smjer gradijenta za sliku u čijoj sredini se nalazi crno područje, a rubovima bijelo je prikazan slikom Sl.-22. Smjer gradijenta je definiran izrazom (33).

$$\psi(x,y) = tan^{-1}\left(\frac{G_y}{G_x}\right) \qquad (33)$$



Sl.-22. Smjer brida i smjer gradijenta.

Razvijen je niz operatora za kvantificiranje diskontinuiteta baziranih na gradijentu. Maske su najčešće kvadratne[18] s neparnim brojem redaka i stupaca kako bi se aproksimirao gradijent u središtu piksela na koji je maska primijenjena.

Smjer brida se može odrediti i "rotiranjem" konvolucijske maske. U tom slučaju smjer brida za određeni piksel odgovara orijentaciji maske s najvećom apsolutnom vrijednošću odgovora.

Najčešći operatori ovog tipa su:

Robertsov operator:

Magnituda promjene intenziteta *M* je definirana izrazom (36).

$$G_x = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad (34)$$

$$G_y = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array} \qquad (35)$$

$$M[f(x,y)] = |G_x| + |G_y| \qquad (36)$$

[18] Jednaki broj stupaca i redaka.

Prewittov operator

Gradijent se ispituje sa osam konvolucijskih maski za svaki piksel. Svaka od maski daje maksimalni odgovor za jednu orijentaciju brida. Izrazi (37), (38) i (39) definiraju tri maske. Ostalih pet se dobiva rotiranjem u smjeru kazaljke po 45°. Magnituda promjene intenziteta M je definirana izrazom (40).

$$m_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{37}$$

$$m_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \tag{38}$$

$$m_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{39}$$

$$M[f(x,y)] = max(m_i) \tag{40}$$

Sobelov operator

Kao i kod Prewittovog operatora, gradijent se ispituje s osam konvolucijskih maski za svaki piksel. Svaka od maski daje maksimalni odgovor za jednu orijentaciju brida. Prve tri od osam maski su definirane izrazima (41), (42) i (43).

$m_1 =$

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

(41)

$m_2 =$

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

(42)

$m_3 =$

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

(43)

Laplaceov operator

Laplaceov operator je dvodimenzionalni ekvivalent druge derivacije funkcije. Prva derivacija funkcije intenziteta slike ima ekstrem na mjestu koje odgovara granici na slici. Druga derivacija je na tom mjestu jednaka nuli (Sl-21).

Laplaceov operator je definiran izrazom (44).

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (44)$$

Aproksimacija Laplaceovog operatora za diskretne funkcije je dana izrazima (45) i (46), a odgovarajuća konvolucijska maska izrazom (47).

$$\frac{\partial^2 f}{\partial x^2} = f[i+1, j] - 2f[i, j] + f[i-1, j] \qquad (45)$$

$$\frac{\partial^2 f}{\partial x^2} = f[i+1, j] - 2f[i, j] + f[i-1, j] \qquad (46)$$

$$\nabla^2[i,j] = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad (47)$$

Očekuje se da odgovori čija je vrijednost bliska nuli odgovaraju bridovima na slici. Pri tome se nule koje su uzrokovali bridovi moraju razlučiti od trivijalnih nula proizašlih iz uniformnih regija jer je druga derivacija konstante također nula.

Laplaceov operator je izotropski operator. Pod time se podrazumijeva da je neovisan o orijentacijama bridova. "Osjetljiviji" je od gradijenta, budući da je lakše detektirati nule nego maksimume funkcija. No, kako se radi o operatoru druge derivacije koji jako naglašava diskontinuitet, vrlo je osjetljiv na šum.

LoG Operator

LoG operator je često korištena inačica Laplaceovog operatora. Ime mu je kratica engleskog naziva Laplacian of Gaussian (Laplace Gausa) što ga potpuno opisuje. Radi se o operatoru koji integrira gausov filtar i laplaceov operator.

Operatori druge derivacije su vrlo osjetljivi na šum te se prije njihove primjene slika uvijek mora "jako" filtrirati. Izraz (48) opisuje primjenu laplace-ovog operatora na sliku filtriranu gausovim filtrom.

$$h[i,j] = (f[i,j] \otimes g[i,j]) \otimes \nabla^2[i,j] \qquad (48)$$

pri čemu je:

$h[i,j]$ – Rezultat konvolucija koji kvantizira diskontinuitete.

$f[i,j]$ – Originalna slika.

$\nabla^2[i,j]$ – Laplaceov operator.

$g[i,j]$ – Gausov filtar.

Linearnost operacija omogućuje promjenu redoslijeda pa je izraz (49) potpuno jednak izrazu (48).

$$h[i,j] = f[i,j] \otimes \left(\nabla^2[i,j] \otimes g[i,j]\right) \qquad (49)$$

LoG maska se dobiva primjenom Laplace-ovog operatora na Gausov filtar. LoG konvolucijska maska je definirana izrazom (50).

$$LoG = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \quad (50)$$

Parametrički modeli slike

Kvantiziranje diskontinuiteta parametričkim modelima se bazira na ideji aproksimacije diskretne funkcije intenziteta slike jednom ili više kontinuiranih funkcija. Za razliku od diskretnih modela slike, na taj se način kontinuirano opisuju svojstva slike po čitavoj površini. Diskretni modeli koncentriraju informacije o površini jednog piksela u samo jednu točku.

Proširenjem domene informacija kojom se opisuje slika omogućuje se preciznija analiza. Pod preciznijom analizom se podrazumijeva mogućnost određivanja svojstava slike kontinuirano na mjestima između središta piksela. Na primjer, bridom se ne mora proglasiti čitav piksel odnosno postaviti u njegov centar, nego u bilo koju točku.

Aproksimacija diskretne funkcije slike samo jednom kontinuiranom funkcijom je vrlo složena, a gotovo i nemoguća zbog velikog broja piksela, budući da broj točaka koji se aproksimira funkcijom uvjetuje red polinoma.

Facetni modeli slike opisuju sliku nizom funkcija. Svaka funkcija opisuje jedan dio slike, facetu koja uobičajeno opisuje područje veličine jednog piksela. Funkcije faceta su relativno jednostavne budući da su dobivene na temelju vrijednosti intenziteta nekolicine piksela u okolici područja koje se opisuje. Najčešće su to bilinearne, bikvadratne ili bikubične funkcije [28].

Koeficijenti k_i bikubične funkcije (51) mogu se dobiti primjenom konvolucijskih maski centriranih na piksel čije se područje aproksimira. Izrazima (52), (53) i (54) dane su maske za koeficijente k_1, k_2 *i* k_{10} [29].

$$f(x,y) = k_1 + k_2 x + k_3 y + k_4 x^2 + k_5 xy + k_6 y^2 + k_7 x^3 + k_8 x^2 y + k_9 xy^2 k_{10} y^3 \quad (51)$$

$$k_1 = \frac{1}{175}\begin{bmatrix} -13 & 2 & 7 & 2 & -13 \\ 2 & 17 & 22 & 17 & 2 \\ 7 & 22 & 27 & 22 & 7 \\ 2 & 17 & 22 & 17 & 2 \\ -13 & 2 & 7 & 2 & -13 \end{bmatrix} \quad (52)$$

$$k_2 = \frac{1}{420}\begin{bmatrix} 31 & -5 & -17 & -5 & -31 \\ -44 & -62 & -68 & -62 & -44 \\ 0 & 0 & 0 & 0 & 0 \\ 44 & 62 & 68 & 62 & 44 \\ -31 & 5 & 17 & 5 & -31 \end{bmatrix} \quad (53)$$

$$k_{10} = \frac{1}{60}\begin{bmatrix} -1 & 2 & 0 & -2 & 1 \\ -1 & 2 & 0 & -2 & 1 \\ -1 & 2 & 0 & -2 & 1 \\ -1 & 2 & 0 & -2 & 1 \\ -1 & 2 & 0 & -2 & 1 \end{bmatrix} \quad (54)$$

Detekcija bridova na slikama opisanim funkcijama svodi se na pronalaženje ekstrema prvih derivacija funkcija. Ukoliko prva derivacija funkcije ima ekstrem na području piksela kojeg opisuje, točka ekstrema ili čitav piksel se proglašavaju bridom.

6.2.2. Segmentacija konturama

Rezultati postupka detekcije bridova ne mogu se direktno iskoristiti za segmentaciju slike. Bridovi su tek lokalno svojstvo slike, odnosno svojstvo piksela. Korak prema simboličkom opisu slike predstavljaju konture. Konture su objekti koji opisuju granice regije.

Općenito, konture mogu biti zatvorene i otvorene. Zatvorene konture u potpunosti opisuju granicu regije i na taj način segmentiraju sliku. Otvorene konture odgovaraju samo jednom segmentu granice regije te ne definiraju regiju u potpunosti. Ipak, otvorene konture mogu biti dostatne kod pojedinih aplikacija strojnog vida. Konture mogu biti reprezentirane kao uređeni popis bridova ili matematički/analitički kao krivulje.

Algoritmi za kreiranje kontura se mogu podijeliti po korištenoj strategiji na lokalne i globalne.

Lokalni algoritmi kreiraju konture na temelju lokalnih svojstva slike, odnosno bridova u ograničenom području. Potrebno je vrlo malo znanja/informacija o svojstvima slike odnosno o objektima koji se na njoj nalaze.

Globalni algoritmi kreiraju konture na temelju bridova na čitavoj slici. Konture se kreiraju grupiranjem bridova na temelju zajedničkih parametara .

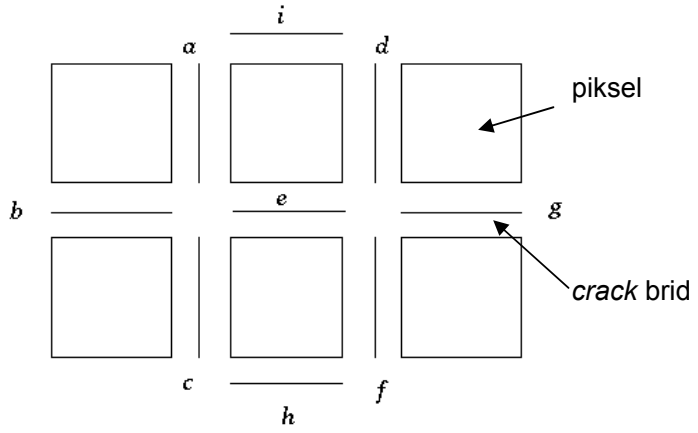
Algoritmi za definiranje kontura suočeni su s dva osnovna problema. Oba su posljedica uvjeta osvjetljenja na prizoru te šumova koji se javljaju prilikom dobave slike.

Prvi problem je povezan s otkrivanjem lažnih bridova. Lažni bridovi su lokacije na slici ili pikseli s izraženim svojstvom brida iako ne odgovaraju stvarnim granicama regija. Drugi problem se sastoji u prepoznavanju granica regija na lokacijama gdje svojstvo brida nije izraženo.

U daljnjem tekstu bit će opisano nekoliko metoda za definiranje kontura.

6.2.2.1. "Crack edge relaxation" algoritam

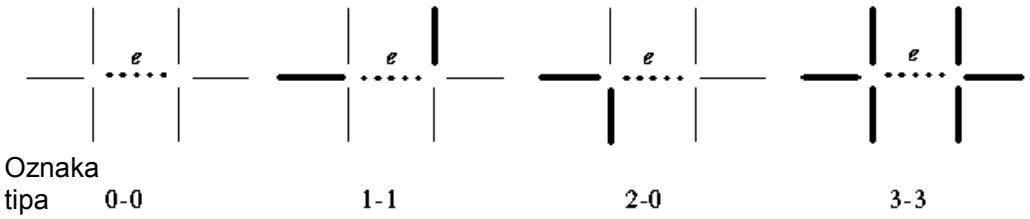
Crack edge relaxation je lokalni iterativni algoritam za određivanje kontura ulančavanjem *crack* bridova. Specifičnost *Crack* bridova jest da se nalaze na granicama susjednih piksela. *Crack* bridovi su prikazani slikom Sl.-23.



Sl.-23. Crack bridovi.

Za bridove se definira značajka snage. Značajka snage brida je mjera vjerojatnosti da brid odgovara granici objekta na slici. Snaga brida se iterativno povećava ili smanjuje, a bridovi se na temelju nje uključuju u daljnje iteracije, odnosno u rješenje.

Osim snage definira se i oznaka tipa brida. Oznaka tipa je formata ***i-j***, pri čemu su *i* i *j* brojevi susjednih bridova na svakom od krajeva promatranog brida. Slikom Sl.-23 prikazane su četiri različite konfiguracije brida (*e*), njemu susjedni bridovi (označeni su debljim linijama) te odgovarajuće oznake tipa.



Sl.-24. Evaluacija bridova na temelju broja susjednih bridova na svakom kraju.

U svakom iteracijskom koraku se najprije određuju tipovi bridova. Na temelju oznake tipa mijenja se snaga brida pa se pojedini bridovi mogu isključiti iz skupa rješenja. Tablica 2 prikazuje utjecaj oznake tipa na snagu brida.

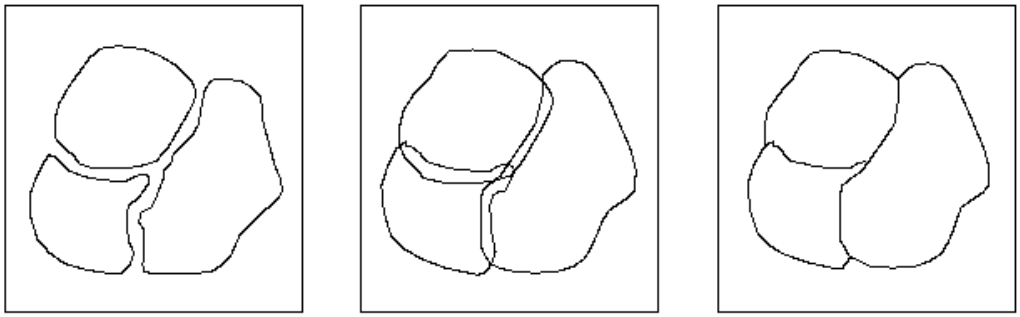
Tablica 2. Utjecaj oznake tipa na snagu brida

Oznaka tipa	Pretpostavljena uloga	Utjecaj na snagu brida
0-0	Izolirani brid	Negativan utjecaj
0-2, 0-3	Slijepi brid	Negativan utjecaj
0-1	Nesigurno	Slab pozitivan utjecaj
1-1	Dio konture	Jak pozitivan utjecaj
1-2, 1-3	Sjecište kontura	Srednji pozitivan utjecaj
2-2, 2-3, 3-3	Spoj među konturama	Nema utjecaja

6.2.2.2. Border tracing algoritam

Border tracing algoritam je lokalni algoritam za određivanje kontura na slikama s poznatim definicijama regija. Regije su najčešće definirane jednim od *treshold* algoritama.

Konture se generiraju po granicama regija. Granice regija se mogu definirati na različite načine, što je ilustrirano slikom Sl.-25. Unutrašnju granicu čine pikseli uz rub regije, a pripadaju regiji. Vanjsku granicu čine pikseli uz samu regiju. Svojstvo produžene granice je da ima isti oblik i veličinu kao i regija koju opisuje te da su objekti međusobno odvojeni samo jednom granicom.



Unutrašnja granica Vanjska granica Produžena granica

Sl.-25 Vrste granica.

S obzirom na granicu koja se koristi pri generiranju kontura međusobno se razlikuju i *Border tracing* algoritmi. Sljedeći *border tracing* algoritam definira konture na temelju unutrašnje granice [31].

1. Pretraživati sliku počevši od gornjeg lijevog ugla dok se ne pronađe nova regija. Prvi pronađeni piksel regije P_0 ima minimalni broj stupca od svih piksela s minimalnim brojem retka i on je početni piksel granice regije. Definirati varijablu *smjer* koja pohranjuje vrijednost pomaka iz prethodnog koraka. Mogući smjerovi pomaka i njihove vrijednosti su prikazani na slici Sl.-26. Dodijeliti varijabli smjer vrijednost:

   Smjer = 0

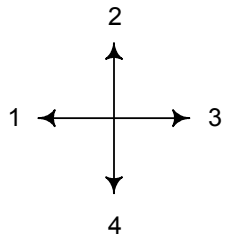
2. Pretraži 3x3 susjedstvo trenutnog piksela u smjeru suprotnom od kazaljke na satu, počevši od piksela:

   (*Smjer* + 3) mod 4

   Prvi pronađeni piksel iste vrijednosti kao i trenutni piksel je novi element granice P_n. Zapisati vrijednost pomaka u varijablu *Smjer*.

3. Ako je trenutni element granice P_n jednak drugom elementu P_1, i prethodni element granice P_{n-1} jednak elementu P_0, prekinuti postupak. Inače, ponoviti korak 2.

4. Kontura određena unutrašnjom granicom je definirana pikselima P_0 ... P_{n-2}.

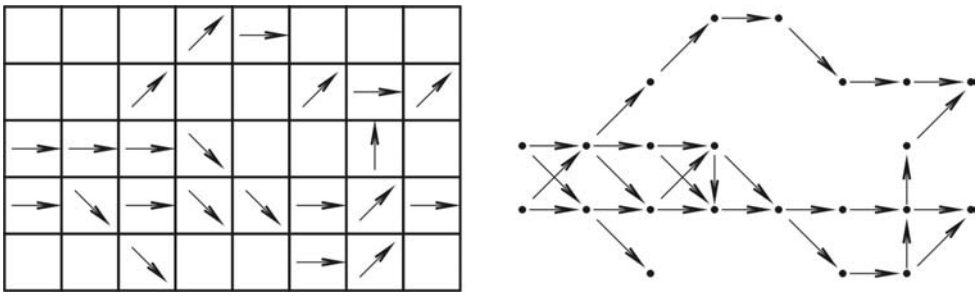


Sl.-26. Mogući smjerovi pomaka i njihove vrijednosti u *border tracing* algoritmu za unutrašnje granice.

6.2.2.3. Određivanje kontura pretraživanjem grafom

Pod grafom se podrazumijeva općenita struktura sastavljena od čvorova međusobno povezanih lukovima. Lukovi su orijentirani i za svaki je definirana težinska vrijednost/funkcija.

Čvorovi grafa predstavljaju piksele s izraženim svojstvom brida, a lukovi se definiraju između susjednih piksela. Na slici Sl.-27 prikazani su pikseli s izraženim svojstvom brida i naznačenim smjerom brida te graf konstruiran na temelju njih.



Sl.-27. Orijentirani graf (desno) konstruiran na temelju signifikantnih bridova (lijevo).

Definicija konture se sastoji od pronalaženja optimalnog puta kroz graf, od početne do krajnje točke konture. Početna i krajnja točka konture u općenitom slučaju nisu poznate. Stoga se moraju aproksimirati heurističkim metodama [31].

Optimalnim putem se smatra put s minimalnom funkcijom cilja. Funkcija cilja mora biti neovisna o ukupnoj duljini konture.

Neke od općenito prihvaćenih funkcija cilja su utemeljene na:

- snazi bridova u konturi,
- zakrivljenosti konture,
- udaljenosti od očekivanog položaja konture,
- procijenjenoj udaljenosti od završetka konture,
- ...

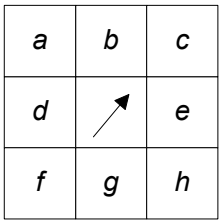
Sljedeći algoritam definira konturu na temelju poznate početnog (n_A) i završnog (n_B) čvora.

1. Proširiti početni čvor n_A i postaviti sve čvorove koji ga slijede skupa sa pokazivačima na prethodni čvor u listu *OTVORENO*. Izračunati funkciju cilja za svaki prošireni čvor.

2. Ako je lista *OTVORENO* prazna, prekini izvođenje zbog pogreške. Pronaći i ukloniti čvor n_i iz liste *OTVORENO* koji rezultira s najmanjom funkcijom cilja. Ako je $n_i = n_B$ postupak je završen.

3. Ekspandirati čvor n_i i postaviti sve čvorove koji ga slijede skupa sa pokazivačima na prethodni čvor u listu *OTVORENO*. Izračunati funkciju cilja za svaki prošireni čvor. Otići na korak 2.

Definiranje kontura bez znanja o početnim i krajnjim točkama je složenije. Konture se definiraju primjenom bidirekcijskog heurističkog pretraživanja. Na slici Sl.-28 prikazano je 3x3 susjedstvo piksela s izraženim svojstvom brida. Za piksele ***b***, ***c*** i ***e*** se uzima da su ispred brida, a ***d***, ***f***, i ***g*** da su iza brida.



Sl.-28. Bidirekcijsko heurističko pretraživanje.

Sljedeći algoritam definira konture primjenom bidirekcijskog heurističkog pretraživanja [30].

Ponavljati

Pretraživati sliku dok se ne pronađe čvor (piksel) s najizraženijim svojstvom brida (većim od određenog praga). Označiti pronađeni čvor kao inicijalnu i trenutnu točku konture.

Ponavljati

Ekspandirati konturu dodavanjem čvora ispred trenutnog čvora koji rezultira s najmanjom funkcijom cilja (isto kao i prethodni algoritam).

Dodani čvor postaje trenutni čvor.

Dok se mogu pronaći čvorovi.

Početna točka ponovno postaje trenutna točka.

Ponavljati

Ekspandirati konturu dodavanjem čvora iza trenutnog čvora koji rezultira s najmanjom funkcijom cilja (isto kao i prethodni algoritam).

Dodani čvor postaje trenutni čvor.

Dok se mogu pronaći čvorovi.

Označeni čvorovi čine konturu.

Dok postoje nedodijeljeni čvorovi.

6.2.2.4. "Hough transformation" metoda

"Hough transformation" je najčešća metoda za određivanje kontura na temelju globalnih informacija o bridovima. Ona postavlja problem segmentacije kao postupak pronalaženja kontura poznatog oblika. Pod poznatim oblikom kontura se ovdje podrazumijeva da je poznat analitički model konture, na primjer da se radi o pravcu, kružnici i slično. Problem se svodi na definiranje parametara analitičkog modela.

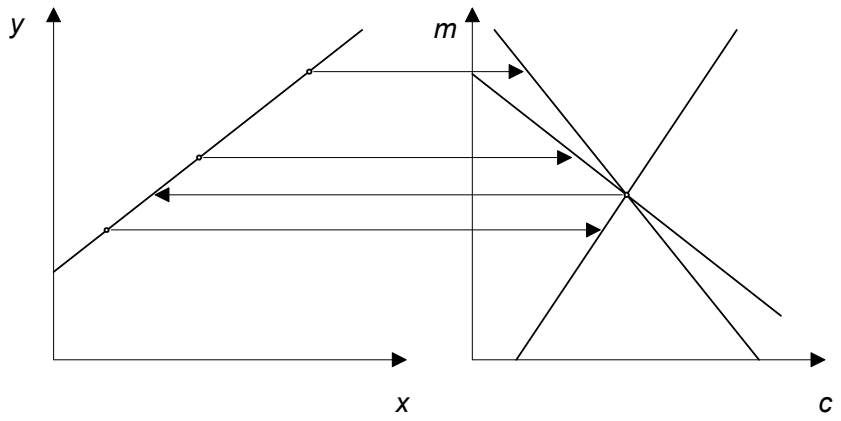
Originalna *"Hough transformation"* metoda je razvijena za otkrivanje ravnih linija i jednostavnih analitičkih krivulja. Metoda će biti prikazana na modelu za otkrivanje pravaca na slici. Jednadžba pravca je dana izrazom (55).

$$y = mx + c \qquad (55)$$

U jednadžbi pravca (55) x i y su koordinate a m i c su parametri. Međusobni odnos koordinata je jednoznačno određen parametrima. Jednadžbu (55) možemo napisati na sljedeći način:

$$c = y - mx \qquad (56)$$

Uz pretpostavku da su u jednadžbi (56) x i y konstante, jednadžba predstavlja pravac u m-c prostoru. Točka u x-y prostoru (konstanta) jednoznačno određuje pravac u m-c prostoru i obrnuto. Točka u m-c prostoru jednoznačno određuje pravac u x-y prostoru. Mapiranje iz x-y prostora slike u c-m parametarski prostor i obrnuto je prikazano na slici Sl.-29.



Sl.-29. Mapiranje iz *x-y* prostora slike u *c-m* parametarski prostor i obrnuto.

Osnovna ideja procesa detekcije je da se svi pikseli kandidati za elemente linije transformiraju u c-m parametarski prostor. Pikseli kandidati su svi pikseli s izraženim svojstvom brida. Točke u parametarskom prostoru u koje se preslikalo najviše piksela definiraju pravce na slici.

Parametri c i m su općenito kontinuirane varijable. No, parametarski prostor se diskretizira na pravokutnu strukturu ćelija čija gustoća ovisi o traženoj točnosti. Elementi strukture se nazivaju akumulatorske ćelije $A(m, c)$.

Za svaki brid – piksel se određuju parametri m i c koji predstavljaju linije u parametarskom prostoru. Linija u parametarskom prostoru povećava vrijednost akumulatorske ćelije $A(m, c)$ kroz koju prolazi.

Pravci na slici su jednoznačno određeni akumulatorskim ćelijama koje na kraju procesa imaju najveću vrijednost, stoga se detekcija kontura svodi na detekciju lokalnih maksimuma u diskretiziranom parametarskom prostoru.

Eksplicitna jednadžba pravca (55) nije pogodna za stvarno izvođenje opisanog postupka budući da parametar m kod vertikalnih pravaca teži u beskonačnost. Da bi se izbjegao ovaj problem pravac se analitički opisuje prema izrazu (57).

$$s = x \cos\theta + y \sin\theta \qquad (57)$$

Oblik analitičke funkcije uvjetuje način na koji se točka preslikava u parametarski prostor. Ako je pravac opisan izrazom (57), točke se u parametarski prostor $s - \theta$ preslikavaju kao sinusoide.

Opisana metoda se može primijeniti i za detekciju složenijih krivulja. Svaki parametar analitičke formule krivulje predstavlja jednu dimenziju parametarskog prostora. Povećanje dimenzija parametarskog prostora znatno povećava obujam proračuna jer se time eksponencijalno povećava broj akumulatorskih ćelija.

Općenita analitička reprezentacija krivulje dana je izrazom (58).

$$f(x, a) = 0 \qquad (58)$$

Pri tome x predstavlja vektor varijabli, a a predstavlja vektor parametara. U slučaju analize slike vektor varijabli je uvijek jednak (x, y). Vektor parametara ovisi o krivulji. Na primjer, za kružnicu vektor ima tri člana (a, b, r).

Slijedi općeniti *"Hough transformation"* algoritam.

1. Kvantizirati parametarski prostor u granicama vektora a.
2. Postavi sve akumulatorske ćelije parametarskog prostora na 0.
3. Za svaku točku slike(x_i, y_i) s izraženim svojstvom brida povećaj sve ćelije parametarskog prostora za koje vrijedi $f(x, a) = 0$.
4. Pronađi lokalne maksimume u parametarskom prostoru [31].

Prednost *"Hough transformation"* metode je robusnost segmentacije. Proces nije osjetljiv na slike degradirane šumom, sjenama ili bljeskovima.

Osnovni nedostatak metode je da postaje računarski ekstenzivna pri pronalaženju složenijih krivulja s više parametara.

6.2.2.5. Reprezentacije kontura

Oblik reprezentacije konture je često određen algoritmom kojim je kontura kreirana. Na primjer, reprezentacija "*chain"* kodom (opisan u nastavku) je prirodna reprezentacija koju producira "*Border tracing"* algoritam. Parametarski opisane krivulje su prirodni rezultat *"Hough transformation"* metode.

Način reprezentacije mora biti koncizan, ali također mora optimirati proces prepoznavanja. Redundantno pohranjivanje informacija je stoga prihvatljivo ukoliko značajno doprinosi brzini prepoznavanja [30].

Parametarska reprezentacija

Najjednostavniji oblik reprezentiranja kontura je reprezentiranje njihovim geometrijskim parametrima. Na primjer, pravocrtni segment se može reprezentirati pomoću četiri parametra:

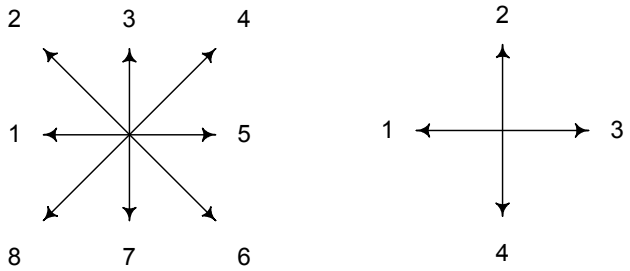
- s dvije koordinate početne i krajnje točke,
- s koordinatama sredine segmenta, njegovom dužinom i nagibom,
- ...

Kružni luk zahtjeva 6 parametara. To mogu biti, na primjer, parovi koordinata početne, krajnje točke i točke na luku.

Nedostatak ove metode jest što opisuje samo pravilne geometrijske strukture sastavljene od linija, kružnih lukova, parabola i slično.

"Chain" kodovi

"Chain" kodovi su notacija kojom se zapisuje lista piksela uzduž konture. *"Chain"* kodom se specificira smjer konture na mjestu svakog piksela u listi. Smjerovi su kvantizirani u osam ili četiri moguća smjera, ovisno o definiciji konektivnosti piksela kako je to prikazano na slici Sl.-30.



Sl.-30. "Chain" kodovi za reprezentiranje smjerova između susjednih piksela na konturi.

Kontura opisana chain kodom se sastoji od definicije položaja prvog piksela u nizu i liste koja sadrži *"Chain"* kodove, a svakim kodom je jednoznačno definiran položaj sljedećeg piksela.

"Chain" kodovi imaju nekoliko zgodnih svojstava. Kontura definirana 8-konektivnošću se rotira tako da se originalnom kodu doda *n mod 8*. Nadalje, neke druge osobine regije, poput površine, broja uglova, opsega i slično mogu se lako izračunati direktno iz koda.

Ograničenje ove reprezentacije je konačan broj *chain* kodova kojim su definirani nagibi konture [12].

θ-S krivulja

θ-S krivulja je kontinuirana verzija *chain* koda opisanog u poglavlju 0. θ je kut između tangente točke konture i referentne osi (najčešće osi x). S je udaljenost od početka krivulje.

Zatvorena kontura je opisana otvorenom θ-S funkcijom. Ravne horizontalne linije u θ-S prostoru odgovaraju ravnim segmentima konture. Kružni segmenti na konturi odgovaraju ravnim kosim segmentima *θ-S* krivulje.

Diskretni polinomi (Splineovi)

θ-S krivulja opisana u predhodnom poglavlju poglavlju može opisati bilo koji oblik konture, ali nije koncizna niti pogodna za prepoznavanje. S druge strane, parametrička reprezentacija kontura je koncizna, no ograničena je oblicima koje može opisati.

Kompromis predstavljaju reprezentacije diskretnim polinomima. Segmenti kontinuiranih kontura aproksimirani su diskretnim polinomima.

Podjela konture na segmente može se izvršiti na temelju *θ-S* krivulje. Peekovi u $d\theta$-dS prostoru ukazuju na diskontinuitete konture. Te točke definiraju granice diskretnih polinoma.

Općenito, broj segmenata je obrnuto proporcionalan redu polinoma kojim opisujemo pojedini segment. Za reprezentiranje kontura koriste se linearni (59), kvadratni (60) i kubični polinomi (61).

$$y = mx + c \tag{59}$$

$$y = ax^2 + bx + c \tag{60}$$

$$y = ax^3 + bx^2 + cx + d \tag{61}$$

Kontura koja se sastoji od skupa piksela – bridova može se matematičkom regresijom opisati polinomnom funkcijom. Izrazima (62) i (63) se dobivaju koeficijenti linearnog polinoma koji aproksimira N točaka koordinata (x_i, y_i).

$$m = \frac{\left[\sum_{i=1}^{N} x_i y_i - \frac{1}{N}\sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i\right]}{\left[\sum_{i=1}^{N} x_i^2 - \frac{1}{N}\sum_{i=1}^{N} x_i^2\right]} \tag{62}$$

$$c = \bar{y}_i - m\bar{x}_i \tag{63}$$

Nedostatak regresijske aproksimacije jest da se kompletni izrazi (62) i (63) moraju nanovo računati za svaku novu točku. Ova metoda se može primijeniti i za polinome višeg reda, ali su odgovarajući izrazi znatno složeniji.

Diskretnim polinomima se dobiva lokalna kontrola i jednostavnija reprezentacija. Izrazima (64) i (65) dani su uvjeti neprekinutosti diskretnih polinoma $p_i(x)$ u rubnim točkama $x_1, x_2, \ldots, x_{k-1}$.

$$p(x) = p_i(x) \quad za \quad x_i \le x \le x_{i+1} \quad gdje \quad i = 0, 1, 2 \ldots k-1 \tag{64}$$

$$p_i^{(j)}(x_i) = p_{i+1}^{(j)}(x) \quad za \quad j = 0, 1, 2 \ldots r-1 \quad i \quad i = 0, 1, 2 \ldots k-1 \tag{65}$$

$p_i^{(j)}(x)$ je j-ta derivacija i-tog polinoma splinea. Ako je $r = 0$, nema geometrijskih ograničenja. Ako je $r = 1$ kontura je kontinuirana, ali ne i derivabilno kontinuirana. Derivabilna kontinuiranost se dobiva za $r \ge 2$.

7 PREPOZNAVANJE

Pod prepoznavanjem se podrazumijeva sposobnost identificiranja objekata i pojava na temelju prethodno stečenog znanja. Znanje se sastoji od niza spoznaja o značajkama objekata i pojava. Prepoznavanje je utemeljeno na identificiranju i analizi značajki nepoznatih objekata i njihovim uspoređivanjem s značajkama referentnih/poznatih objekata. Značajka je atribut objekta kojim se on opisuje i po kojem ga se razlikuje od ostalih objekata. Značajke mogu biti vizualne, poput boje, oblika, veličine, teksture, gibanja, ali sasvim općenito nije nužno. Objekti i pojeve su opisane i ne-vizualnim značajkama poput mirisa, zvuka okusa i slično [32].

Ljudi lakoćom prepoznaju objekte na temelju velike količine općenitog (iskustvenog) znanja. Osim toga, biološki vizijski sustav posjeduje složenu upravljačku strategiju koja uključuje paralelno procesiranje, promjene ponašanja, promjene fokusa pažnje i slično [31].

Suvremeni razvoj vizijskih sustava nastoji različitim tehničkim sredstvima i postupcima ostvariti funkcionalnost usporedivu s biološkim vizijskim sustavom. To je vrlo složeni zadatak koji usprkos intenzivnim istraživanjima u posljednjih 30-tak godina još uvijek nije u potpunosti ostvaren. Glavni uzrok poteškoćama pri umjetnim vizijskim sustavima je visoki stupanj neodređenosti, posebno:

- varijacije povezane s osvijetljenošću objekata,
- promjena pozicije i orijentacije objekta u odnosu na kameru,
- objekti istog tipa (klase) se međusobno razlikuju.

Kod sustava strojnog vida, pod prepoznavanjem se podrazumijeva postupak kojim se na slikama identificiraju objekti na temelju modela za prepoznavanje. Osim prepoznavanja, zadaća sustava je određivanje pozicija i orijentacije prepoznatih objekata. Sustavi strojnog

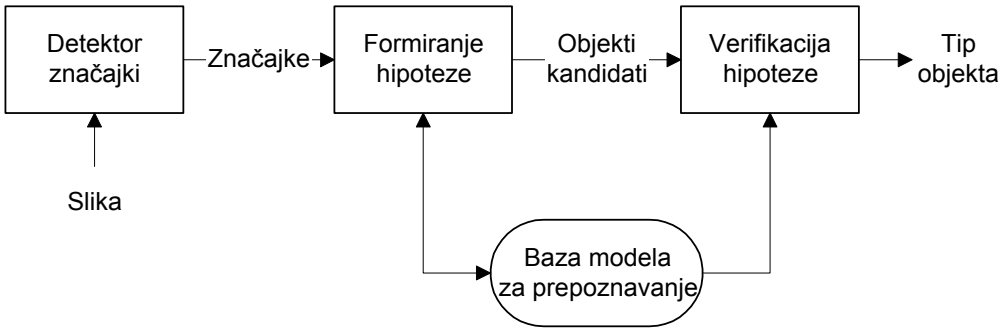
vida konstruiraju interne modele procesirane slike prizora, provjeravaju ih i ažuriraju te ih procesiraju na odgovarajući način. Ako interni model odgovara stvarnosti, slika je ispravno interpretirana, odnosno prepoznati su objekti.

Formalno, postupak prepoznavanja se može definirati i kao postupak imenovanja objekata. Ako slika sadrži jedan ili više "zanimljivih" objekata, prepoznavanjem im se dodjeljuju odgovarajuća imena.

Prepoznavanje je usko povezano s postupkom segmentacije. Bez barem djelomičnog prepoznavanja nije moguće segmentirati sliku, a bez segmentacije prepoznavanje nije moguće [12].

7.1. Komponente sustava za prepoznavanje

Općenita struktura sustava za prepoznavanje je prikazana slikom Sl.-31.



Sl.-31. Struktura sustava za prepoznavanje.

Baza modela za prepoznavanje sadrži sve modele koje koristi sustav. Način na koji su modeli reprezentirani ovisi o strategiji i tehnici prepoznavanja. U svakom slučaju, modeli za prepoznavanje su skupovi značajki objekata.

Pri prepoznavanju objekata i konstruiranju njihovih modela se koristi mnogo različitih značajki. Većina ih se odnosi na regije ili konture slike. Značajke se mogu podijeliti na:

- globalne značajke,
- lokalne značajke,
- relacijske značajke.

Globalne značajke se odnose na čitavu sliku. To su obično karakteristike regija, poput površine, opsega, momenta, konture i slično.

Lokalne značajke reprezentiraju male distinktne dijelove regija. Tipični predstavnici ove skupine su segmenti kontura, zakrivljenost konture i uglovi.

Relacijske značajke opisuju odnose različitih regija, kontura, segmenata kontura ili drugih globalnih ili lokalnih značajki.

Detektor značajki primjenjuje različite operatore na slike u vidu primarnih vizijskih algoritama i pronalazi značajke. Koje značajke se otkrivaju ovisi o strategiji prepoznavanja, objektima koje treba prepoznati i organizaciji baze modela za prepoznavanje.

Formiranje hipoteze je dodjeljivanje vjerojatnosti skupovima značajki da oni pripadaju traženom objektu na temelju znanja o domeni aplikacije. Ovim korakom se smanjuje prostor za pretraživanje na temelju heuristike. Verifikacijom hipoteze se potvrđuju ili odbacuju objekti kandidati uspoređivanjem s njihovim modelima.

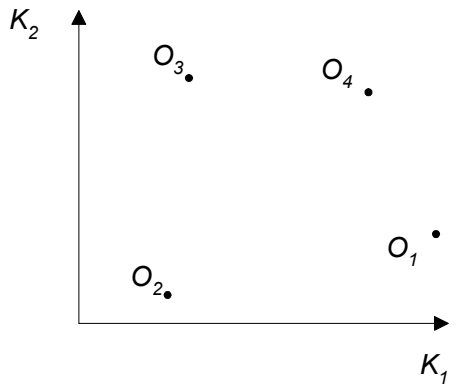
Svi sustavi za prepoznavanje koriste modele za prepoznavanje, eksplicitno ili implicitno. Također, otkrivanje značajki je uvijek prisutno. Prisutnost i važnost komponenta formiranja i verifikacije hipoteze varira u ovisnosti o pristupu procesu prepoznavanja. Neki sustavi samo formiraju hipoteze te kao rješenje prezentiraju hipoteze s najvećom vjerojatnošću. Neki pak sustavi koncentriraju se uglavnom na fazu verifikacije.

7.2. Metode prepoznavanja

7.2.1. Klasifikacija po metodi najbližeg susjeda

Osnovna ideja klasifikacije je prepoznavanje objekata na temelju značajki koje se odnose na prethodno definirane regije. Sve metode klasifikacije pretpostavljaju da je otkriveno N značajki koje se mogu normalizirati i prikazati u zajedničkom metričkom prostoru.

Značajke pojedinih regija, kao i značajke modela se reprezentiraju kao vektor značajki. Slika Sl.-32 prikazuje četiri vektora u dvodimenzionalnom prostoru značajki K_1 i K_2 koje su normalizirane što znači da imaju uniformne jedinice. Uniformne jedinice omogućavaju mjerenje udaljenosti, odnosno sličnosti, između vektora u prostoru značajki.



Sl.-32. Vektori značajki objekata u dvodimenzionalnom prostoru značajki.

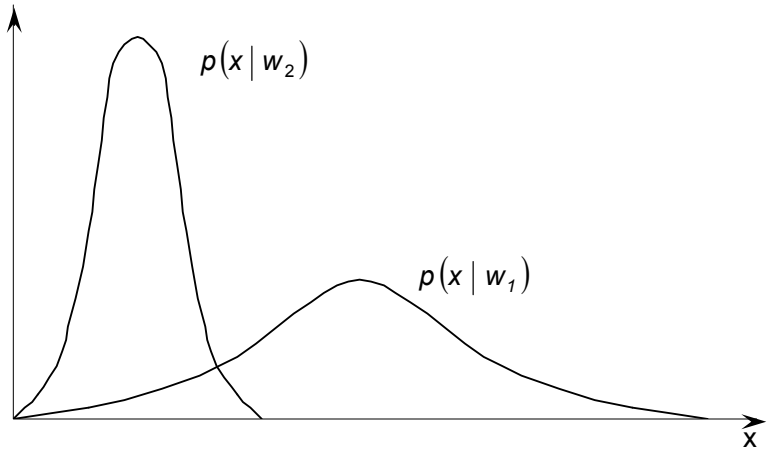
Modeli za prepoznavanje su reprezentirani kao vektori k_{ij}., pri čemu je $i = 1, ..., M$ brojač modela a $j = 1, ..., N$ brojač značajki. Značajke nepoznatog objekta su reprezentirane vektorom u_j. Izrazom (66) je dana udaljenost između nepoznatog objekta i modela za prepoznavanje *i*.

$$d_i = \left[\sum_{j=1}^{N} (u_j - k_{ij})^2 \right]^{1/2} \qquad (66)$$

Nepoznati objekt se dodjeljuje klasi objekta čijem vektoru modela za prepoznavanje je najbliže u vektorskom prostoru značajki.

7.2.2. *Bayesova* klasifikacija

Bayesova klasifikacija se koristi za prepoznavanje objekata kada ne postoje signifikantne razlike između značajki različitih objekata. Na slici Sl.-33 prikazane su statističke razdiobe iznosa iste značajke kod dva različita objekta u jednodimenzionalnom prostoru značajki. Prilikom razmatranja nepoznatog objekta u prostoru značajki, za neke vrijednosti značajki, on se može klasificirati u obje klase. U takvim situacijama koristi se *bayesian* pristup pri donošenju odluka.



Sl.-33. Statističke razdiobe iznosa iste značajke kod dva objekta.

Kod *bayesian* pristupa klasifikaciji koristi se znanje o vjerojatnostima značajke $p(x|w_j)$ za objekt j te vjerojatnostima prisutnosti objekata j u skupu rješenja $P(w_j)$. Na temelju tih informacija mogu se po bayesovom pravilu dobiti induktivne vjerojatnosti $p(w_j|x)$ za objekte koje prepoznajemo. Induktivna vjerojatnost $p(w_j|x)$ opisuje vjerojatnost da nepoznati objekt sa vrijednošću značajke x odgovara klasi j. *Bayesovo* pravilo dano je izrazom (67).

$$P(w_j \mid x) = \frac{p(x \mid w_j) P(w_j)}{\sum_{j=1}^{N} p(x \mid w_j) P(w_j)} \qquad (67)$$

7.2.3. Klasifikacija neuronskim mrežama

Prepoznavanje na temelju klasifikacije značajki može se izvršiti i neuronskom mrežom. Prednost neuronskih mreža je mogućnost particioniranja prostora značajki nelinearnim granicama. Particioniranje se vrši učenjem mreže.

Mreža uči na temelju velikog skupa primjera objekata koje treba prepoznati. Ako je taj skup dovoljno velik i obuhvaća sve klase problema s kojima se mreža pri eksploataciji susreće, granice u prostoru značajki su postavljene ispravno. Prilikom prepoznavanja, neuronska mreža funkcionira kao svaki drugi klasifikator.

7.2.4. Preklapanje (*matching*) značajki

Ako nisu poznate vjerojatnosti značajki ili pojavljivanja objekata, ili nema dovoljno podataka za izradu klasifikatora, prepoznavanje se može izvršiti usporedbom, odnosno preklapanjem nepoznatih objekata i modela za prepoznavanje. Ovom metodom se uspoređuju svi modeli za prepoznavanje s objektima na slici i pri tome se model nastoji prilagoditi podacima sa slike.

Nepoznati objekt se dodjeljuje onoj klasi objekata čiji model mu je najsličniji, odnosno onoj s kojom se najbolje preklapa. Sličnost nepoznatog objekta s modelom za prepoznavanje i S_i je definirana izrazom (68).

$$S_i = \sum_{j=1}^{N} w_j \left| u_j - m_{ij} \right| \qquad (68)$$

U izrazu (68) w_j je težinska vrijednost kojom se kvantificira važnost značajke j. Modeli za prepoznavanje su reprezentirani kao vektori m_{ij}., pri čemu je $i = 1, ..., M$ brojač modela a $j = 1, ..., N$ brojač značajki. Značajke nepoznatog objekta su reprezentirane vektorom u_j.

7.2.5. Preklapanje grafova

Modeli za prepoznavanje se mogu reprezentirati orijentiranim ili neorijentiranim grafom u kojem skup čvorova $V = \{v_1, v_2, v_3, ...\}$ reprezentira značajke modela/objekta a skup lukova $E = \{e_1, e_2, e_3, ...\}$ reprezentira njihove međusobne odnose. Orijentirani lukovi su odnosi poput *veće od* ili *manje od*, dok neorijentirani lukovi predstavljaju odnose poput *paralelno* i *povezano*.

Dva osnovna grafa $G_A = (V_A, E_A)$ i $G_B = (V_B, E_B)$ su izomorfna ako postoji funkcija f definirana izrazom (69).

$$f : V_A \to V_B : \forall (v_i, v_j) \in E_A \; \exists \left(f(v_i), f(v_j)\right) \in E_B \qquad (69)$$

Drugim riječima, postoji odnos između čvorova G_A i G_B kojim su očuvani odnosi lukova. No, da bi se utvrdilo da su dva objekta/modela ekvivalentna, provjera izomorfnosti grafova nije dovoljna. Potrebno je provjeriti da li međusobno odgovaraju oznake čvorova i lukova.

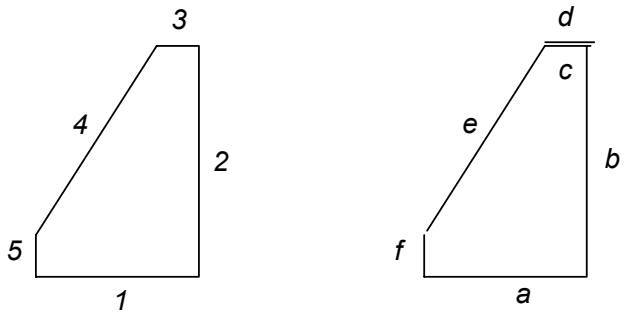
Ovako formulirani kriterij izomorfnosti grafova se ne može primjenjivati pri analizi slike. Uzrok tome je činjenica da se ne mogu očekivati savršeni rezultati segmentacije, što znači da su otkrivene sve potrebne značajke objekata s tim da skup rješenja ne sadrži značajke koje su nastale od sjena i bljeskova. Osim toga, uobičajeno je da je potrebno prepoznati nekoliko različitih objekata na slici.

Stoga je potrebno riješiti problem izomorfizma podgrafova. Pod tim se podrazumijeva problem izomorfizma grafa (V_A, E_A) i podgrafa (V_B, E_B) ili podgrafa (V_A, E_A) i podgrafa (V_B, E_B). Ovaj problem je složeniji od izomorfizma grafova jer nije unaprijed poznato koji čvorovi nedostaju odnosno koji čvorovi se međusobno poklapaju.

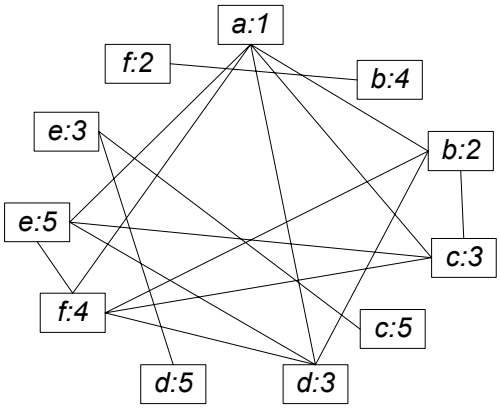
Algoritam *maksimalne klike* [30] primijenjen na asocijacijski graf rješava ovaj problem. Asocijacijski graf čine čvorovi koji predstavljaju kompatibilne i usporedive značajke te lukovi između čvorova s ekvivalentnim relacijama.

Na slici Sl.-34 prikazan je model za prepoznavanje (lijevo) i konture kao značajke za prepoznavanje (desno) nekog objekta kojeg treba prepoznati. Asocijacijski graf načinjen na temelju tog primjera prikazan je slikom Sl.-35. Čvorovi predstavljaju parove detektiranih kontura i kontura modela jednakih (unutar neke tolerancije) duljina. Lukovi su uspostavljeni između parova čvorova koji zadovoljavaju ove uvijete:

- Segmenti kontura modela i detektirani segmenti kontura imaju istu konektivnost. Pod konektivnošću se podrazumijeva broj kontura koje započinju na krajevima analizirane. Naravno, i ovdje postoji tolerantno područje u kojem se pretražuju krajevi kontura.
- Kutna promjena između odgovarajućih segmenata kontura mora biti jednaka.



Sl.-34. Model za prepoznavanje (lijevo) i konture kao značajke za prepoznavanje (desno).



Sl.-35. Asocijacijski graf.

Sljedeći rekurzivni algoritam *maksimalne klike* pronalazi kompatibilne skupove značajki - klike. Najveće klike odgovaraju objektima. Pri tome je N skup čvorova u asocijacijskom grafu a Φ prazni skup.

Klike (Φ, N); (Inicijalni poziv rekurzivnoj proceduri)

Klike (X, Y) :=

Ako nema čvora u *Y* povezanog sa svim elementima u *X* onda

X je klika,

Inače

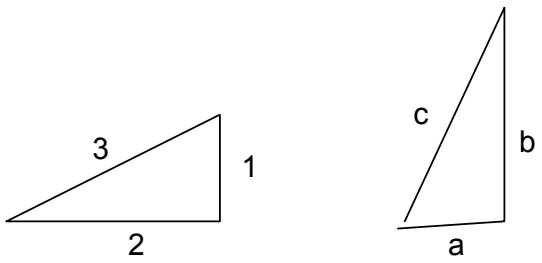
Odaberi čvor *Y* koji je povezan sa svim elementima u *X*

Klike(X U {y}, Y) U Klike(X, Y-{y})

7.2.6. Pretraživanje interpretacijskog drveta

Interpretacijsko drvo se sastoji od čvorova koji predstavljaju sparene značajke detektirane na ulaznoj slici s značajkama modela za prepoznavanje. Lukovi između čvorova reprezentiraju konzistentnost interpretacije. Praćenjem konzistentnih lukova od zadnje razine drveta prema prvoj dobiva se konzistentna interpretacija odnosno prepoznaje se objekt.

Osnovni oblik problema interpretacijskog drveta biti će prikazan na jednostavnom primjeru. Na slici Sl.-36 prikazane su značajke detektirane na slici i model trokuta na temelju kojeg se trebaju interpretirati.



Sl.-36. Jednostavan problem prepoznavanja. Model je prikazan s lijeve strane a značajke slike s desne.

U osnovnom obliku interpretacijskog drveta, prema primjeru, u prvoj razini se nalaze tri čvora. Ako je za prvi čvor (slučajno) odabran par *(a, 1)* preostala dva čvora su parovi značajki *(a, 2)* i *(a, 3)*. Proširivanjem drveta preko ovih čvorova dobiva se novih devet čvorova pri čemu se svaki od čvorova na prvoj razini proširuje na *(b, 1)*, *(b, 2)* i *(b, 3)*. Dakle, potpuno neograničeno interpretacijsko drvo za prepoznavanje prema modelu s n značajki na slici s detektiranih n značajki na prva razina interpretacijskog drveta ima n čvorova, druga n^2, a posljednja n-ta ima n^n čvorova.

Zbog ove kombinatorne eksplozije nužno se vrši pretraživanje ograničenog interpretacijskog drveta. U ograničeno interpretacijsko drvo se ne uključuju čvorovi na temelju kojih proizlazi interpretacija koja nije konzistentna s geometrijskim ograničenjima ili drugim znanjima o sadržaju prizora.

Na primjer, ako je poznato da prizor sadrži samo jedan objekt, onda se čvor *(a, 1)* na prvoj razini može proširiti samo s čvorovima *(b, 2)* i *(b, 3)*.

Geometrijska ograničenja mogu biti unarna i binarna. Unarna definiraju odnos između jedne značajke modela i značajke slike. Primjer je ograničenje kojim se zahtijeva da čvor mogu činiti samo značajke jednake dužine. Binarna ograničenja definiraju odnos između dva para značajki modela i slike. Primjer je ograničenje kojim se zahtijeva da kut između značajki jednog čvora bude jednak kutu između značajki drugog čvora. Novi čvorovi se mogu zadavati tako da su zadovoljena binarna ograničenja samo s prethodnim čvorom ili sa svim prethodnom čvorovima.

Redoslijed proširivanja čvorova može biti *depth - first* ili *bredth - first*. U svakom slučaju potrebna je mjera kvalitete čvora kao i čitave interpretacijske grane. Kvaliteta čvora se može odrediti, na primjer, na temelju razlike duljina segmenata kontura ili njihove zakrivljenosti. Kvaliteta interpretacijske grane se najbolje očituje u duljini grane jer je ona pokazatelj količine konzistentnih značajki.

U idealnom slučaju proširivanje grafa završava kada se spare sve značajke modela sa svim značajkama na slici. Ipak, to je teško ostvarivo budući da na slici uglavnom ne sadrži sve značajke, a korumpirana je i značajkama od sjena i bljeskova. Stoga je uobičajeno da proces završava na temelju heurističkog znanja.

8 SUSTAV ZA AUTOMATSKU MONTAŽU PRIMJENOM STROJNOG VIDA

8.1. Koncept fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj okolini

U radu se pojam sređenosti definira u kontekstu znanja o stanju sustava i procesa. Ovako definiran pojam sređenosti proizlazi iz perspektive subjekata sustava. Može se reći da povećavanjem količine informacija/znanja kojima raspolažu elementi sustava relativno povećavamo stupanj sređenosti, iz čega proizlazi sljedeća hipoteza.

Hipoteza 1:

Sređenost sustava ne proizlazi iz fizičkog ograničavanja objekata i subjekata sustava, već naprotiv, sređenost se interpretira isključivo kao informacijska kategorija.

Kod automatskog sklapanja sređenost sustava povezana je sa sređenošću radne okoline. Pod nesređenom se okolinom pri automatskom sklapanju podrazumijeva svaka situacija/konfiguracija objekata i subjekata koja nije potpuno poznata.

Potrebno je ustvrditi da postoje različite razine sređenosti. Potpuno nesređena radna okolina podrazumijeva da se o uvjetima rada unaprijed ne zna (gotovo) ništa. Druga krajnost su uvjeti gdje je i unaprijed i tijekom montažnog procesa konfiguracija ugradbenih elemenata u potpunosti poznata. Između ove dvije krajnosti je golem međuprostor različitih situacija.

Lema 1:

> **Kako bi automatski montažni sustav s nesređenom radnom okolinom postao sređen, sustav mora raspolagati informacijama/znanjem o stanju okoline te o samom procesu.**

Informacijsko sređivanje sustava omogućuje implementaciju inteligentnih upravljačkih programa, sposobnih djelovati u nepredvidivim okolnostima. Nadalje, informacijskim sređivanjem sustava na temelju percepcije radne okoline smanjuje se količina potrebnog unaprijed strukturiranog, konceptualiziranog i operacionaliziranog znanja.

Sljedećim informacijama/znanjima povećava se spoznaja o montažnom sustavu i procesu:

- plan montaže,
  - redoslijed sklapanja,
  - putanje sklapanja,
  - tehnike spajanja,
  - ...
- konfiguracija (stanje) montažne opreme,
- pozicije, orijentacije i ostale značajke ugradbenih dijelova,
- pozicije i orijentacije narastajućeg proizvoda,
- ostali događaji i stanja radne okoline (na primjer, kolizije sa ostalim objektima u proizvodnom procesu)

Određenost pozicija i orijentacija ugradbenih elemenata su jedan od važnijih čimbenika koji određuju ukupni stupanj sređenost okoline montažnog procesa. Usko je povezana s operacijama rukovanja. Prema definiciji [2], pod rukovanjem se razumijevaju operacije kojima se ugradbeni elementi dovode u položaj za spajanje. Ukoliko su pozicije i orijentacije dijelova neodređene, rukovanje nužno uključuje tranziciju stanja u određene pozicije i orijentacije.

Prijelaz iz proizvoljnog u određeno/poznato prostorno stanje, odnosno upravljanje pozicijama i orijentacijama predmeta rada jedan od najtežih zadataka prilikom razvoja automatskih montažnih sustava. Te se poteškoće manifestiraju prilikom projektiranja operacija rukovanja pa otuda i sljedeća teza: ukoliko je moguće automatizirati rukovanje, moguće je automatizirati i cijeli montažni proces [7].

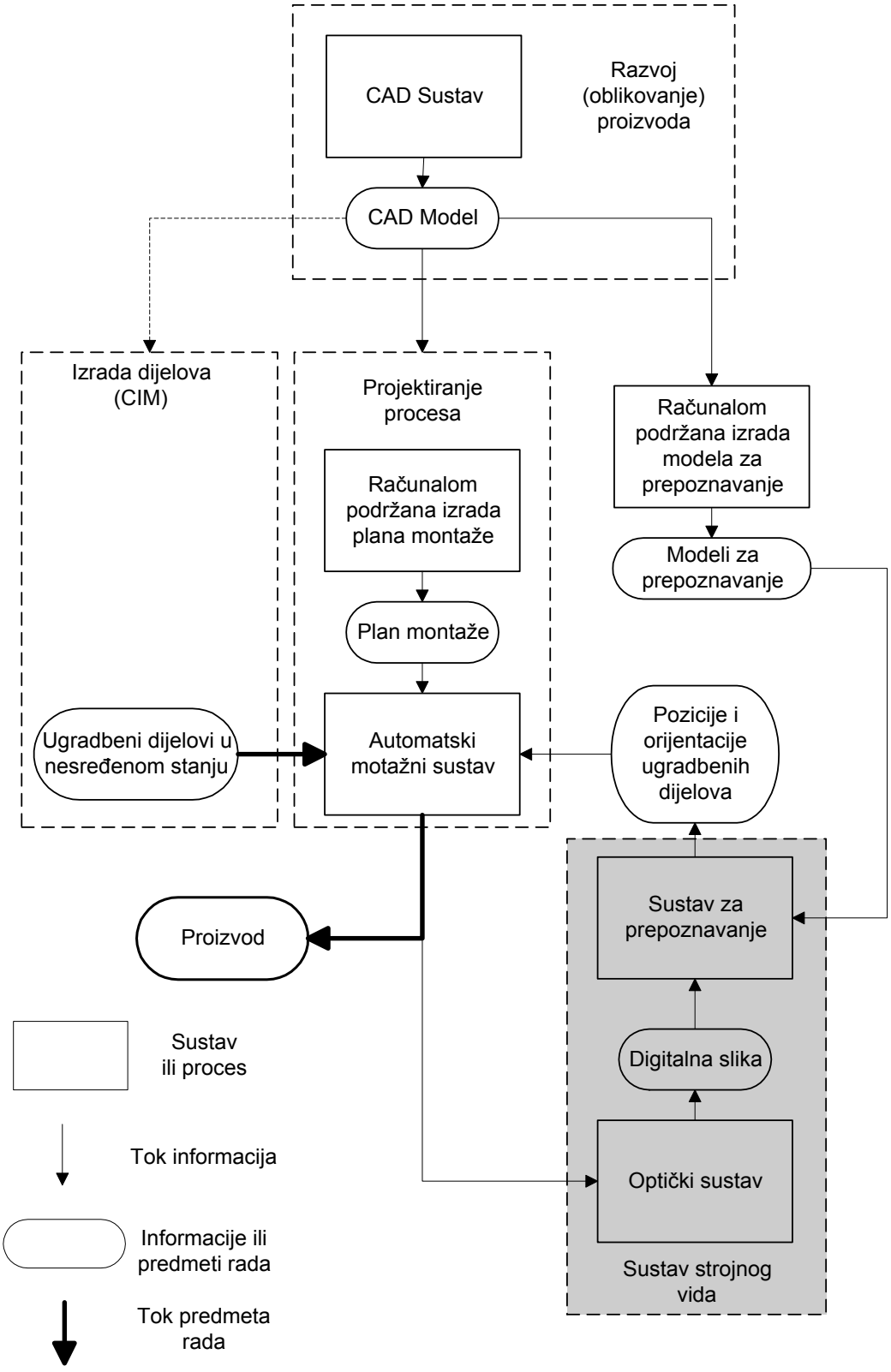
Hipoteza 2:

> **Strojnim vidom se mogu prepoznati odgovarajući ugradbeni elementi te odrediti njihove pozicije i orijentacije, kao i pozicije i orijentacije sklopa u svim fazama montažnog procesa. Na taj način pozicije i orijentacije dijelova postaju određene, te taj aspekt montažnog sustava postaje sređen.**

Lema 2:

> **U takvim okolnostima programska podrška na upravljačkim uređajima montažnih robota ili neke duge montažne opreme mora osigurati da se ugradbeni dijelovi, sada određenih pozicija i orijentacija, po zadanim putanjama i prema ostalim zadanim tehnološkim parametrima ugrade u sklop.**

Na osnovi postavljenih hipoteza i lema proizlazi pretpostavka da se upotrebom suvremenih inženjerskih alata te metoda za prepoznavanje oblika i određivanje pozicija i orijentacija može realizirati fleksibilni integrirani sustav za automatsko sklapanje nesređenih ugradbenih elemenata. Sustav strojnog vida je osnovni izvor znanja o prostornim odnosima u sustavu. Koncept sustava, definiranog na temelju postavljenih hipoteza, je prikazan slikom Sl.-37.



Sl.-37. Koncept fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj okolini.

Integracija djelatnosti podrazumijeva integraciju informacija. Ukoliko su informacije dovoljno sadržajne i točne, i ukoliko ih možemo učinkovito prenositi, moguće je na temelju njihove integracije umrežiti i paralelizirati različite inženjerske djelatnosti, smanjujući uska grla u toku informacija i skraćujući razvojni ciklus. To znači da je moguće osigurati razmatranje različitih tehnoloških i proizvodnih aspekata već u ranoj fazi oblikovanja proizvoda. Na taj način se postiže usmjereni konvergirajući kolaborativni pristup razvoju bez značajnih povratnih tokova. Pretpostavka za oblikovanje takvih sveobuhvatnih informacija i njihovu integraciju je primjena računala i računarskih metoda. Putem računalnih mreža inženjeri mogu projektne podatke trenutno razmjenjivati i istovremeno obrađivati neovisno o tome gdje se nalaze. Integracija tako dobiva potpuni smisao, uključujući informacijsku i prostornu integraciju inženjerskih resursa [33].

8.1.1. O oblikovanju proizvoda

Ideja predstavljenog koncepta za automatsko sklapanje počiva na minimaliziranju entropije informacija, te stoga tok informacija polazi od razvoja proizvoda, odnosno od mjesta gdje se prvi put definiraju i strukturiraju inženjerske informacije o proizvodu.

Suvremeni razvoj proizvoda podrazumijeva primjenu računarskih tehnologija. Proizvodi se oblikuju pomoću CAD[19] sustava. Pod CAD sustavom se podrazumijeva inženjerski alat, sastavljen od računala i odgovarajuće programske podrške, koji služi za oblikovanje proizvoda.

Oblikovanje proizvoda CAD sustavom rezultira CAD modelom. CAD model je analitička/matematička reprezentacija oblikovnih (Sl.-38), funkcionalnih (Sl.-39), vizualnih (estetskih) i fizikalnih značajki proizvoda (Sl.-40).

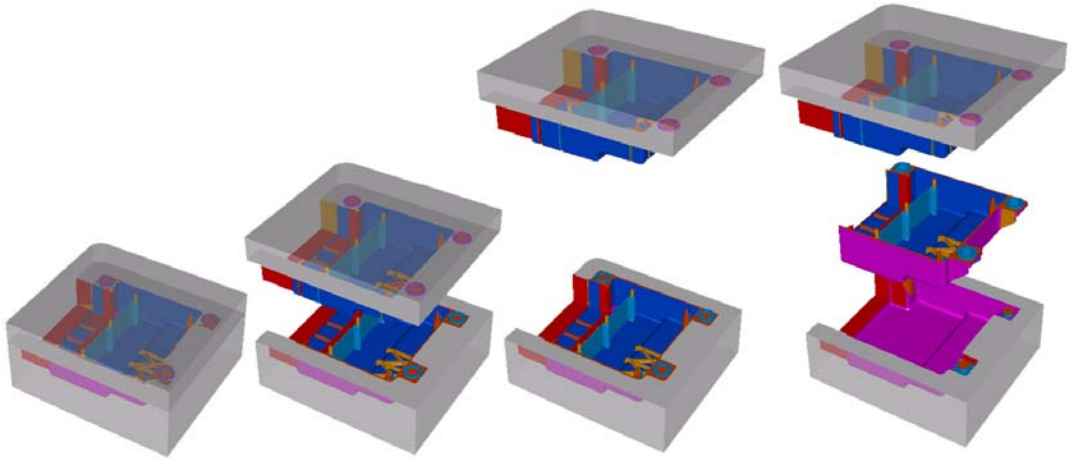
Inženjerske informacije ugrađene u CAD model neminovno se višestruko koriste u sljedećim fazama proizvodnog procesa, bez obzira da li se izvode iz CAD modela ili se nanovo strukturiraju. Procesi čiji parametri izravno ovise o obliku proizvoda, mogu se, dakle, u određenoj mjeri definirati pomoću CAD modela. Ukoliko je ta mogućnost propuštena, neminovno se povećava informacijska entropija.

U prikazanom konceptu informacije sadržane u CAD modelu se koriste kod:

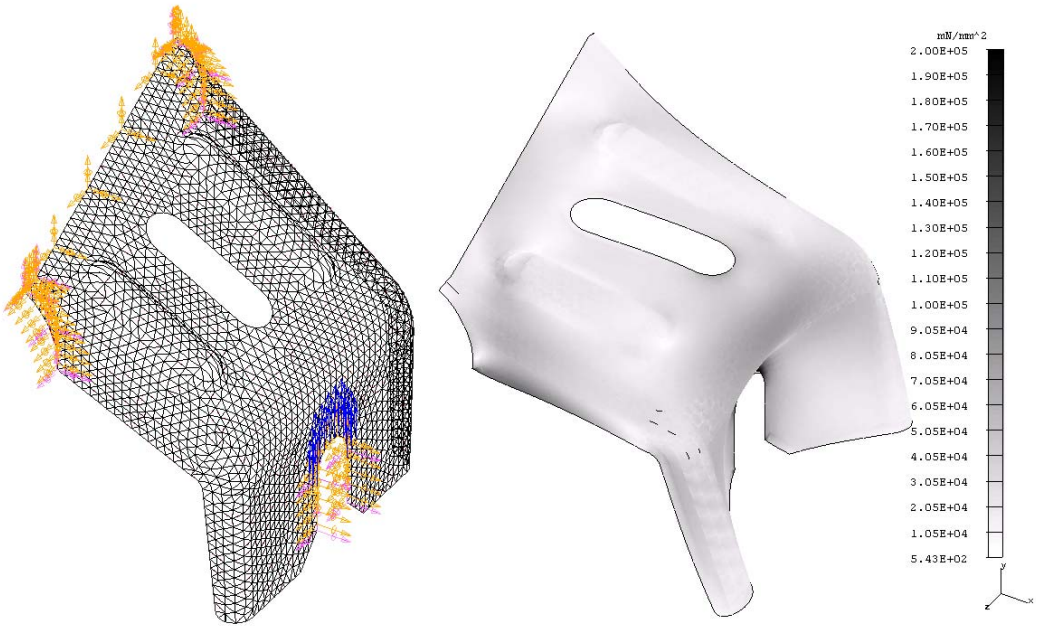
- izrade dijelova,
- projektiranja montažnog procesa,
- izvođenja montažnog procesa (strojni vid)

[19] CAD – eng. Computer Aided Design – Računalom Podržano Oblikovanje

Sl.-38. CAD model sadrži informacije o obliku proizvoda.



Sl.-39. Definiranje funkcionalnih značajki proizvoda kinematičkim modelom.



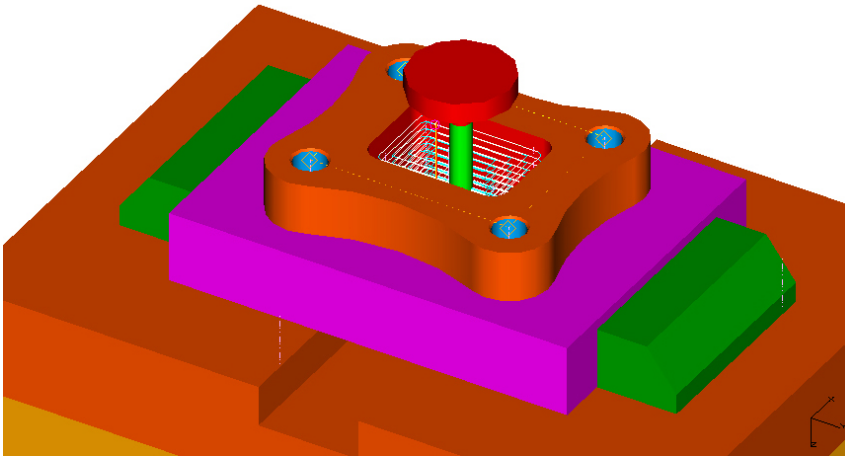
Sl.-40. CAD model omogućuje simulaciju fizikalnih svojstava

8.1.2. O izradi dijelova

Pomoću izradbenih tehnologija proizvod se materijalizira. Tehnološki parametri izradbenih tehnologija ovise o obliku proizvoda. S obzirom da CAD model opisuje sve relevantne informacije o budućem proizvodu, logično je da predstavlja osnovu za definiranje tehnoloških parametara.

Takav je pristup već vrlo dobro implementiran u obradi alatnim strojevima. CAD model se koristi za definiranje putanja numerički upravljanih alatnih strojeva prilikom obrada složenih ploha (Sl.-41). Te su tehnologije poznate kao CAM[20] ili CAD-CAM.

[20] CAM – eng. Computer Aided Manufacturing – računalom podržana izrada



Sl.-41. Putanja alata definirana na osnovi CAD modela proizvoda.

8.1.3. O projektiranju montažnoga procesa

U sklopu projektiranja procesa, u kontekstu automatskog sklapanje u nesređenoj okolini, posebno je zanimljiva izrada plana montaže. Naime, plan automatske montaže predstavlja skup informacija potrebnih za obavljanje montaže nekoga proizvoda, prikazujući postupak izvođenja i sredstva za montažu.

Ovisno o namjeni i uvjetima izrade, plan automatske montaže poprima različite oblike i obuhvaća različite sadržaje kao što su:

- redoslijed sklapanja,
- operacije sklapanja,
- putanje sklapanja,
- pozicije i orijentacije dijelova u proizvodu,
- početni položaji dijelova,
- hvataljke,
- uređaji,
- sile sklapanja,
- ...,

a za oblikovanje, prikaz i izvođenje montaže [1].

Računalom podržana izrada plana montaže je predmet aktualnih istraživanja. Neka od istraživanja prikazana su u [3][34][35][36]. Alati za izradu planova montaže su najčešće djelomično automatizirane aplikacije koje generiraju plan montaže na temelju CAD modela proizvoda.

U postavljenom konceptu fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj okolini pretpostavlja se da je struktura montažnog sustava poznata i konstantna, a

to znači da se pretpostavlja da je raspored elemenata montažnog sustava nepromjenjiv. Varijabla sustava ostaju pozicije i orijentacije ugradbenih elementa te postupak sklapanja.

Potrebno je iskoristiti informacije koje su u planu montaže već strukturirane i operacionalizirane kako bi se automatski definirali parametri programske podrške upravljačkih jedinica. Kako je uzeto da je struktura sustava konstantna, mijenjaju se samo parametri programske podrške karakteristične za pojedini proizvod.

Konstantna struktura automatskog montažnog sustava za montažu različitih sklopova je moguća samo uz uvjet da se montažni uređaji odlikuju prilagodljivošću s obzirom na položaj predmeta rada. Industrijski robot tipičan je primjer za takvu vrstu strojeva.

Sljedeće informacije iz plana montaže direktno definiraju parametre programske podrške za upravljanje robota:

- Redoslijed sklapanja definira redoslijed izvođenja programskih ciklusa upravljačke jedinice robota.
- Pozicije i orijentacije dijelova u sklopu definiraju položaj robotske prihvatnice na kraju putanje umetanja.
- Putanje umetanja dijelova definiraju putanje robotske prihvatnice prilikom umetanja.
- Sile potrebne za ostvarivanje spojeva definiraju sile koje mora ostvariti robot prilikom sklapanja.
- ...

8.1.4. O strojnom vidu u konceptu fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj okolini

Sustav strojnog vida u fleksibilnom integriranom sustavu za automatsko sklapanje u nesređenoj okolini identificira odgovarajuće ugradbene elemente, prema planu montaže, i određuje njihove pozicije i orijentacije u sustavu.

Kako se sustavom strojnog vida određuje prostorno stanje objekata montažnog procesa, pod pojmom prepoznavanje se u ovom radu podrazumijeva postupak identificiranja odgovarajućih ugradbenih elemenata ili sklopova te određivanje njihovih položaja, odnosno pozicija i orijentacija.

Optički sustav (digitalna kamera) proizvodi digitalnu sliku prizora u kojoj se nalaze ugradbeni dijelovi. Programska podrška sustava strojnog vida na temelju digitalne slike identificira "potrebne" ugradbene dijelove te dostavlja informacije o njihovim položajima kao tehnološke parametre upravljačkim uređajima montažnog sustava.

Pretpostavlja se da su gotovo svi parametri programske podrške strojnog vida u donekle konstantnim uvjetima su nepromjenjivi. Pod konstantnim uvjetima se ovdje razumijevaju položaji kamera, optičke postavke kamere, osvjetljenje radnog prostora i slično. Primarni algoritmi vizije u takvim uvjetima ne zahtijevaju konfiguriranje za svaki novi proizvod.

Promjenjivi parametri su naravno oni koji opisuju oblik proizvoda. U sustavu strojnog vida to je model za prepoznavanje.

Model za prepoznavanje ugradbenih dijelova je reprezentacija geometrije proizvoda prikladna za programsku podršku strojnog vida.

Kako CAD model također opisuje geometriju proizvoda, nameće se ideja da se iskoristi za kreiranje modela za prepoznavanje.

Izrada modela za prepoznavanje iz CAD modela proizvoda povećava stupanj integracije programske podrške sustava. Na taj se način smanjuje entropija informacija strukturiranih u fazi razvoja koje opisuju proizvod i proces. Integrirana programska podrška omogućuje povećanje razine autonomnosti sustava za montažu, a time i cjelokupnog proizvodnog sustava.

8.1.5. O Automatskom montažnom sustavu u konceptu fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj okolini

Zadaća automatskog montažnog sustava, u predstavljenom konceptu, jest da sklopi proizvod, unaprijed definirane strukture, iz ugradbenih dijelova dobavljenih i nasumično postavljenih u radnu okolinu robota.

To se ostvaruje tako da programska podrška upravljačke jedinice robota integrira informacije strukturirane u planu montaže te informacije iz sustava strojnog vida.

U planu montaže su definirane informacije o procesu sklapanja. Izravno se koriste podaci o:

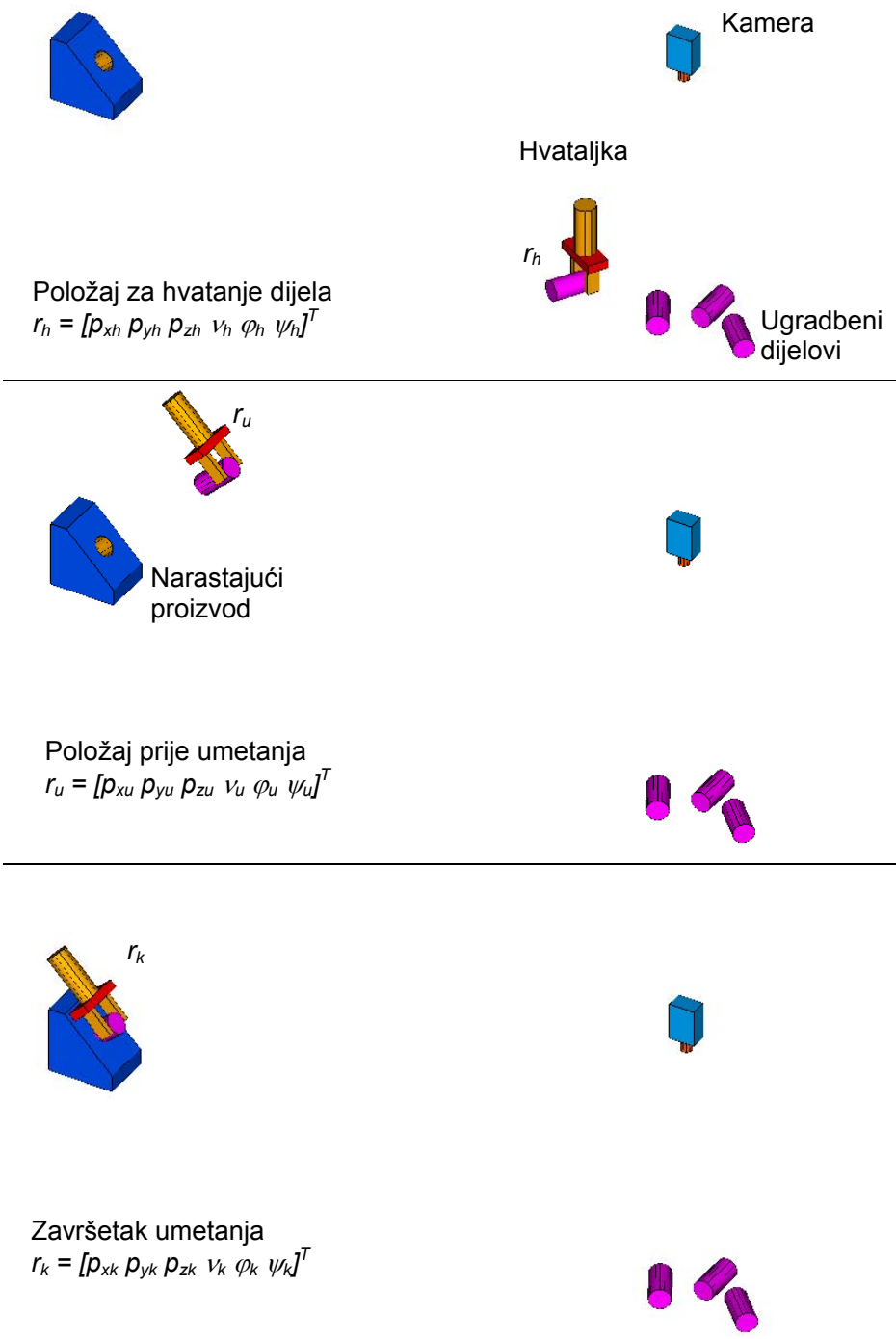
- redoslijedu sklapanja,
- putanjama sklapanja

Sustav strojnog vida inducira informacije o položajima ugradbenih dijelova. Redoslijed sklapanja definira koje ugradbene elemente treba ugraditi, i naravno, redoslijed izvođenja odgovarajućih robotskih operacija.

Robotske operacije obuhvaćaju sve potrebne radnje/zahvate za postavljanje ugradbenog dijela u odgovarajući položaj u sklopu. Potrebno je, dakle, definirati putanju hvataljke robotske ruke uključujući:

- hvatanje odgovarajućeg dijela u radnoj okolini robota,
- prenošenje ugradbenog dijela do sklopa, odnosno do pozicije iz koje će započeti umetanje u sklop,
- spajanje (umetanje, naslagivanje …).

Robotska operacija, odnosno putanja robotske hvataljke se dobiva na temelju minimalno tri zadana položaja i orijentacije robotske hvataljke. Tri položaja robotske hvataljke, iz kojih se konstruira robotska operacija, su ilustrirana slikom Sl.-42.



Sl.-42. Tri položaja robotske hvataljke koje određuju putanju robota prilikom sklapanja jednog ugradbenog dijela.

Položaj robotske hvataljke je jednoznačno određen s vektorom položaja i orijentacija robota *r*. Vektor položaja i orijentacija robota *r* definiran je s tri kartezijske koordinate p_x, p_y *i* p_z, te tri eulerova kuta ν, φ, *i* ψ [37].

Vektori r_h r_u r_k su tehnološki parametri programske podrške upravljačke jedinice robota. Na temelju njih se konstruira kritični dio robotske operacije. Naime, cjelokupna robotska operacija sadrži još i dio putanje kojim hvataljka dolazi položaj za hvatanje te dio putanje kojom se robot odmiče od proizvoda nakon umetanja dijela.

U primjeru sa slike Sl.-42 putanja umetanja je pravocrtna. Samo u tom slučaju su dovoljni vektori r_u i r_k za definiranje putanje umetanja, pri čemu su njihovi eulerovi kutovi jednaki. U slučaju krivocrtne putanje, cjelokupna definicija putanje iz plana montaže mora biti uzeta u obzir prilikom definiranju robotske operacije.

Vektor r_h odgovara početnom položaju ugradbenog dijela. Budući da robotska okolina nije sređena, njegove komponente se definiraju sustavom strojnog vida i prosljeđuje upravljačkoj jedinici robota.

Komponente vektora r_u i r_k su definirane planom montaže. Vektor r_u odgovara položaju ugradbenog dijela u sklopu. Vektor r_k odgovara početnoj točci putanje umetanja. Ukoliko položaj proizvoda nije unaprijed definiran i konstantan, sustav strojnog vida mora biti uključen i određivanje ovih položaja. Zahtijevani položaji robota r_u i r_k mogu se dobiti zbrajanjem vektora zahtijevanih pozicija iz plana montaže te stvarnih pozicija narastajućeg proizvoda.

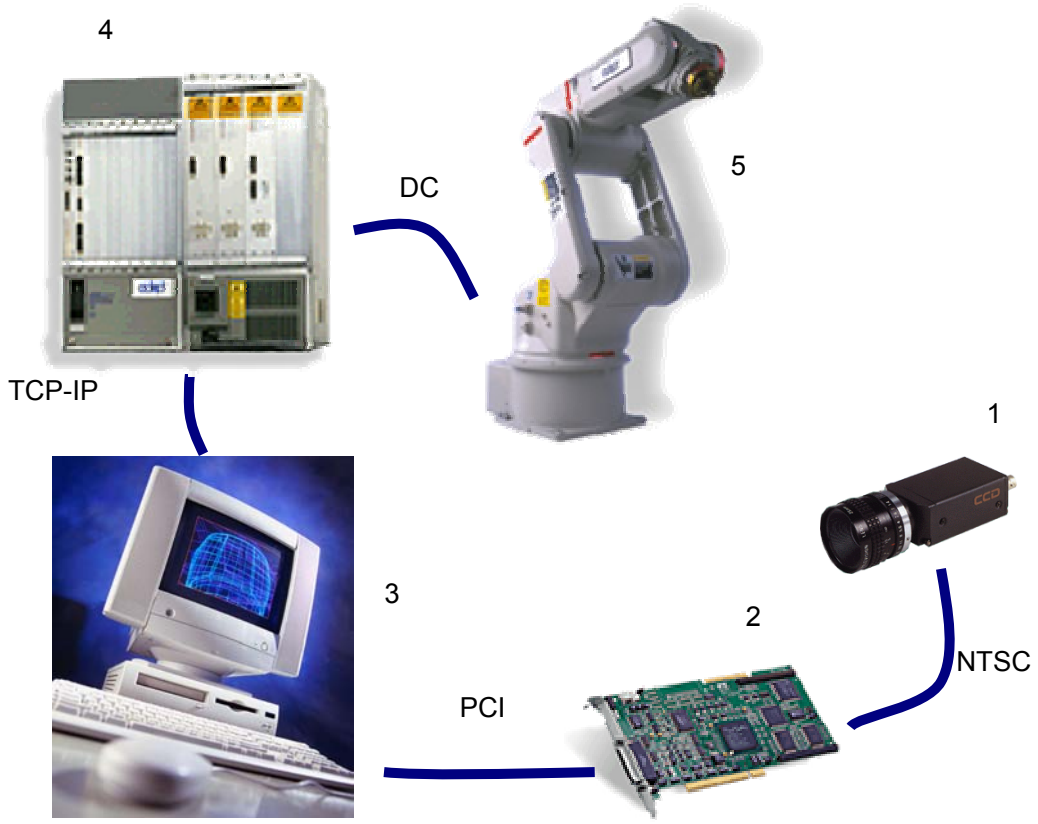
8.2. Robotski sustav sa strojnim vidom za sklapanje u nesređenoj okolini

U svrhu istraživanja realiziran je robotski sustav sa strojnim vidom za sklapanje u nesređenoj okolini. On predstavlja jednu komponentu fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj okolini opisanog u poglavlju 8.1.

Sustav se sastoji od povezanih hardverskih komponenti te odgovarajuće programske podrške. Slika Sl.-43 prikazuje hardverske komponente sustava i način na koji su međusobno povezane.

Prema slici Sl.-43 sustav se sastoji od sljedećih komponenti:

1. CCD digitalna kamera
2. Digitalizator
3. PC kompatibilno računalo
4. Upravljačka jedinica robota
5. Robot



Sl.-43. Hardverske komponente te njihove međusobne veze.

8.2.1. CCD Digitalna kamera

Sustav je opremljen Panasonic GP-MF602 CCD digitalnom kamerom. Ova kamera je pogodna za aplikacije strojnog vida gdje se zahtjeva velika brzina te za aplikacije u fleksibilnim proizvodnim sustavima [39].

Kamera sadrži CCD chip s linijskim prijenosom razlučivosti 768 (H) x 494 (V) piksela. Podržano je skaniranje integracijom polja i okvira. Kamera producira monokromatski video prema NTSC standardu [46].

8.2.2. Digitalizator

Digitalizator je elektronički uređaj koji preuzima video signal s kamere i iz njega konstruira matrične digitalne slike. Programska podrška strojnog vida zahtjeva matričnu digitalnu sliku kao što je opisano u poglavlju 4.2.

Sustav je opremljen *Matrox Meteor-II/Multi-Channel* digitalizatorom. Ovaj uređaj je sposoban istovremeno obraditi video signale sa šest monokromatskih kamera ili dvije RGB kolor kamere [40].

Uređaj je spojen s PC kompatibilnim računalom preko PCI sabirnice [41]. Konstruirane digitalne slike se pohranjuju u radnu memoriju računala i na taj su način dostupne programskoj podršci strojnog vida.

8.2.3. Upravljačke jedinica robota

Upravljačke jedinica robota je uređaj kojim se upravlja izvršnim članom u sustavu – robotom. Gibanjem robota se upravlja napajanjem njegovih servomotora. Upravljačka jedinica i servomotori robota su povezani energetskim kablovima. Upravljačka jedinica je povezana i s PC računalom na kojem se izvodi programska podrška strojnog vida te koji služi i kao njezino korisničko sučelje.

Sustav je opremljen upravljačkom jedinicom robota *Adept 540 Robot Controller* [42]. Uređaj je baziran na *040 AdeptWindows* Arhitekturi. Sastoji se od *MV-10* upravljačke šasije te *PA-4* pojačala snage.

Upravljačka šasija je opremljena *AWC-II* sistemskom upravljačkom pločom s *68040* procesorom. Na upravljačkoj ploči se nalaze sučelja prema ostalim uređajima i to:

- Ethernet konektor
- IEEE 1394 konektor
- SCSI konektor

Ethernet konektorom se uređaj povezuje s PC računalom Ethernet odnosno TCP/IP protokolom [43]. IEEE 1394, odnosno *Firewire* konektorom [44] međusobno su povezani *MV-10 i PA-4* pojačalo snage koje upravljačke signale pretvara u radni napon servomotora robota. Preko standardnog SCSI sučelja povezan je *upravljački privjesak* robota.

Upravljačka jedinica robota je u osnovi specijalno računalo te kao svako računalo posjeduje operacijski sustav. Operacijski sustav razvijen za *Adept* upravljačke jedinice je *V*$^+$ [45].

8.2.4. Robot

Sustav je opremljen robotom *AdeptSix 300*. *AdeptSix 300* je kompaktni, visokoučinkoviti robot sa 6 stupnjeva slobode gibanja. Maksimalni doseg robota je 677 mm a nominalna nosivost je 3 kg. Točnost ponavljanja iznosi ± 0.025 mm.

Robot je pogonjen *Adept 540 Robot Controller* upravljačkom jedinicom. Robotski programi, pisani su u *V*$^+$ programskom jeziku, izvode se na upravljačkoj jedinici, sa *V*$^+$ operacijskim sustavom.

8.3. Programska podrška robotskog sustava sa strojnim vidom za sklapanje u nesređenoj okolini

Za izvođenje automatskog montažnog procesa u nesređenoj okolini razvijena je odgovarajuća programska podrška. Ona se sastoji od programske podrške strojnog vida i upravljačkog programa robota.

Programska podrška strojnog vida je razvijena koristeći *HEXSIGHT* knjižnicu programskih objekata koji uključuju osnovne vizijske algoritme (na primjer algoritmi za filtriranje, segmentaciju, prepoznavanje, kalibriranje i slično).

Njezina je zadaća da na temelju slike preuzete s kamere interpretira sadržaj prizora. Pod tim se podrazumijeva da mora identificirati potrebne ugradbene dijelove te odrediti njihove pozicije i orijentacije.

U skladu s predloženim konceptom fleksibilnog integriranog sustava za automatsko sklapanje u nesređenoj okolini (8.1), mora sadržavati značajke koje će omogućiti integraciju s ostalim elementima sustava. To se u prvom redu odnosi na vezu s CAD sustavom gdje se definiraju oblikovne značajke proizvoda i vezu sa sustavom za projektiranje montažnog sustava i procesa.

Budući da je razvijeni sustav prvenstveno istraživački i edukacijski, proces prepoznavanja treba biti u što većoj mjeri vizualiziran. To se prvenstveno odnosi na prikaz ulaznih podataka, rješenja na srednjoj razini procesa i konačnih rezultata.

Rezultate programske podrške strojnog vida preuzima upravljački program robota. Osim njih, on preuzima i informacije iz sustava za projektiranje montažnog sustava i procesa. Za razliku od programske podrške strojnog vida kojoj je potrebna tek informacija o redoslijedu sklapanja, ovaj program mora preuzeti potpune definicije putanja sklapanja budući da je njegova zadaća da na temelju ta dva skupa podataka definira gibanje robota koje omogućuje izvođenje montaže proizvoda.

Programska podrška robotskog sustava sa strojnim vidom za sklapanje u nesređenoj okolini izvodi se na PC računalu i na upravljačkoj jedinici robota. U skladu s koncepcijom sustava PC i upravljačka jedinica robota su povezani putem ethernet mrežnog protokola. Razmjena informacija je ostvarena NFS[21] protokolom, odnosno integracijom zapisničkih sustava.

8.3.1. Programska podrška strojnog vida

Osnovna struktura programske podrške strojnog vida je prikazana slikom Sl.-44. Pod osnovnom strukturom se podrazumijeva ustrojstvo samo onih elemenata sustava koji direktno sudjeluju prepoznavanju ugradbenih dijelova i određivanju njihovih pozicija i

[21] eng. Network file system – Mrežni zapisnički sustav.

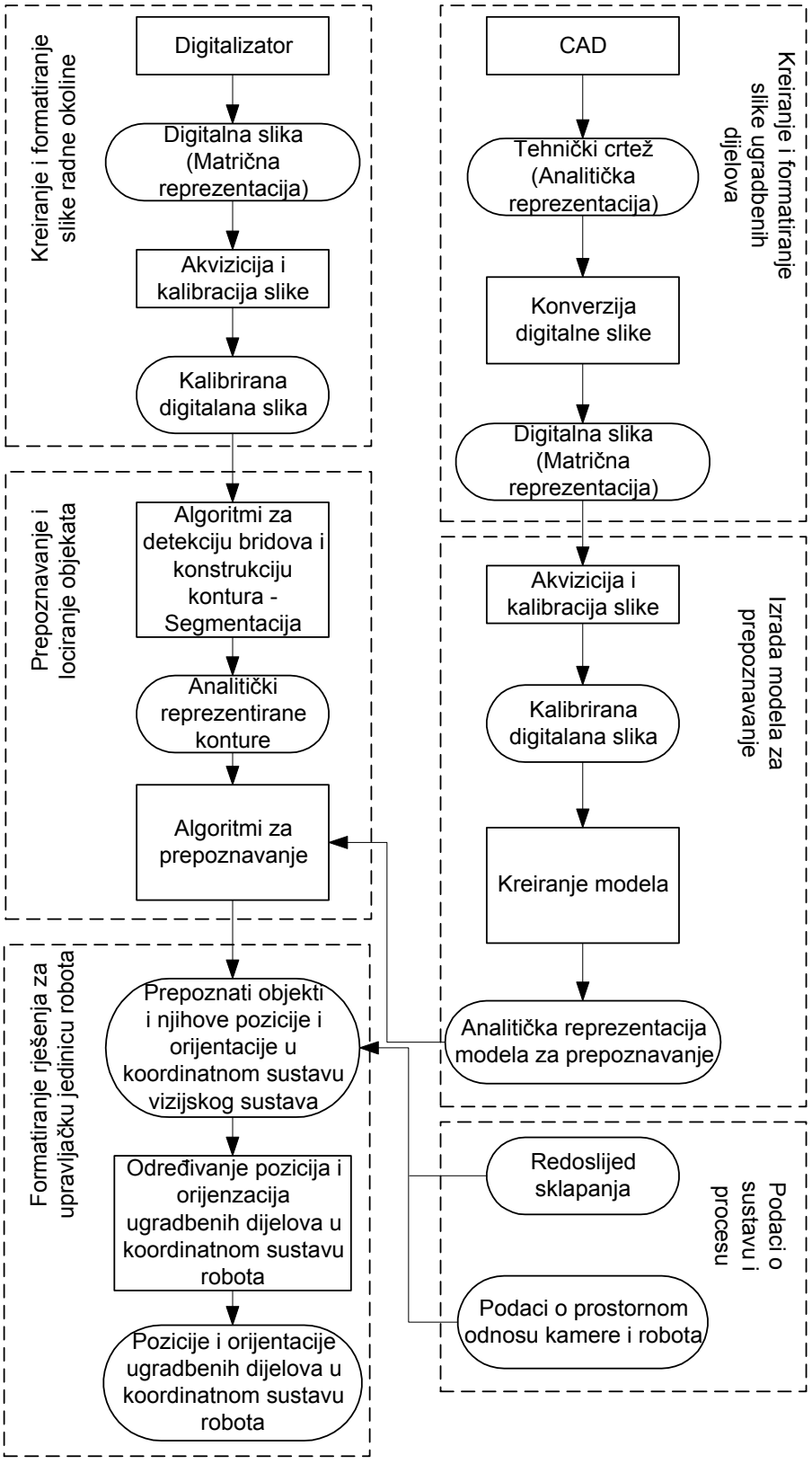
orijentacija. Dijelovi sustava koji osiguravaju vizualizaciju rješenja, podešavanja parametara komponenata sustava i upravljanje podacima nisu uključeni u shemu.

Programska podrška je razvijena u programskom jeziku *C++*. Korišteno je razvojno okruženje *Microsoft Visual Studio 6.0*. Programski kod je dan u prilozima

Ulazi u programsku podršku su:

- monokromatska digitalna slika u matričnoj reprezentaciji kreirana na digitalizatoru iz analognog video signala.
- vektorska digitalna slika kreirana CAD sustavom iz 3D modela proizvoda.
- redoslijed sklapanja.
- podaci o prostornom odnosu kamere i robota.

Rezultat, odnosno izlaz programske podrške su pozicije i orijentacije ugradbenih dijelova. Koordinate pozicija i orijentacije su definirane u koordinatnom sustavu robota pa se mogu direktno iskoristiti za definiranje položaja robotske prihvatnice pri hvatanju dijelova. Nadalje, rezultati su sortirani prema redoslijedu sklapanja.



Sl.-44. Struktura programske podrške strojnog vida.

8.3.1.1. Kreiranje i formatiranje slike radne okoline

Digitalna slika se kreira na digitalizatoru. Digitalizator je elektronički uređaj koji iz video signala s kamere konstruira statičku, matričnu, digitalnu sliku te je pohranjuje u memoriju računala.

"Sirova" digitalna slika s digitalizatora nije pogodna za procese vezane uz prepoznavanje, budući da je degenerirana greškama kamere (8.3.2). Osim toga potrebno je uspostaviti odnos između stvarnih dimenzija predmeta i njihovih dimenzija u pikselima.

Kalibracija je postupak kojim se kompenziraju greške kamere i definira mjerilo između veličine u pikselima i stvarne veličine predmeta. Kalibracija je podrobnije opisana u poglavlju 8.3.2 gdje se opisuje alat za akviziciju i kalibraciju slike.

Proces kalibracije rezultira s kalibriranom digitalnom slikom koja se u sljedećem koraku obrađuje algoritmom za detekciju bridova te algoritmom za kreiranje kontura.

8.3.1.2. Prepoznavanje i lociranje objekata

Proces segmentacije je baziran na bridovima. Detekcija bridova je izvršena *Sobelovim* algoritmom.

Na temelju bridova su kreirane konture specifičnim "*border tracking*" algoritmom. Konture su opisane analitički kao spline krivulje. Time završava proces segmentacije [47].

Algoritam za prepoznavanje generira hipoteze[22] i pronalazi objekte uspoređivanjem kreiranih kontura i analitičkog modela za prepoznavanje. Prepoznavanje se vrši algoritmom klasifikacijskog tipa. Alat za prepoznavanje i lociranje objekata koji predstavlja implementaciju algoritma podrobnije je opisan u poglavlju 8.3.2.

Algoritam za prepoznavanje producira rješenje koje sadrži popis prepoznatih dijelova te odgovarajuće pozicije i orijentacije. Pozicije i orijentacije dijelova su određene koordinatama te kutom rotacije u vizijskom koordinatnom sustavu. Popis sadrži sve prepoznate dijelove. Ukoliko je prepoznato više istih dijelova, svi su sadržani u skupu rješenja.

8.3.1.3. Kreiranje i formatiranje slike ugradbenih dijelova

Iz CAD modela se kreiraju 2D tehnički crteži. Tehnički crteži (između ostalog) sadrže poglede. Pogledi su (u pravilu) ortogonalne projekcije 3D CAD modela, odnosno predmeta koje se opisuje, na ravnine. Položaj ravnina uobičajeno odgovara mogućim orijentacijama.

Za izradu modela za prepoznavanje potreban je pogled koji odgovara pogledu kamere na ugradbeni dio. Pogled kamere je pogled koji se dobiva ortogonalnom projekcijom na ravninu kamere (CCD chipa).

[22] Hipoteze po kojima se prvi put skupovi kontura razmatraju kao mogući objekti (7.1).

Pogledi, odnosno tehnički crteži su analitički formalizirani. Moraju biti prevedeni u format kompatibilan zapisu (formatu) slika koji proizvodi kamera, odnosno digitalizator. Iz tog se razloga izvodi konverzija u sliku matrične reprezentacije.

8.3.1.4. Model za prepoznavanje

Model za prepoznavanje je objekt koji opisuje geometriju predmeta kojeg je potrebno prepoznati i locirati. Kreira se na temelju kontura kreiranih iz matrične digitalne slike ugradbenog dijela. Konture se generiraju istim algoritmima kao i konture slike s digitalizatora. Alat za kreiranje modela koji predstavlja implementaciju algoritma podrobnije je opisan u poglavlju 8.3.2.

Kao i slike s digitalizatora, matričnu sliku ugradbenog dijela je potrebno prethodno kalibrirati.

8.3.1.5. Formatiranje skupa rješenja za upravljačku jedinicu robota

Skup rješenja se reducira na onoliko objekata koliko je definirano planom sklapanja. Nadalje, skup rješenja se sortira tako da odgovara redoslijedu sklapanja.

Potrebno je izvršiti i homogene transformacije koordinata položaja u koordinatni sustav robota. Homogene transformacije se definiraju na temelju prostornog odnosa vizijskog koordinatnog sustava i koordinatnog sustava robota. Postupak povezivanja dvaju koordinatnih sustava podrobnije je opisanu u poglavlju 8.3.4.

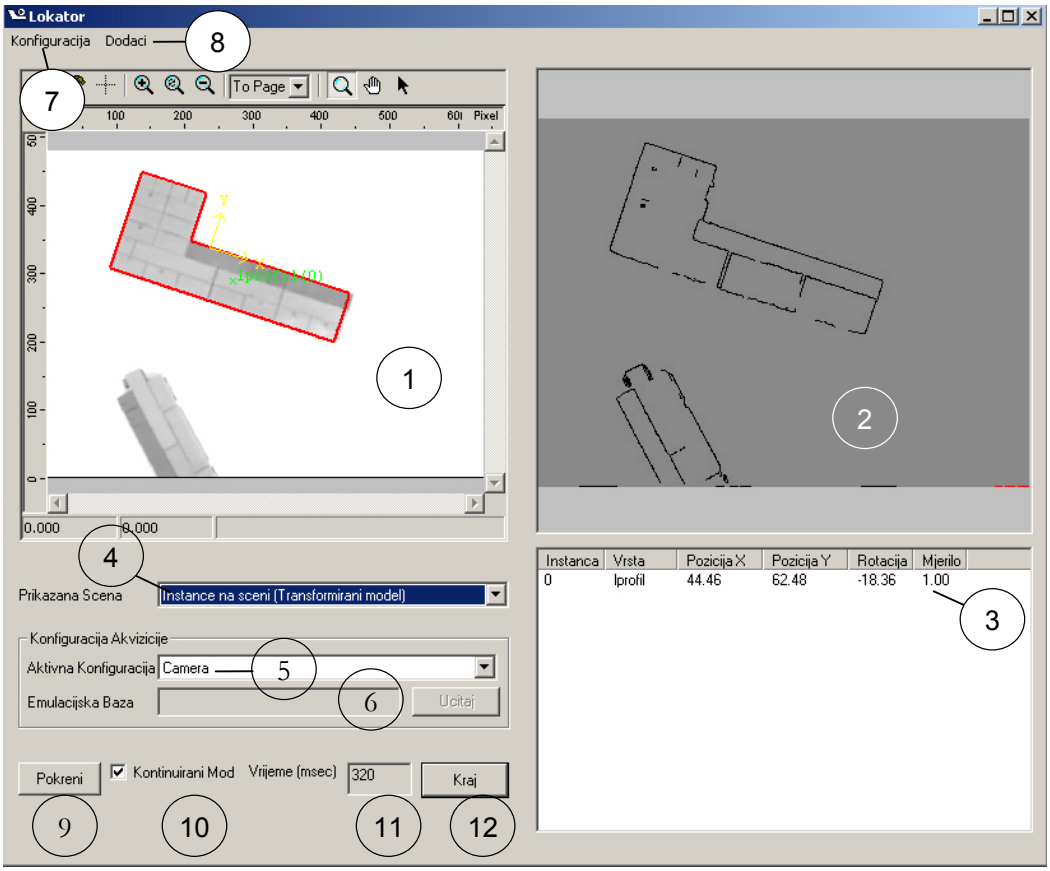
Nakon reduciranja i sortiranja skupa rješenja te homogenih transformacija koordinata, rješenje se daje na raspolaganje upravljačkoj jedinici robota. Rješenje je popis ugradbenih elemenata sortiran po redoslijedu sklapanja. Za svaki dio navedena je pozicija točke hvatanja u koordinatnom sustavu robota. Kut rotacije je definiran u odnosu na os *X* koordinatnog sustava robota.

8.3.1.6. Korisničko sučelje programske podrška strojnog vida

Putem korisničkog sučelja definiraju se parametri programske podrške. Osim toga, sučelje služi za vizualizaciju procesa prepoznavanja. Korisničko sučelje programske podrške strojnog vida je prikazano slikom Sl.-45. Prema oznakama na slici sučelje sadrži:

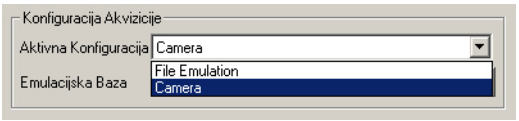
1. prozor ulazne slike,
2. prozor detektiranih bridova,
3. tablicu pronađenih instanci,
4. padajući izbornik za prikaz prizora,
5. padajući izbornik za ulaznu sliku,
6. polje za izbor emulacijske baze,
7. padajući izbornik za konfiguraciju,
8. padajući izbornik dodataka,
9. gumb za pokretanje inspekcijskog ciklusa,
10. uključivanje kontinuiranog moda,

11. trajanje inspekcijskog ciklusa,
12. gumb za isključivanje programske podrške.



Sl.-45. Sučelje programske podrške strojnog vida.

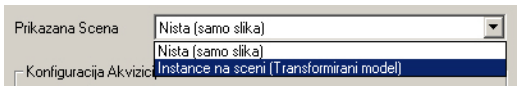
U prozoru ulazne slike prikazana je digitalizirana slika radnog prostora. Ulazna slika se može uzimati s kamere ili iz emulacijske baze slika. Izvor ulazne slike se definira padajućim izbornikom #5 – na slici Sl.-45 koji je detaljnije ilustriran slikom Sl.-46.



Sl.-46. Padajući izbornik za definiranje izvora ulazne slike.

Osim ulazne slike u ovom prozoru se mogu prikazati rezultati. Rezultati su prikazani kao transformirani modeli. Transformirani modeli su modeli prepoznavanja ukomponirani u ulaznu sliku tako da preklapaju tražene objekte. Osim bridova objekata, transformirani modeli prikazuju lokalni koordinatni sustav koji definira točku hvatanja te rotaciju/orijentaciju objekta.

Opcija prikaza rezultata se uključuje padajućim izbornikom za prikaz prizora #4 - Sl.-45 koji je detaljnije ilustriran slikom Sl.-47.



Sl.-47. Padajući izbornik za definiranje opcije prikaza rezultata.

U prozoru detektiranih bridova prikazane su konture kreirane na temelju ulazne slike. Treba naglasiti da su prikazane samo konture kreirane preciznijim procesom. Naime, detekcija bridova se vrši na dvije razine točnosti. Prikazani su rezultati finije razine točnosti, jer se revizijom kreiranih kontura olakšava podešavanje parametara programske podrške te postavljanje adekvatnog osvjetljenja i postavki optičkog sustava.

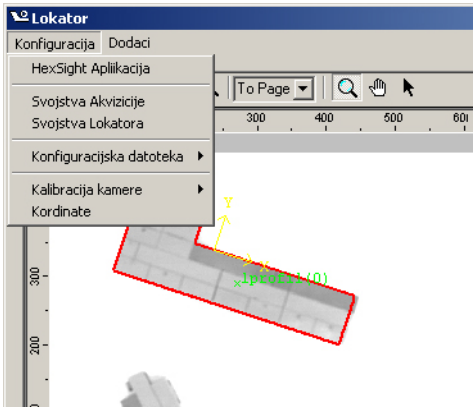
Tablica pronađenih instanci, označena oznakom #3 na slici Sl.-45, služi za ispis rezultata analize slike. Svaki pronađeni objekt opisan je u jednom redu tablice kao što je to ilustrirano slikom Sl.-48 kojom je prikazan tipičan oblik rezultata analize slike. Objekti su opisani sljedećim značajkama:

- rednim brojem u listi pronađenih objekata,
- imenom,
- koordinatama točke hvatanja,
- kutom rotacije lokalnog koordinatnog sustava transformiranog modela,
- mjerilom uvećanja ili umanjenja objekta u odnosu na model za prepoznavanje.

Instanca	Vrsta	Pozicija X	Pozicija Y	Rotacija	Mjerilo
4	bazni	552.49	49.79	-120.61	1.00
0	valjak	519.61	-37.42	-177.86	1.00
5	zatik	550.06	-20.72	-51.49	1.00
3	plocica	575.28	-95.98	50.32	1.00
1	vijak	514.00	-97.47	-90.76	1.00
8	vijak	487.74	-16.87	-177.47	1.00

Sl.-48. Rezultati analize slike u tablici pronađenih instanci.

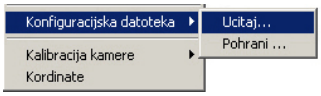
Slikom Sl.-49 prikazana je prva razina padajućeg izbornika za konfiguriranje parametara programske podrške označenog oznakom #7 na slici Sl.-45.



Sl.-49. Padajući izbornik za konfiguriranje parametara programske podrške strojnog vida

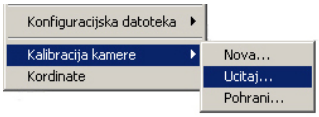
Prve tri stavke na izborniku, *HexSight aplikacija, svojstva akvizicije i svojstva lokatora* otvaraju sučelja korištenih komercijalnih vizijskih alata. One će biti opisane u sljedećim poglavljima.

Opcijom *konfiguracijska datoteka*, ilustriranom slikom Sl.-50, pohranjuju se sve postavke vizijske aplikacije u zapisnik. Pohranjene postavke mogu se naravno i učitati iz zapisnika.



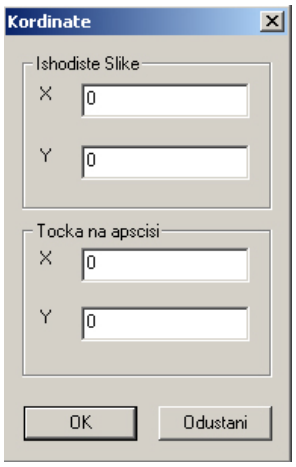
Sl.-50. Izbornik za pohranjivanje i učitavanje konfiguracijskog zapisnika.

Stavkom *kalibracija kamere*, ilustriranom slikom Sl.-51, se definiraju, pohranjuju i učitavaju kalibracijski parametri kamere.



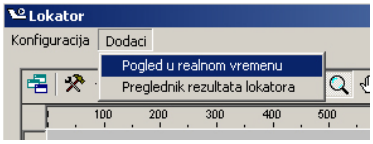
Sl.-51. Izbornik za definiranje, pohranjivanje i učitavanje kalibracijskih parametara kamere.

Stavkom *koordinate* otvara se prozor, ilustriran slikom Sl.-52, za unos parametara kojima je definiran prostorni odnos vizijskog koordinatnog sustava i koordinatnog sustava robota.



Sl.-52. Sučelje za unos očitanih koordinata.

Padajući izbornik dodataka, označen oznakom #8 na slici Sl.-45, detaljno ilustriran slikom Sl.-53 omogućuje pokretanje preglednika slike s kamere u realnom vremenu te dodatnog preglednika rezultata prepoznavanja.



Sl.-53. Padajući izbornik dodataka.

Uključivanjem kontinuiranog moda, označenog oznakom #10 na slici Sl.-45, pokreće se beskonačna petlja ciklusa traženja objekata. Ovisno o snazi računala te broju objekata na ulaznoj slici, programska podrška je sposobna producirati oko 10 skupova rezultata u sekundi.

Kako bi podaci bili dostupni programskoj podršci na upravljačkoj jedinici robota, upisuju se u tekstualni zapisnik. Svaka značajka rezultata predstavlja jedan zapis[23]. Zapisi su međusobno odvojeni "new line" znakovima. U prilogu P.3 prikazan je sadržaj zapisnika s rezultatima.

8.3.2. Primijenjeni vizijski alati

Programska podrška strojnog vida je utemeljena na komercijalnim vizijskom programskom paketu/sustavu *Adept HEXSIGHT*. *Adept HEXSIGHT* skup programskih alata u formi eng. *Active X control.*

[23] Eng. Record.

ActiveX control je programski objekt kojeg mogu koristiti različiti programi na jednom ili više umreženih računala. Mogu se koristiti za obavljanje bilo koje zajedničke zadaće za više programa u novijim *Windows* ili *Macintosh* okruženjima [38].

Pod vizijskim alatima se, dakle, razumijevaju različiti *Adept HEXSIGHT* ActiveX programski objekti.

Realizirana programska podrška strojnog vida se koristi sljedećim vizijskim alatima u obliku *HEXSIGHT* ActiveX programski objekata:

- *HS Application Control* – programski objekt za upravljanje vizijskim alatima i razmjenu podataka među njima,
- *HS Acquisition Device* – programski objekt za akviziciju i kalibriranje digitalne slike,
- *HS Locator* – programski objekt za obradu slike primarnim vizijskim algoritmima te prepoznavanje i lociranje objekata,
- *HS Model Editor* – programski objekt za izradu modela za prepoznavanje,
- *HS Display* – programski objekt za vizualizaciju.

Upravljanje vizijskim alatima i razmjena podataka

Programski objekt, odnosno alat za upravljanje vizijskim alatima i razmjenu podataka, je osnovni objekt u programskoj podršci strojnog vida utemeljenoj na *HEXSIGHT* ActiveX programskim objektima. Njime se upravlja ostalim alatima uključenim u programsku podršku te radnom bazom [47].

Parametar ovog programskog objekta je lista alata koji sudjeluju u programskoj podršci. Važan je redoslijed popisa, jer se operacije, koje su realizirane alatima, izvršavaju po tom redoslijedu. Pojedine su operacije pak definirane parametrima programskih objekata kojima su realizirane.

Ipak treba naglasiti da se u listu alata ne uključuju alati za prikaz rezultata, poput alata za vizualizaciju. Dovoljno ih je samo uključiti kao *ActiveX* programske objekte u programsku podršku strojnog vida.

Programski objekt za upravljanje vizijskim alatima upravlja i radnom bazom. Sadržaj radne baze kreiraju i koriste ostali alati.

Radna baza sadrži dvije vrste podataka. To su:

- slike kao matrice piksela,
- vektorski opisane konture objekata.

Akvizicija i kalibriranje digitalne slike

Programski objekt, ili alat, za akviziciju i kalibriranje digitalne slike dobavlja monokromatsku sliku. Slika se pohranjuje u radnu bazu podataka kako bi je mogli koristiti ostali alati.

Slika se dobavlja s jednog od ovih izvora:

- s digitalizatora,
- iz emulacijske baze slika,
- iz radne baze.

Uobičajeni izvor digitalne slike prilikom eksploatacije programske podrške je digitalizator. Digitalna slika s kamere je pohranjena u memoriji samog uređaja ili u radnoj memoriji računala.

Emulacijskom bazom digitalnih slika se emuliraju slike koje prilikom normalnog rada programske podrške generira digitalizator. Na taj se način olakšava razvoj programske podrške. Prilikom testiranja nije potrebno imati kameru niti stvarnu radnu okolinu.

Radna baza također može biti izvor slike. Slika u tom slučaju potječe od drugog alata ili iz neke druge programske podrške.

Programska podrška sadrži integrirane alate za kalibriranje slike. Kalibriranjem se:

- definira položaj "koordinatnog sustava svijeta",
- uspostavlja odnos između jedinica koordinatnog sustava svijeta (fizikalne jedinice dužine) i koordinatnog sustava slike (pikseli).
- kompenziraju greške kamere.

Koordinatni sustavi

Primijenjeni alati služe se sa slijedećim koordinatnim sustavima:

- koordinatni sustav slike,
- koordinatni sustav svijeta,
- koordinatni sustav objekta.

Koordinatni sustav slike koriste programi za analizu slike. Ishodište mu je u jednom od uglova slike, najčešće u gornjem lijevom. Jedinice ovog koordinatnog sustava su pikseli.

Koordinatni sustav svijeta je referentni koordinatni sustav. Jedinice ovog sustava su fizikalne jedinice dužine. To je referentni koordinatni sustav za prikaz rezultata.

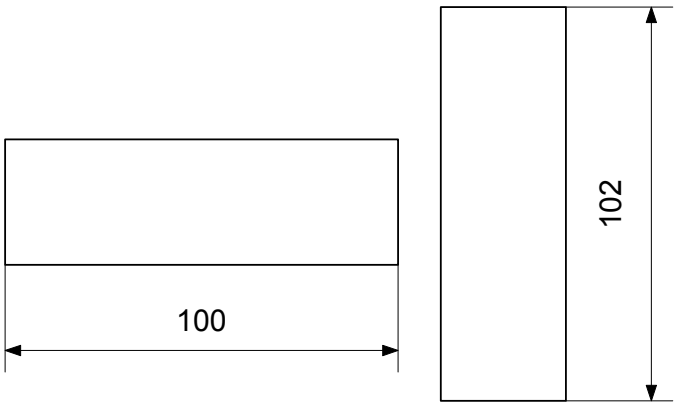
Koordinatni sustav objekta definira poziciju i orijentaciju objekta u koordinatnom sustavu svijeta.

Pogreške kamere

Pogreške se odnose na odstupanja slike od stvarnih geometrijskih odnosa objekata. Kalibracijom se kompenziraju pogreške na slici i tako se doprinosi točnosti programske podrške. Pogreške na slici nastaju kao posljedica:

- odstupanja od kvadratnog oblika piksela,
- perspektivne distorzije,
- distorzije leće kamere.

Pogreška odstupanja od kvadratnog oblika piksela se očituje prilikom rotacije objekta. Ovaj problem je ilustriran slikom Sl.-54 na dva identična objekta u dvije različite orijentacije.



Sl.-54. Dva jednaka objekta i njihove dimenzije u pikselima za dvije različite orijentacije.

Perspektivna distorzija se javlja uvijek kada kamera nije savršeno okomita na radnu površinu. To se na slici očituje kao pojava kuta između paralelnih bridova na objektu.

Distorzije leće kamere uzrokuje radijalnu distorziju slike. Njezin intenzitet je obrnuto proporcionalan fokalnoj dužini leće.

Programski objekt za akviziciju i kalibriranje digitalne slike raspolaže alatima za kalibraciju slike po sljedećim metodama:

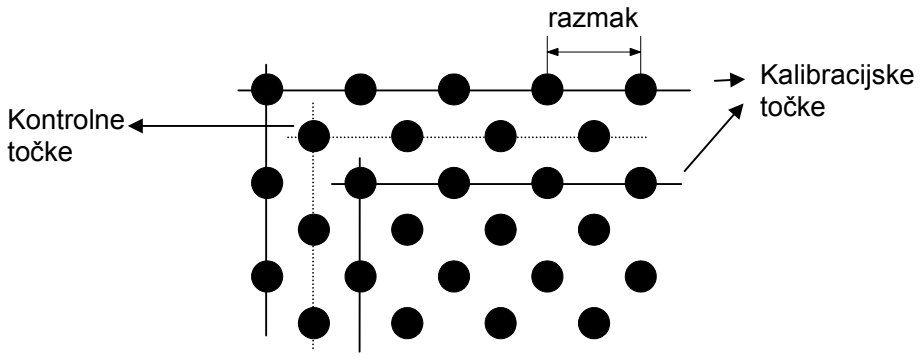
- XY kalibracija,
- perspektivna kalibracija,
- distorzijska LUT kalibracija,
- kalibracija distorzijskim modelom.

U radu razvijena programska podrška strojnog vida koristi XY kalibraciju i kalibraciju distorzijskim modelom.

XY kalibracija se provodi pomoću bilo kojeg objekta poznatih dimenzija. Objekt se snimi kamerom, a zatim se definiraju dimenzije objekta na slici. Ovom metodom kalibracije nije moguće kompenzirati pogreške perspektivne distorzije i distorzije leće. Zbog toga se ova metoda koristi samo u slučajevima kada je os kamere okomita na radnu površinu.

Kalibracijom distorzijskim modelom mogu se kompenzirati svi tipovi pogrešaka. Postupak se vrši pomoću kalibracijske mete prikazane slikom Sl.-55. Meta je sačinjena od ekvidistantnih kalibracijskih i kontrolnih točaka.

Na temelju slike mete, metoda kreira nelinearni model koristeći *best-fit* pristup, uzimajući u obzir sve vidljive točke [47]. Poznati razmak točaka omogućuje kompenzaciju grešaka na slici.



Sl.-55. Kalibracijska meta za kalibraciju distorzijskim modelom.

Kalibracijom se definira i položaj koordinatnog sustava svijeta. Ishodište desnokretnog koordinatnog sustava se postavlja u središte točke najbliže donjem lijevom kutu slike. Osi koordinatnog sustava su koincidentne s pravcima na kojima leže središta kalibracijskih točaka prvog vidljivog reda i prvog vidljivog stupca.

Prepoznavanje i lociranje objekata

Alatom za prepoznavanje i lociranje se identificiraju objekti na slici te se određuju njihove pozicije i orijentaciju.

Ulaz je monokromatska slika iz radne baze. Kako bi se osigurala točnost rezultata, slika mora biti prethodno kalibrirana.

Rezultati programske podrške, odnosno izlazi su:

- alfanumerički opisi pozicija i orijentacija pronađenih objekata,
- grafički prikaz pronađenih bridova i objekata.

Pozicija i orijentacija svakog pronađenog objekta je opisana s:

- imenom objekta,
- X *i* Y koordinatama ishodišta koordinatnog sustava pronađenog objekta u koordinatnom sustavu svijeta,
- kutom između osi $+X$ koordinatnih sustava svijeta i objekta,
- faktorom uvećanja pronađenog objekta u odnosu na model za prepoznavanje.

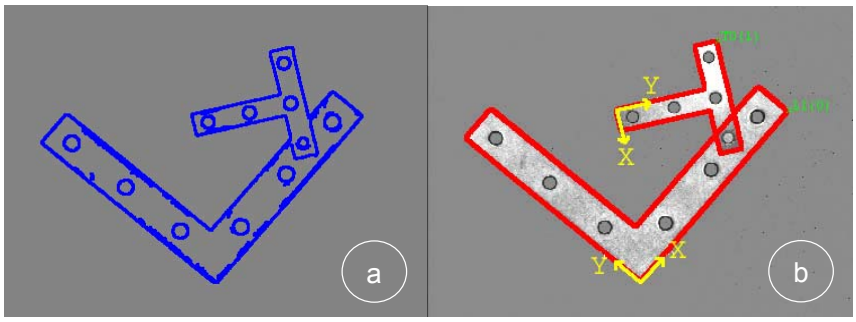
Grafički prikazi rezultata pohranjuju se u radnoj bazi kako bi ih mogli koristiti drugi alati. Grafički rezultati uključuju:

- prizor kontura,
- prizor objekata,
- integriran prizor objekata.

Prizor kontura je vektorska slika koja sadrži konture generirane na temelju ulazne monokromatske slike (Sl.-56).

Prizor objekata je vektorska slika koja sadrži samo konture pronađenih objekata. Konture kreirane postavljanjem konture modela na odgovarajuću poziciju.

Integriran prizor objekata je prizor objekata integriran u ulaznu digitalnu sliku (Sl.-56).



Sl.-56. Prizor kontura (a) i integrirani prizor objekta (b).

Objekti se identificiraju uspoređivanjem s modelom za prepoznavanje. Model za prepoznavanje je reprezentacija objekta bazirana na opisu geometrije njegovih kontura. Kao i monokromatska ulazna slika, modeli za prepoznavanje nalaze se u radnoj bazi.

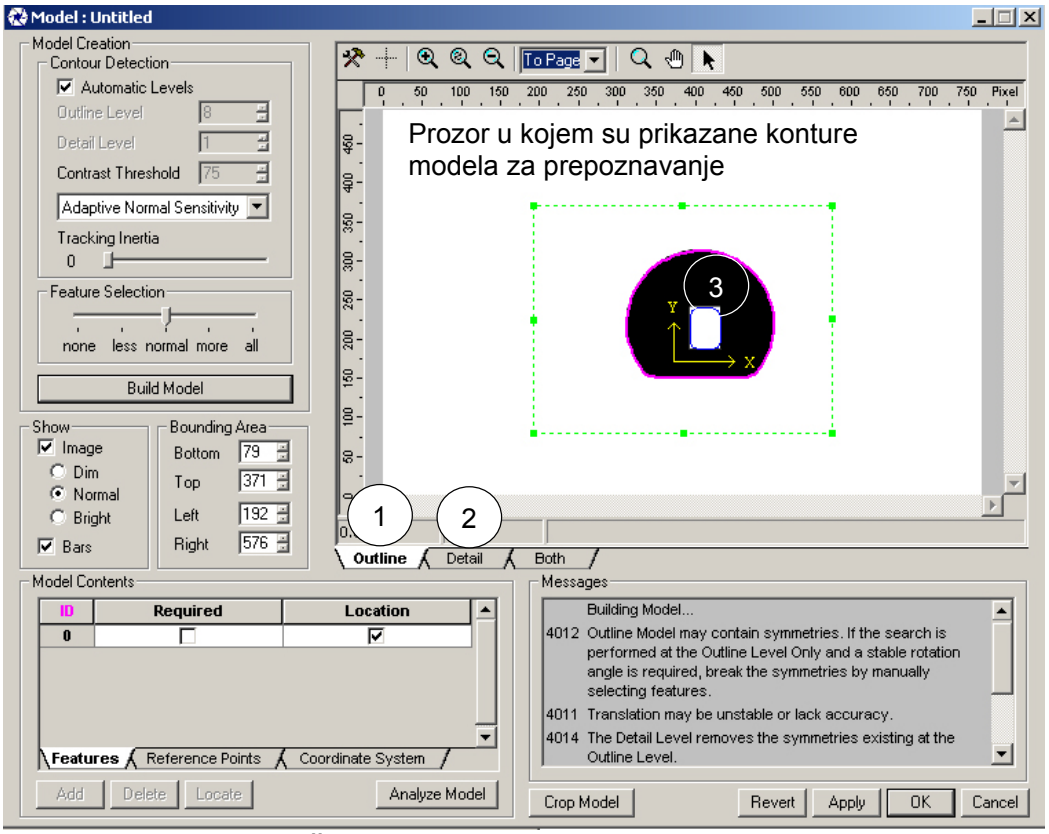
Izrada modela za prepoznavanje

Modeli za prepoznavanje, koje koriste *HEXSIGHT* vizijski alati, kreiraju se posebnim *HEXSIGHT* alatom na temelju referentne monokromatske slike objekta. Alatom se producira model za prepoznavanje koji je sačinjen od kontura, generiranih iz referentne monokromatske slike, a kojima su pridružena određena svojstva. Konture modela su matematički/analitički opisane kao spline krivulje.

Referentna monokromatska slika mora biti kalibrirana kako bi se mogle ispravno odrediti dimenzije modela. Nadalje, referentna slika treba biti "vrlo kvalitetna". Pod time se podrazumijeva da na slici nema izraženih sjena ili bljeskova, da su konture objekta oštre te da je izražen kontrast između objekta i pozadine. Drugim riječima, potrebna je slika objekta na temelju koje će se "lako" moći generirati konture koje odgovaraju stvarnim konturama objekta.

Uz pretpostavku da postoje CAD modeli dijelova koji će se sklapati, modeli za prepoznavanje se mogu generirati iz njih. Prednost ovog pristupa je uklanjanje svih pogrešaka na modelu koje nastaju kao posljedica snimanja kamerom. Potrebno je samo postaviti kalibriranu sliku generiranu iz CAD modela u radnu bazu kako bi bila dostupna programskoj podršci za izradu modela. Taj postupak će biti opisan u slijedećem poglavlju.

Alatu za izradu modela, ilustriranog slikom Sl.-57, pristupa se preko njegovog korisničkog sučelja, a modeli se kreiraju interaktivno. Na slici Sl.-57 je prikazan i model za prepoznavanje koji je grafički ilustriran u naznačenom prozoru.



Sl.-57 Sučelje alata za izradu modela za prepoznavanje.

Kreiranje modela ovim alatom sastoji se od sljedećih koraka:

- učitavanje referentne slike iz radne baze,
- kreiranje kontura na referentnoj slici,
- odabir odgovarajućih kontura ili njihovih segmenata,
- definiranje značajki odabranih kontura,
- određivanje položaja koordinatnog sustava objekta.

Konture se generiraju istim algoritmom kao i konture ulazne slike kod postupka prepoznavanja i lociranja objekata (8.3.1.2). Nadalje, kreiraju se dva skupa kontura s dva praga kao parametra algoritma.

Prvi prag je veći pa proces kreiranja kontura rezultira s manjim brojem kontura, a rezultat su *konture grube razine* (*Outline level contours)*. Drugi prag je manji i proces rezultira s više kontura. Rezultat su *konture fine razine* (*Detail level contours*). Oznakama #1 i #2, na slici Sl.-57, su označene kartice sa konturama grube i fine razine.

Dva skupa kontura, koja čine model, omogućuju ubrzavanje procesa prepoznavanja. Kod prepoznavanja hipoteze se generiraju i verificiraju na temelju *kontura grube razine*. Naime, *konture fine razine* se koriste kod konačnog, odnosno preciznog određivanja pozicija identificiranih objekata [47].

U model nije potrebno uključiti sve konture generirane na referentnoj slici. Uključuje se samo onoliko njihovih segmenata koliko je potrebno da bi objekti bili dovoljno dobro opisani.

Prilikom odabira kontura potrebno je pronaći odgovarajuću mjeru. Naime, što je manje kontura uključeno u model to je veća vjerojatnost da će model biti prepoznat, budući da se kod prepoznavanja konture modela nastoje uklopiti u konture generirane na slici sa kamere. No premalo kontura može uzrokovati da programska podrška pogrešno identificira objekte. Taj problem je posebno izražen kada se u radnom prostoru nalaze objekti s istim ili sličnim oblikovnim značajkama ili kada su slike s kamere degenerirane sjenama i bljeskovima koje uzrokuju generiranje "fantomskih" kontura.

Odabranim konturama se mogu dodijeliti značajke *lokacije (Location)* i *obavezne prisutnosti (Required)* te na taj način olakšati proces prepoznavanja.

Konture kojima je dodijeljena značajka *lokacije* koriste se prilikom određivanja pozicije objekta, odnosno moraju se nalaziti na očekivanoj poziciji.

Konture kojima je dodijeljena značajka *obavezne prisutnosti* obavezno moraju biti prisutne kako bi se neki skup krivulja proglasio objektom u procesu prepoznavanja.

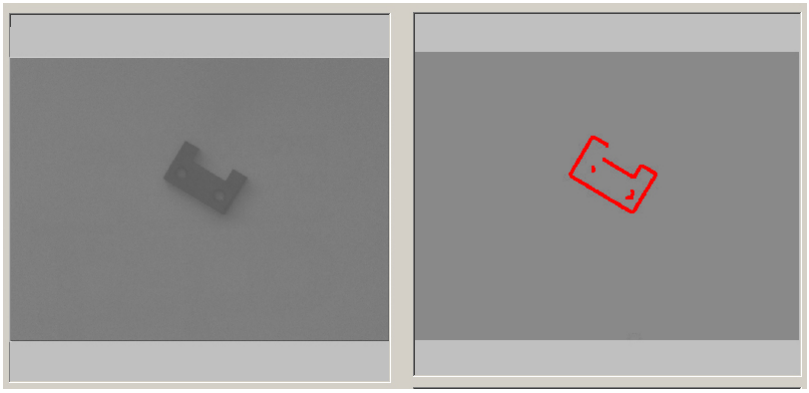
U modelu se definira i koordinatni sustav objekta. Pozicije i orijentacije lociranih objekata definirane su pozicijama i orijentacijama koordinatnih sustava objekta. Dakle, pozicija i orijentacija koordinatnog sustava objekta određuje način hvatanja ugradbenog dijela robotskom prihvatnicom. Koordinatni sustav objekta je označen oznakom #3 na slici Sl.-57.

Parametri programske podrške za prepoznavanje i lociranje objekata

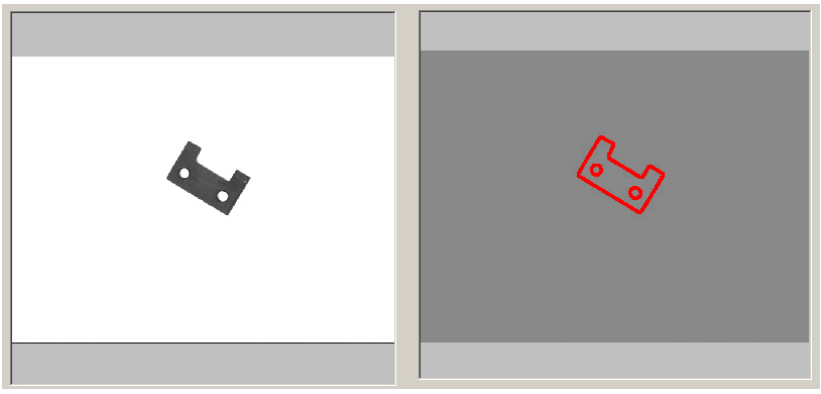
Definiranjem odgovarajućih parametara utječe se na postupak pronalaženja i lociranja objekata.

Parametar *minimalnog udjela modela (Minimum Model Percentage)* definira minimalan udio kontura modela koji se mora pronaći na slici da bi se skup kontura proglasio objektom. Parametar *minimalni udio obaveznih značajki (Minimum Percentage of Required Features)* definira minimalan udio kontura sa značajkom *obavezne prisutnosti.*

Ovakve tolerancije se moraju postaviti jer kod radnih uvjeta, zbog pojava sjena, bljeskova i slabog kontrasta na ulaznoj slici, često nije moguće pronaći sve konture objekta. Na slici Sl.-58 prikazana je slika objekta sa slabim kontrastom između objekta i pozadine te pronađene konture. Slika s kvalitetnim, velikim kontrastom i set pronađenih kontura koji potpuno opisuje objekt ilustriran je slikom Sl.-59.



Sl.-58. Slika objekta s niskim kontrastom (lijevo) i pronađene konture (desno).



Sl.-59. Slika objekta s visokim kontrastom (lijevo) na temelju koje su generirane konture koje potpuno opisuju objekt (desno).

Povećavanjem tolerancija raste vjerojatnost prepoznavanja objekata, no povećava se i vjerojatnost pogrešnog identificiranja. Ovaj problem je to izraženiji ako treba prepoznati više ugradbenih dijelova sa sličnim oblikovnim značajkama ili kod dijelova koje samo neki detalji čine nesimetričnim.

Ovaj problem se rješava tako da se konturama koje odgovaraju važnim oblikovnim značajkama za raspoznavanje dodaje značajka *obavezne prisutnosti*. Budući da su te konture važne za razlikovanje dijelova, parametar *minimalni udio obaveznih značajki* je obično veći.

Parametar *tolerancije prilagodbe (Conformity Tolerance)* definira maksimalno dozvoljeno lokalno odstupanje konture objekta kandidata od konture modela za prepoznavanje. Njegova veličina odgovara maksimalnoj udaljenosti, u kalibriranim jedinicama dužine, za koju konture objekta kandidata odstupaju od njihovih očekivanih pozicija prema modelu. Kao i kod ostalih tolerancija, ovdje treba pronaći mjeru između potrebe da se prepoznaju objekti i vjerodostojnosti rješenja. Osim na točnost prepoznavanja, ovim parametrom se utječe i na točnost pozicioniranja pronađenih dijelova.

Parametrom temeljitosti prepoznavanja definira se "intenzitet" postupka prepoznavanja. Povećavanjem iznosa ovog parametra raste vjerojatnost pronalaženja objekata, no raste i vrijeme izvođenja postupka.

Parametrom temeljitosti pozicioniranja definira se "intenzitet" postupka pozicioniranja. Povećavanjem iznosa ovog parametra raste točnost pozicija pronađenih objekata, ali raste i vrijeme izvođenja postupka pozicioniranja.

Parametar *uvećanja (Scale)* definira dozvoljeni faktor uvećanja pronađenog objekta u odnosu na model za prepoznavanje.

Broj objekata u skupu rezultata može se ograničiti parametrom *broj instanci (Instances To Find)*, a ukupno vrijeme izvođenja operacija prepoznavanja i lociranja objekata se može ograničiti parametrom *prekid (Timeout)*.

Redoslijed objekata u skupu rješenja alata za identificiranje i lociranje definira se parametrom *redoslijed instanci (Instance Ordering)*. Pronađeni objekti mogu biti poredani prema slijedećim redoslijedima, prema:

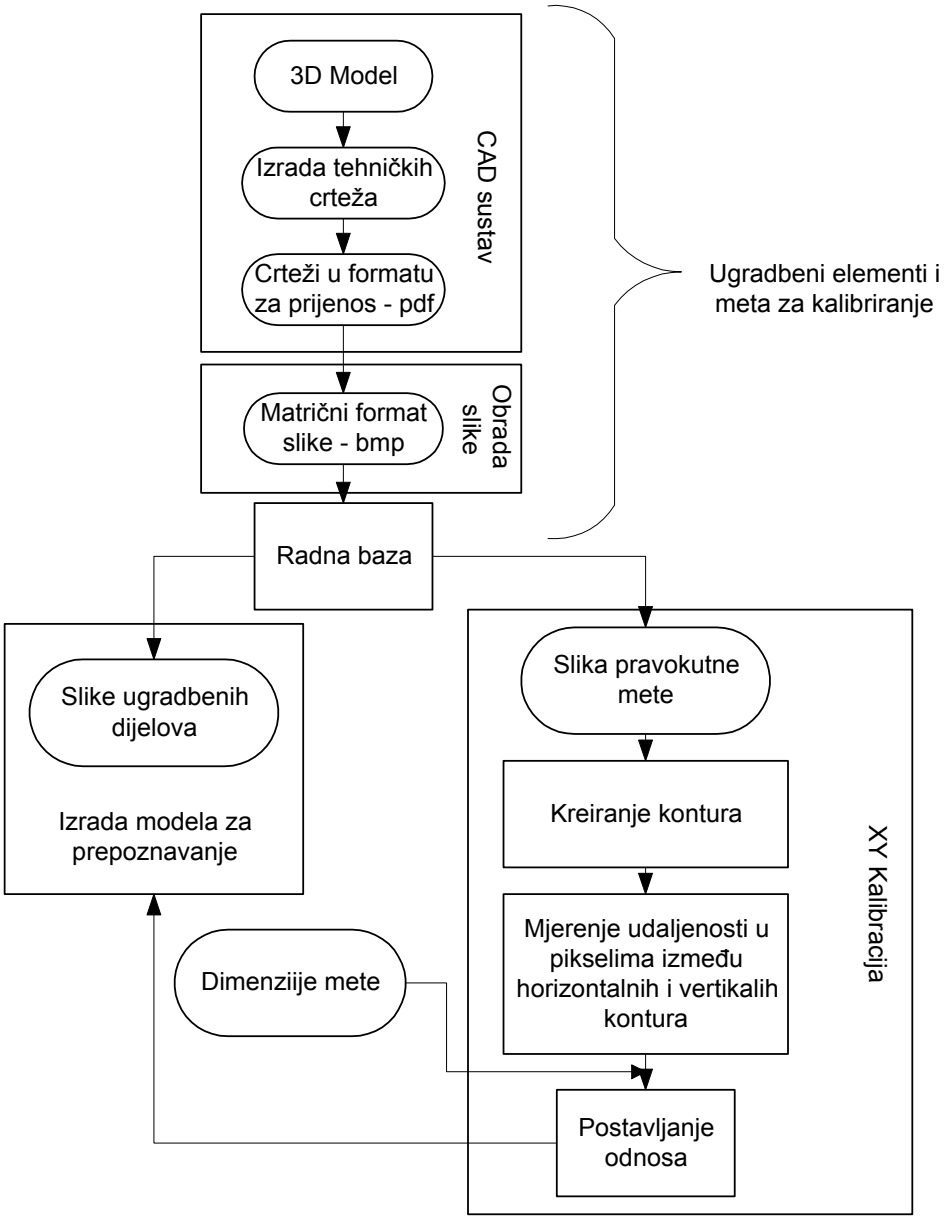
- dokazima (po snazi hipoteze prilikom kreiranja hipoteze),
- kvaliteti (po krajnjem stupnju sličnosti s modelom),
- udaljenosti od određene točke,
- s lijeva na desno, od gore prema dolje,
- ....

8.3.3. Izrada Modela za prepoznavanje iz CAD modela proizvoda

Alat za prepoznavanje i lociranje objekata te alat za kreiranje modela za prepoznavanje opisani su u prethodnim poglavljima. Modeli za prepoznavanje se kreiraju na temelju kalibrirane slike predmeta u radnoj bazi. Stoga se rješenje problema izrade modela za prepoznavanje iz CAD modela proizvoda svodi na:

- kreiranje digitalnih slika odgovarajućih projekcija proizvoda,
- kreiranje emulacijske baze s monokromatskim slikama iz CAD-a,
- kalibriranje alata za akviziciju prema slikama iz CAD-a.

Cjeloviti postupak izrade modela za prepoznavanje iz CAD modela proizvoda je prikazan slikom Sl.-60.



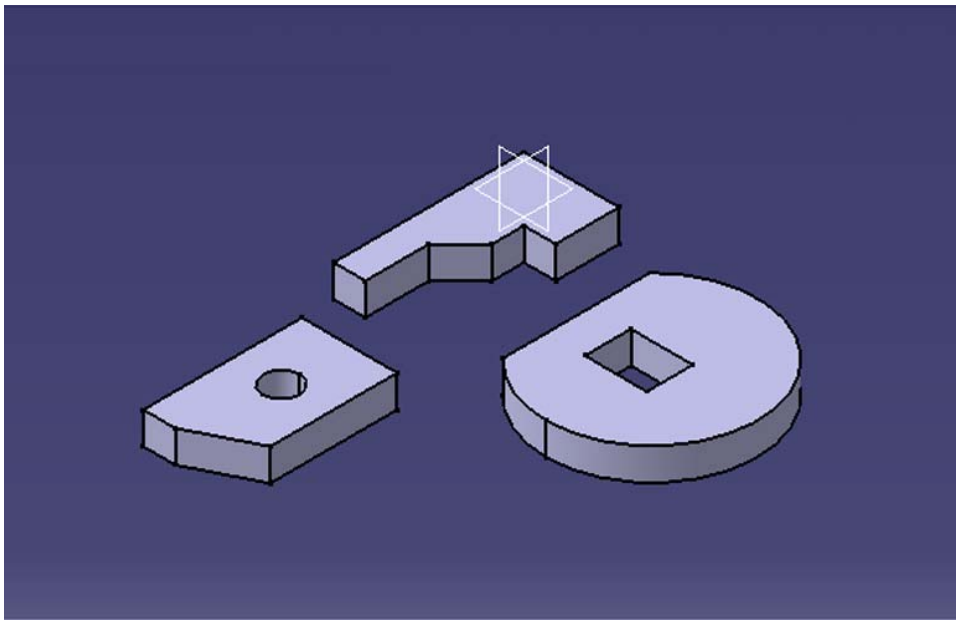
Sl.-60. Postupak izrade modela za prepoznavanje iz CAD modela proizvoda

8.3.3.1. Izrada slika

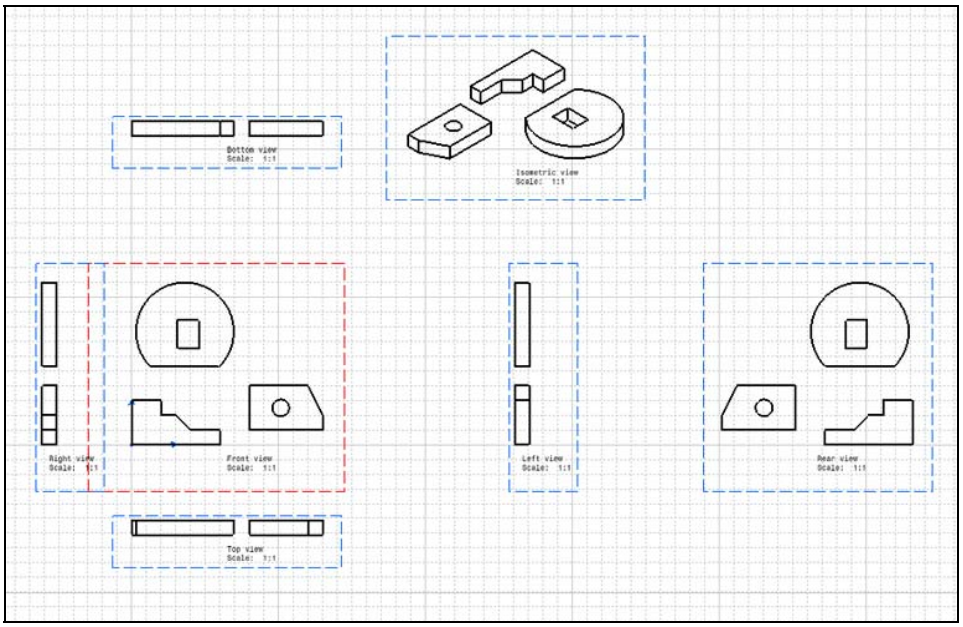
Postupak kreiranja digitalnih slika započinje od 3D CAD modela proizvoda. Na slici Sl.-61 prikazani su 3D CAD modeli ugradbenih dijelova kreiranih CAE aplikacijom - Dassault Systemes CATIA V5 [48].

Na temelju 3D CAD modela kreiraju se 2D projekcije u obliku tehničke dokumentacije, odnosno tehničkih crteža. CAE sustav posjeduje programski modul za izradu tehničke dokumentacije. Crteži se inicijalno pohranjuju u 2D vektorskom formatu CATDrawing.

Karakteristika vektorskih formata jest da su sve dimenzije definirane fizikalnim jedinicama dužine, dakle neovisne o razlučivosti slike. Slikom Sl.-62 je prikazan tehnički crtež, u vektorskom formatu, kreiran na temelju 3D CAD modela.



Sl.-61. 3D CAD modeli ugradbenih dijelova u CATPart formatu.



Sl.-62. Projekcije 3D modela kao tehnički crtež u vektorskom 2D CATDrawing formatu.

Slike za emulacijsku bazu moraju biti u 256 bit-nom MS Windows BMP formatu. Taj format je kompatibilan monokromatskim slikama s digitalizatora.

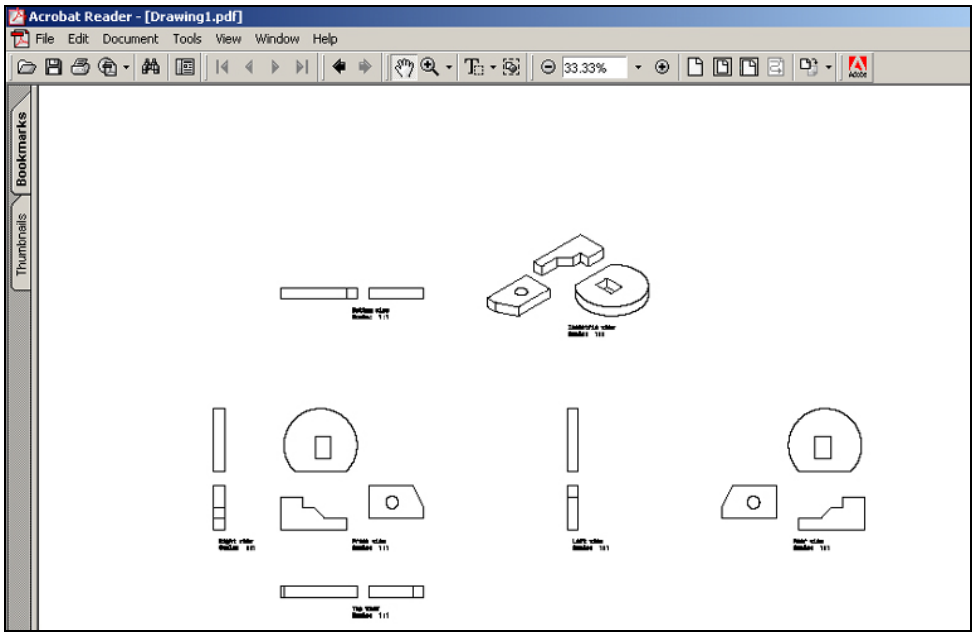
Ovakve slike nije moguće napraviti izravno u CAE aplikaciji. Stoga se crtež pohranjuje u standardni vektorski format kako bi se učitao u programsku podršku za obradu matričnih slika. Slika Sl.-63 prikazuje tehnički crtež u "standardnom" vektorskom formatu Adobe PDF [49].

Ovaj format crteža se može učitati u komercijalnu programsku podršku za obradu matričnih slika Adobe Photoshop [49]. Pod matričnim slikama[24] se razumijevaju slike definirane položajima i intenzitetima piksela.

Prilikom učitavanja vektorskog formata definira se prostorna razlučivost matrične slike. Pošto se tehnički crtež već nalazi u programu za uređivanje slike, nepotrebni pogledi se mogu izbaciti.

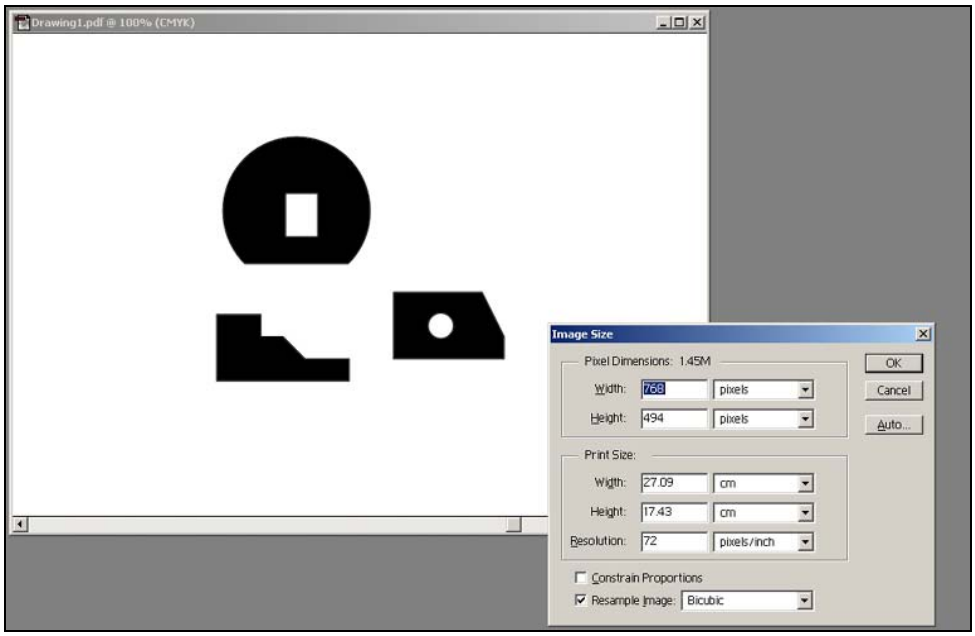
Nadalje, područje unutar kontura (objekt) se mora ispuniti bojom kako je to prikazano na slici Sl.-64. Naime, ako bi se ispustio ovaj korak linije iz tehničkog crteža prelaskom u matrični oblik postaju nizovi piksela. Algoritam za kreiranje kontura u ovakvom slučaju pronalazi jednu konturu sa svake strane linije iz tehničkog crteža na način kako je to ilustrirano slikom Sl.-65.

Sliku objekta obrađenu na opisani način potrebno je još samo pohraniti u zahtijevanom *Windows* BMP formatu.

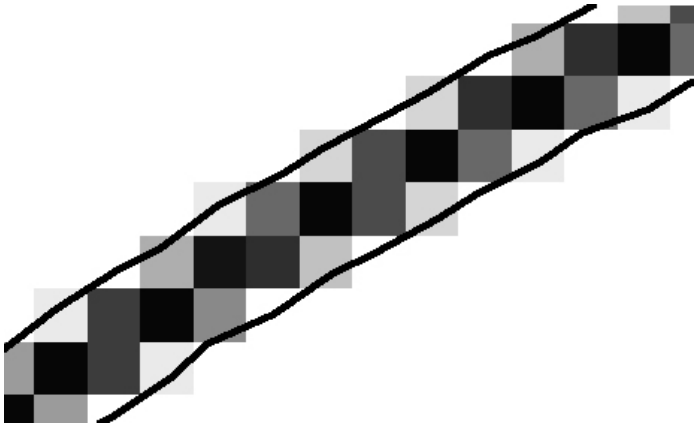


Sl.-63. CATDrawing prevoditeljem preveden u vektorski PDF format.

[24] Matrične slike ili bit-mape.



Sl.-64. Vektorski PDF format učitan u program za obradu slike.



Sl.-65. Linija u matričnoj reprezentaciji slike ima površinu i u stvari predstavlja područje koje je opisano sa dvije konture.

Emulacijska baza slika se kreira programskom podrškom za akviziciju i kalibriranje digitalne slike. Potrebno je samo pokazati gdje se u zapisničkom sustavu nalaze MS Windows BMP slike predmeta.

8.3.3.2. Kalibracija za izradu modela

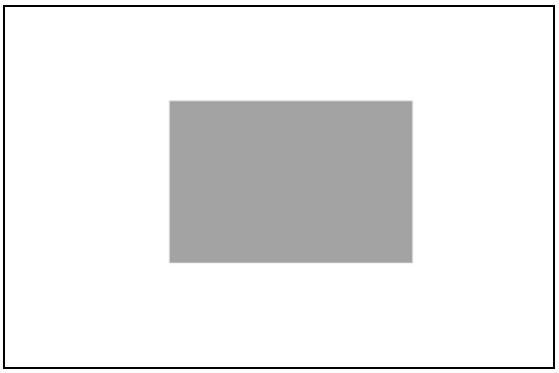
Kao i za slike s digitalizatora, potrebno je izvršiti kalibraciju alata za akviziciju slike, budući da se modeli za prepoznavanje uvijek moraju kreirati na temelju kalibrirane slike [47].

Kalibriranje alata za akviziciju slike može se izvršiti XY kalibracijom. Nedostaci XY kalibracije su irelevantni, jer slike generirane iz CAD-a nisu degenerirane greškama kamere. XY kalibracijom se uspostavlja odnos između koordinatnog sustava slike (pikseli) i

koordinatnog sustava svijeta (fizikalne jedinice dužine). Stoga je potrebna samo digitalna slika predmeta poznatih dimenzija.

Kalibracija se u načelu može načiniti i digitalnom slikom ugradbenog dijela čiji model kreiramo, pod uvjetom da posjeduje odgovarajuće oblikovne značajke. Pod odgovarajućim oblikovnim značajkama se podrazumijeva par ravnih vertikalnih i horizontalnih kontura na slici predmeta.

Dijelovi složene geometrije često ne posjeduju takve značajke pa se kalibriranje izvodi pravokutnom metom (Sl.-66). Odnos razlučivosti slike mete i dimenzija mete mora biti identičan tom odnosu na slikama predmeta.



Sl.-66. Meta za XY kalibraciju.

Slika pravokutne mete se kreira i dodaje u emulacijsku bazu istim postupkom kao i slike predmeta čije modele kreiramo. Alatom za akviziciju postavlja se sliku mete iz emulacijske baze u radnu bazu. Nakon toga se pokreče alat za XY kalibraciju.

Alatom za XY kalibraciju najprije se generiraju konture na temelju slike mete, a zatim se mjeri udaljenost između para horizontalnih i vertikalnih kontura u pikselima. Definiranjem visine i širine pravokutnika uspostavlja se odnos između fizikalnih jedinica dužine i dužina izraženih pikselima, odnosno vrši se kalibracija.

Učitavanjem slika ugradbenih elemenata iz emulacijske baze u radnu bazu omogućuje se izrada odgovarajućih modela za prepoznavanje pomoću alata za kreiranje modela za prepoznavanje.

8.3.4. Povezivanje vizijskog koordinatnog sustava i koordinatnog sustava robota

Programska podrška za prepoznavanje i lociranje objekata producira koordinate pronađenih dijelova u vizijskom koordinatnom sustavu. Kako bi te informacije bile uporabljive za upravljačku jedinicu robota, potrebno je izvršiti homogenu transformaciju u koordinatni sustav robota.

Homogene transformacije definiraju položaj i orijentaciju jednog koordinatnog sustava u odnosu prema drugome [37]. Matrica homogenih transformacija definirana je jednadžbom (70).

$$A = \begin{bmatrix} i_x & j_x & k_x & p_x \\ i_y & j_y & k_y & p_y \\ i_z & j_z & k_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{70}$$

Prva tri stupca predstavljaju projekcije jediničnih vektora početnog koordinatnog sustava na ciljni. Četvrti je stupac daje koordinate ishodišta ciljnog koordinatnog sustava u početnom koordinatnom sustavu (70).

Pri tome se pod početnim koordinatnim sustavom podrazumijeva onaj koordinatni sustav u kojem su nam poznate koordinate točaka, a ciljni je onaj za koje koordinate računamo.

Ako je u početnom koordinatnom sustavu zadana točka p_1, i ciljni je koordinatni sustav prema početnom definiran matricom A, tada je točka p_0 u ciljnom koordinatnom sustavu definirana izrazom (71).

$$p_0 = A\, p_1 \tag{71}$$

Zbog pravila množenja matrica točke se u izraz (71) uvrštene kao vektori $p_0\ (x_0, y_0, z_0, 1)$ *i* $p_1\ (x_1, y_1, z_1, 1)$.

Iako vizijski koordinatni sustav i koordinatni sustav robota mogu biti međusobno postavljeni općenito, kod realiziranog sustava uzeto je da su vertikalne osi z dvaju koordinatnih sustava paralelne. To je tehnološko opravdana pretpostavka koja se ostvaruje tako da se osigura horizontalnost stola robota i radne površine pokrivene kamerom. Nadalje, zanemarena je i razlika u Z koordinati, jer je korekcija visine stola uključena je u programsku podršku za upravljanje robotom.

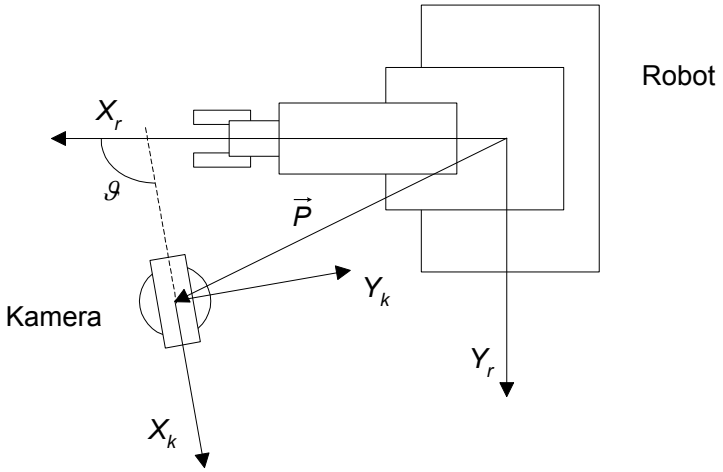
Shodno pretpostavkama, matrica homogenih transformacija poprima oblik definiran izrazom (72).

$$A = \begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 & p_x \\ \sin\vartheta & \cos\vartheta & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{72}$$

Pri tome je s ϑ definiran kut rotacije oko vertikalne osi koordinatnog sustava robota u odnosu na vizijski koordinatni sustav.

Prema izrazu (72), da bi se definirala matrica homogenih transformacija iz vizijskog koordinatnog sustava u koordinatni sustav robota, potrebno je definirati položaj ishodišta vizijskog koordinatnog sustava u koordinatnom sustavu robota te kut rotacije. Odnos koordinatnog sustava robota i vizijskog sustava je prikazan slikom Sl.-67. Horizontalne osi koordinatnog sustava robota su označene indeksom r, a osi vizijskog koordinatnog sustava su

označene indeksom k. Vektor P definira položaj ishodišta vizijskog koordinatnog sustava, a kut ϑ definira njegovu rotaciju.



Sl.-67. Odnos koordinatnih sustava robota i vizijskog koordinatnog sustava.

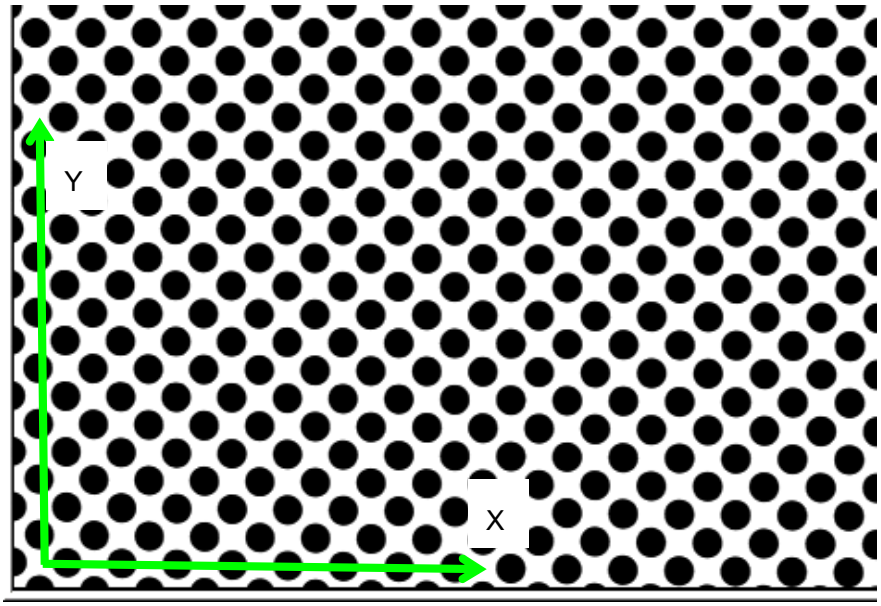
Vektor P i kut ϑ se mogu izračunati prilikom postupka kalibracije kamere distorzijskim modelom.

Kod kalibracije kamere distorzijskim modelom ishodište vizijskog koordinatnog sustava se postavlja u kameri vidljivu kalibracijsku točku najbližu donjem lijevom uglu slike. Os X koordinatnog sustava je poravnata s retkom u kojoj se nalazi ta točka.

Položaj i orijentacija koordinatnog sustava slike, kod kalibracije distorzijskim modelom, su prikazani slikom Sl.-68. Vektor P se određuje tako da se vrh ticala montiranog na robotsku ruku dovede u točku u kojoj je ishodište. Položaj ticala u koordinatama robota se može dobiti pomoću upravljačke jedinice robota. Vektor P je definiran izrazom (73), pri čemu su x_1 i y_1 koordinate točke ishodišta.

$$\vec{P} = (x_1, y_1) \tag{73}$$

Da bi se odredio kut ϑ potrebne su koordinate bilo koje točke iz reda točaka koji se poklapa s osi X. Do koordinata se također dolazi pomoću ticala.



Sl.-68. Položaj i orijentacija koordinatnog sustava slike kod kalibracije distorzijskim modelom.

Kut ϑ je definiran izrazima (74) - (81). Pri čemu su x_1 i y_1 koordinate točke ishodišta, a x_2 i y_2 koordinate točke na osi X. Sučelje za unos koordinata ishodišta i točke na osi je prikazano slikom Sl.-52.

$$x_1 > x_2 \quad y_1 < y_2 \qquad \vartheta = \frac{\Pi}{2} + a\tan\frac{x_1 - x_2}{y_1 - y_2} \tag{74}$$

$$x_1 < x_2 \quad y_1 < y_2 \qquad \vartheta = a\tan\frac{y_2 - y_1}{x_2 - x_1} \tag{75}$$

$$x_1 < x_2 \quad y_1 > y_2 \qquad \vartheta = \frac{2\Pi}{2} - a\tan\frac{y_1 - y_2}{x_2 - x_1} \tag{76}$$

$$x_1 > x_2 \quad y_1 > y_2 \qquad \vartheta = \Pi + a\tan\frac{y_1 - y_2}{x_1 - x_2} \tag{77}$$

$$x_1 = x_2 \quad y_1 > y_2 \qquad \vartheta = \frac{\Pi}{2} \tag{78}$$

$$x_1 = x_2 \quad y_1 < y_2 \qquad \vartheta = \frac{3\Pi}{2} \tag{79}$$

$$x_1 <= x_2 \quad y_1 = y_2 \qquad \vartheta = 0 \tag{80}$$

$$x_1 > x_2 \quad y_1 = y_2 \qquad \vartheta = \Pi \tag{81}$$

Na temelju kuta ϑ i vektora P konstruira se matrica homogenih transformacija prema izrazu (71), a koordinate se preračunavaju prema izrazu (70).

8.3.5. Upravljački program robota

Radom robota, kao izvršnog elementa/člana sustava, upravlja program koji se izvodi na upravljačkoj jedinici robota. Na temelju ulaznih parametara upravljački program definira gibanje robota koje omogućuje izvođenje montaže proizvoda.

Upravljački program uzima dva skupa ulaznih parametara/podataka kojima su definirane montažne operacije za pojedini proizvod. Prvi skup parametara su pozicije i orijentacije ugradbenih dijelova u radnom prostoru robota. One su određene sustavom strojnog vida i znanjem o vertikalnoj komponenti točke hvatanja.

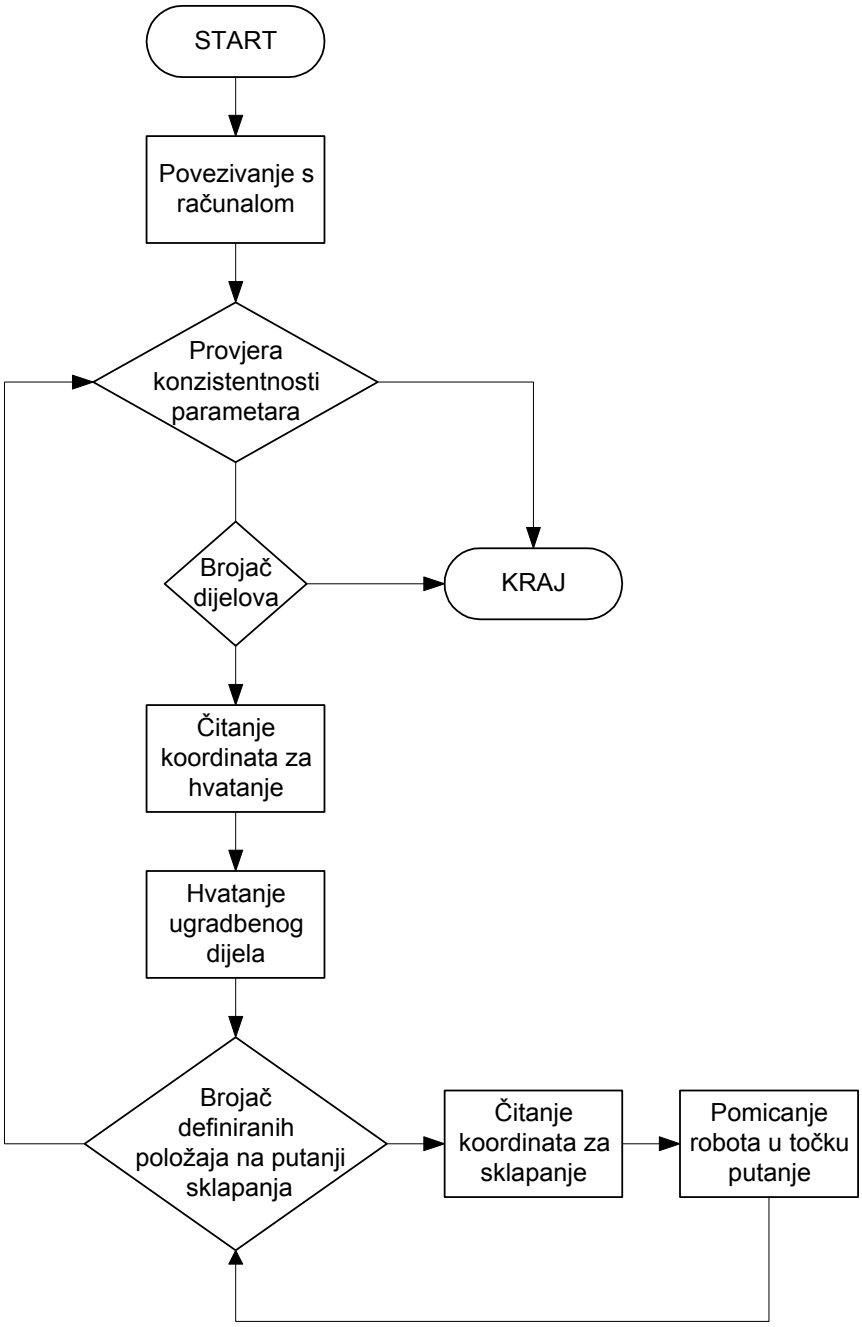
Drugi skup parametara određuje putanje sklapanja ugradbenih dijelova. Montažne operacije, odnosno putanje sklapanja su definirane planom montaže.

Upravljački program robota je razvijen u programskom jeziku V^+. Naime, V^+ je programski jezik razvijen za *Adept Technology* industrijske robote, vizijske sustave i sustave za upravljanje gibanjem. Pomoću V^+ programskog jezika izračunavaju se kontinuirane putanje u realnom vremenu, čime je omogućeno brzo izvođenje složenih gibanja.

Jezik V^+ omogućuje uobičajenu funkcionalnost suvremenih programskih jezika poput:

- potprograma i funkcija,
- kontrole toka izvođenja,
- višezadaćnog okruženja,
- rekurzivnog izvođenja programa [50].

Struktura upravljačkog programa robota je prikazana slikom Sl.-69. Programski kôd je dan u prilogu P.2.



Sl.-69. Struktura upravljačkog programa robota.

8.3.5.1. Povezivanje s računalom

Na početku izvođenja potrebno se povezati s PC računalom, budući da se na njemu nalaze zapisnici u kojima su pohranjeni parametri za izvođenje programske podrške. Povezivanje s računalom je ostvareno NFS protokolom. NFS je protokol koji omogućuje integriranje zapisničkih sustava u homogenim i heterogenim računalnim mrežama. Prijenos podataka je ostvaren TCP/IP protokolom.

Program otvara tri veze koje se otvaraju zbog toga što su tijekom izvođenja programa istovremeno otvorena za čitanje tri zapisnika, a prilikom čitanja zapisnika se adresira veza.

Za čitanje se otvaraju:

- zapisnik u koji su zapisane pozicije i orijentacije ugradbenih dijelova,
- zapisnik sa visinama točke hvatanja,
- zapisnik u kojem su opisane putanje sklapanja.

8.3.5.2. Konzistentnost zapisnika s parametrima

U upravljački program je uključena osnovna provjera konzistentnosti parametara pročitanih iz zapisnika. Provjerava se da li zapisnici odgovaraju predviđenom planu montaže na temelju broja ugradbenih elemenata te njihovih imena.

Čitanje zapisnika se izvodi zapis po zapis. Pod zapisom se razumijevaju znakovi u tekstualnom zapisniku odvojeni *new line (\n)* znakom.

Prvi pročitani zapis iz svih zapisnika je broj ugradbenih dijelova. Ukoliko su brojevi ugradbenih dijelova definirani u zapisnicima različiti, izvođenje programske podrške se prekida. Nadalje, ukoliko se utvrdi da imena dijelova, koja su u svim zapisnicima poredana po redoslijedu sklapanja, nisu konzistentna izvođenje programa se prekida.

8.3.5.3. Hvatanje ugradbenog dijela

Gibanje robota prilikom hvatanja ugradbenog dijela je podijeljeno na tri faze.

1. pozicioniranje robota iznad odgovarajućeg ugradbenog dijela,
2. pozicioniranje robota za hvatanje,
3. udaljavanje robota od radne površine.

Sve tri faze gibanja robota mogu se definirati točkom hvatanja budući da se faze približavanja i udaljavanja mogu definirati V^+ funkcijama. Funkcije `APPRO` i `DEPART` definiraju približavanje odnosno udaljavanje od zadane točke, po Z osi alata u orijentaciji zadane točke. Duljina puta približavanja i udaljavanja je definirana parametrom funkcije [37][50].

Točka za hvatanje, koja se zadaje kao parametar za sva tri segmenta gibanja, je definirana sa tri prostorne koordinate i tri eulerova kuta [37]. Pri tome su kutovi ν i φ konstantni i iznose 0^o i 180^o.

Vertikalna koordinata Z je definirana visinom radne površine u koordinatnom sustavu robota kojoj je dodana visina točke hvatanja svakog pojedinog dijela relativno na radnu površinu. Visina točke hvatanja je definirana u zapisniku s parametrima.

Preostale koordinate X, Y, ψ su definirane su u zapisniku s pozicijama i orijentacijama ugradbenih dijelova.

Zapisnik u koji su zapisane pozicije i orijentacije ugradbenih dijelova je prikazan u prilogu P.3. U prvom retku je zapisan ukupan broj ugradbenih dijelova. Slijede skupine od po četiri zapisa u kojima su zapisani redom ime ugradbenog dijela, X i Y koordinata te kut valjanja ψ.

U zapisniku s visinama točke hvatanja nakon prvog retka s brojem dijelova slijede parovi zapisa. Prvi zapis je ime dijela po redoslijedu hvatanja, a drugi udaljenost točke hvatanja od radne površine.

Gibanje robota u točku se ostvaruje V^+ naredbom `MOVE` s definicijom položaja kao parametrom.

8.3.5.4. Izvođenje operacije sklapanja

Nakon hvatanja dijela programska podrška započinje ciklus operacije sklapanja. Operacija sklapanja je definirana putanjom sklapanja određenom planom montaže.

Putanja sklapanja je definirana vektorima položaja i orijentacija r. Koordinate položaja i eulerovi kutovi se čitaju iz zapisnika te se zadaju kao parametar za svaki segment gibanja. Zapisnik s putanjama sklapanja mora opisivati položaje koje osim samih putanja umetanja definiraju putanju približavanja i odmicanja robota od sklopa.

U zapisniku s putanjama sklapanja prvi redak je ukupan broj dijelova. Njemu slijedi definicija putanja. Definicija putanje se sastoji od zapisa s imenom dijela, ukupnog broja položaja koje definiraju jednu putanju te zapisima s koordinatama. Svaki položaj na putanji je definiran sa po 6 zapisa, odnosno s koordinatama X, Y, Z, ν, φ i ψ.

Ciklus operacije sklapanja završava s pozicioniranjem u zadnju pročitanu točku. Izvođenje programske podrške se nastavlja s čitanjem pozicije i orijentacije te hvatanja slijedećeg dijela prema redoslijedu sklapanja sve dok nisu sklopljeni svi dijelovi.

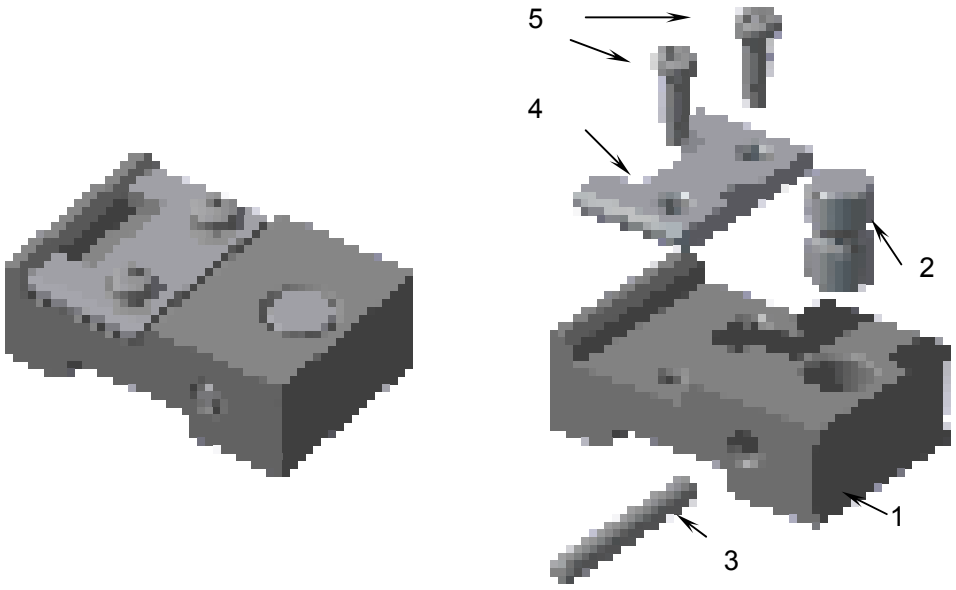
9 PRIMJENA I VERIFIKACIJA PROGRAMSKE PODRŠKE

Primjena i verifikacija sustava za automatsko sklapanje u nesređenoj robotskoj okolini prikazat će se na primjeru sklapanja jednostavnog sklopa prikazanog slikom Sl.-70. Sklop je oblikovan posebno za istraživanja automatskih montažnih sustava i demonstraciju njihove primjene. Njegova posebnost je da sadrži oblikovne značajke koje omogućuju primjenu najvažnijih i najčešćih montažnih operacija.

Sklop se sastoji od 5 različitih dijelova. Prema slici Sl.-70 to su:

1. bazni dio,
2. valjak,
3. zatik,
4. pločica,
5. vijci.

Ugradbeni dijelovi su numerirani shodno primijenjenom redoslijedu sklapanja.



Sl.-70. Sklop razvijen za istraživanje automatskih montažnih sustava.

9.1. Priprema programske podrške strojnog vida.

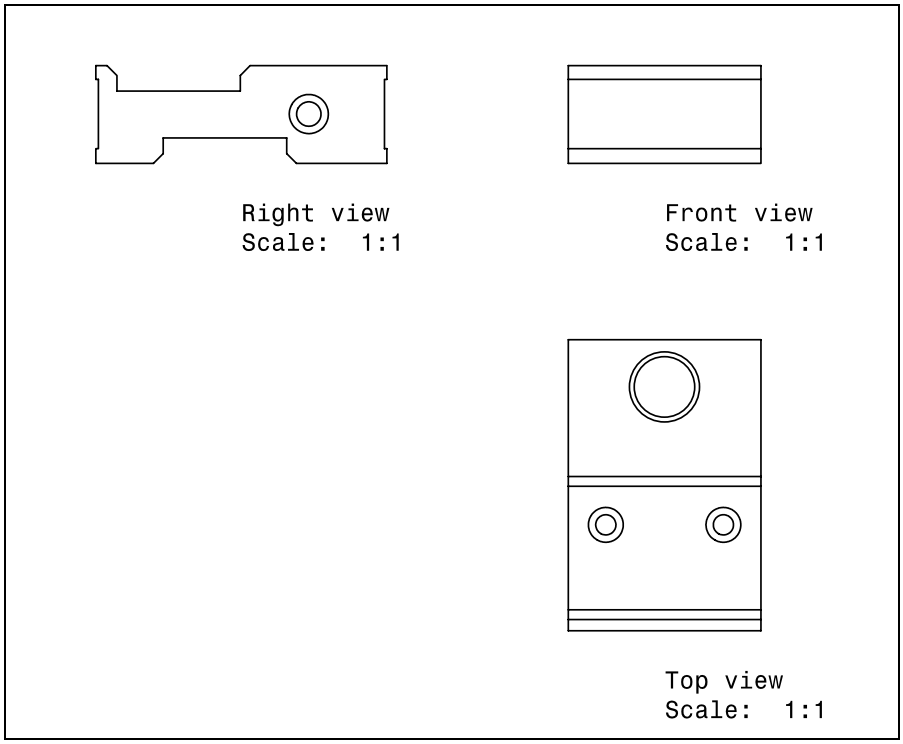
Kako bi se mogli prepoznati i locirati ugradbeni dijelovi sustavom strojnog vida potrebno je prethodno definirati parametre programske podrške karakteristične za proizvod koji će se sklapati, kao i neke parametre koji opisuju konfiguraciju samog sustava.

Priprema programske podrške strojnog vida uključuje:

- izradu modela za prepoznavanje,
- definiranje parametara programske podrška za prepoznavanje i lociranje objekata,
- postavljanje odnosa između vizijskog koordinatnog sustava i koordinatnog sustava robota.

9.1.1. Modeli za prepoznavanje

Kao što je opisano u poglavlju 8.3.3 izrada modela počinje od 3D CAD modela, odnosno od tehničkih crteža kreiranih na temelju njega. Tehnički crtež baznog dijela u vektorskom CATDrawing formatu je prikazan slikom Sl.-71. Kreirano je ukupno pet tehničkih crteža, po jedan za svaki ugradbeni dio.

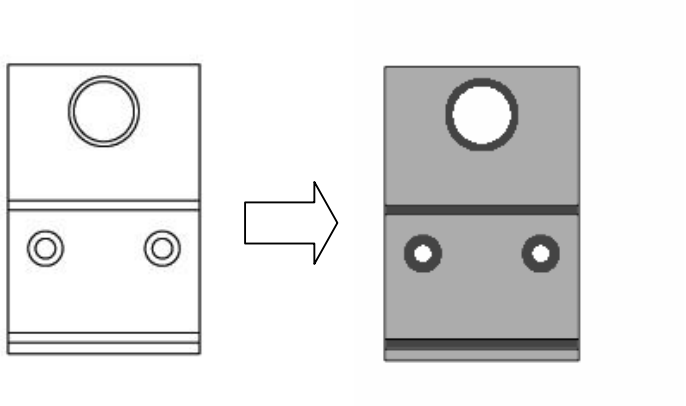


Sl.-71. Tehnički crtež baznog dijela kreiran CAD sustavom na osnovu 3D CAD modela proizvoda.

Tehnički crteži u vektorskom formatu učitani su u program za obradu slika gdje su izbačene suvišne projekcije. Nadalje, područje unutar kontura se ispunjava s dvije nijanse sive boje.

Potrebne su dvije nijanse sive kako bi se mogle generirati sve potrebne konture. Naime, da bi se mogle razlikovati dvije orijentacije baznog dijela, model mora sadržavati i unutrašnje konture. Pod unutrašnjim konturama se razumijevaju konture koje odjeljuju regije istog objekta.

Na slici Sl.-72 je prikazana matrična digitalna slika kontura baznog dijela i slika objekta s regijama ispunjenim bojom kreirana na temelju nje. Analogno ovoj slici, kreirane su i slike ostalih ugradbenih elemenata.



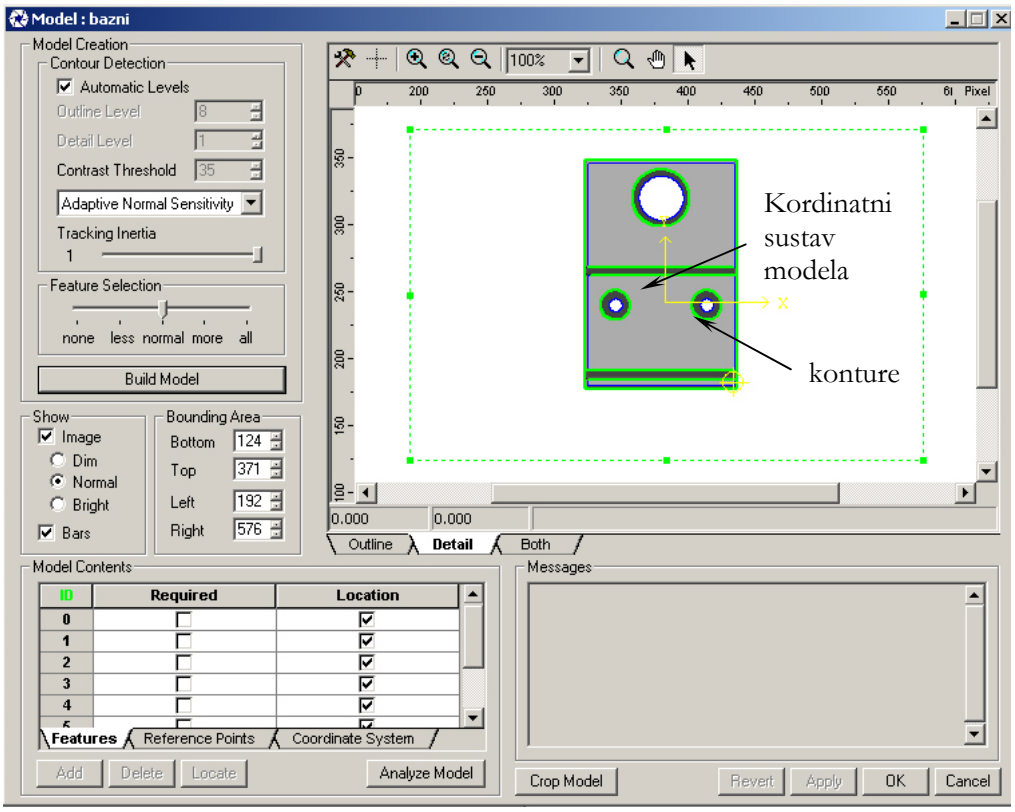
Sl.-72. Matrična digitalna slika kontura baznog (lijevo) dijela i slika objekta s regijama ispunjenim bojom (desno).

Iz matričnih slika u Windows BMP formatu se kreira emulacijska baza iz koje se slike postavljaju u radnu bazu. Naravno, sustav je prethodno kalibriran XY kalibracijom. Ovaj postupak je opisan u poglavlju 8.3.3.

Pošto se kalibrirane slike ugradbenih dijelova nalaze u radnoj bazi pokreće se alat za izradu modela za prepoznavanje. Sučelje alata za izradu modela, prilikom izrade model za prepoznavanje baznog dijela, je prikazano slikom Sl.-73.

Model sačinjava 8 kontura koje su na slici prikazane zelenim linijama. Osim kontura, definirana je pozicija i orijentacija koordinatnog sustava modela. Ishodište koordinatnog sustava odgovara točki hvatanja, a orijentacija koordinatnog sustava modela definira orijentaciju ugradbenog dijela.

Na analogan način kreirani su i modeli za prepoznavanje ostalih ugradbenih elemenata.



Sl.-73. Model za prepoznavanje baznog dijela.

9.1.2. Parametri programske podrška za prepoznavanje i lociranje objekata

Parametri programske podrške za prepoznavanje i lociranje objekata opisani su u poglavlju 8.3.2.

Vrijednosti parametara za prepoznavanje ugradbenih elemenata sklopa iz ovog primjera određivane su interaktivno modifikacijama predloženih vrijednosti. Pokazalo se da, pri danim uvjetima osvijetljenosti radnog prostora, programska podrška postiže najbolje rezultate ako parametri imaju sljedeće vrijednosti:

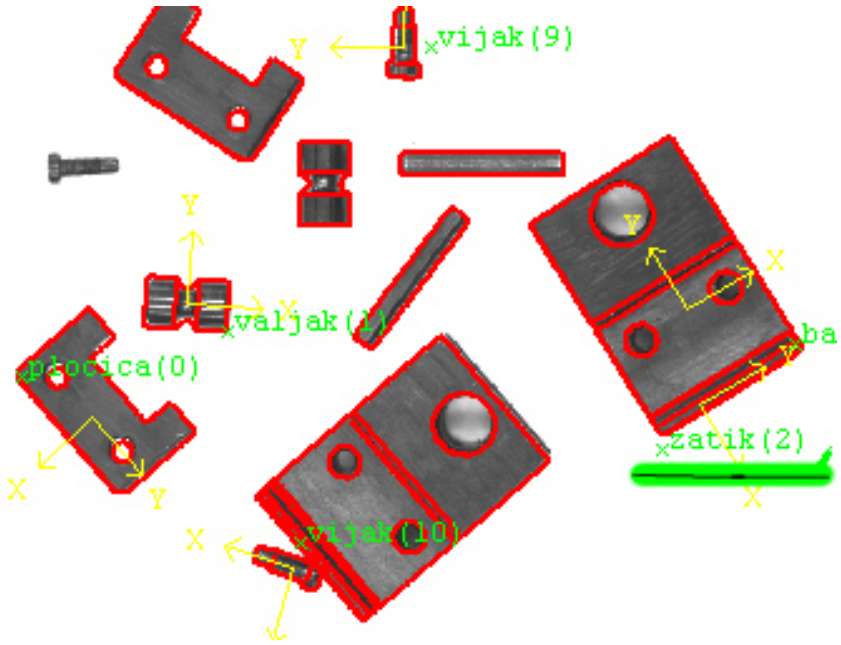
parametar *minimalnog udjela modela* = 65%
parametar *minimalnog udjela obaveznih značajki* = 100%
parametar temeljitosti prepoznavanja = 60%
parametar temeljitosti pozicioniranja = 70%
parametar *tolerancije prilagodbe* = 0.747

Dakako, za neke druge uvjete osvjetljenosti prizora ili vizualne i oblikovne značajke objekata koji se prepoznaju, vrijednosti ovih parametara se moraju prilagoditi tim, drugačijim, uvjetima na prizoru kako bi se postigli optimalni rezultati procesa prepoznavanja.

Parametar *minimalnog udjela obaveznih značajki* definira minimalan udio kontura s značajkom *obavezne prisutnosti*. Postavljen je na maksimalni iznos čime je naprosto ukinuta tolerancije za te konture.

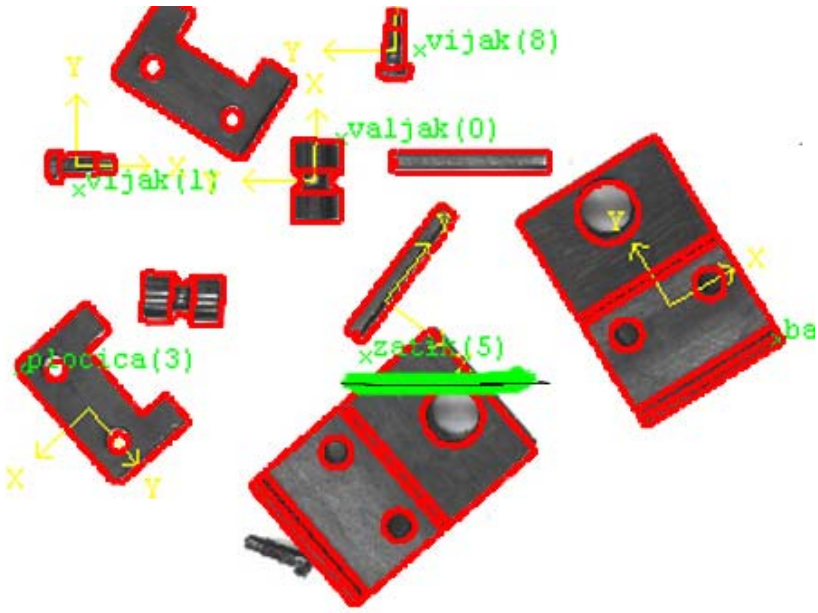
Jedini model koji sadrži konture sa značajkom *obavezne prisutnosti* je model za prepoznavanje zatika. Bez obzira što zatik na prvi pogled izgleda kao najjednostavniji dio za prepoznavanje, kod podešavanja parametara programske podrške pokazalo se da je najviše grešaka uzrokovano upravo zatikom.

Zatik, odnosno model za prepoznavanje zatika, uzrokuje poteškoće kod prepoznavanja zbog toga što su njegove konture vrlo slične konturama jedne oblikovne značajke baznog dijela. Primjer pogrešnog rješenja procesa prepoznavanja s pogrešno identificiranim zatikom je prikazan slikom Sl.-74. Konture koje čine model zatika, a moraju biti pažljivo odabrane na način da se naglasi zakrivljenost kontura na skošenjima zatika, a što se postiže tako da su konture kojima su opisani krajevi zasebne značajke. Na slici Sl.-75 je shematski prikazan model za prepoznavanje zatika. Na temelju tog modela za prepoznavanje je dobiveno ispravno rješenje procesa prepoznavanja prikazano slikom Sl.-76. Sve konture modela zatika imaju pridruženu značajku *obavezne prisutnosti*



Sl.-74. Pogrešno rješenje prepoznavanja dijelova uzrokovano preniskim tolerancijama modela za prepoznavanje zatika.

Sl.-75. Shematski prikaz modela za prepoznavanje zatika.



Sl.-76. Ispravno interpretirana slika.

Vrijednosti parametara temeljitosti prepoznavanja i lociranja predstavljaju kompromis između vremena potrebnog za izvršenje procesa prepoznavanja i pouzdanosti rješenja.

Potrebno je definirati i redoslijed sklapanja kako bi rješenje bilo u prikladnom obliku za programsku podršku za upravljanje robotom. Redoslijed sklapanja je definiran u zapisniku *redos.txt*.

9.2. Priprema upravljačkog programa robota

Priprema upravljačkog programa robota uključuje izradu dva zapisnika. Prvi zapisnik, *vis_ins.txt* sadrži visine točaka hvatanja ugradbenih dijelova. Sadržaj zapisnika *vis_ins.txt* je :

```
6
bazni
20
valjak
14
zatik
6
plocica
10
vijak
6
vijak
6
```

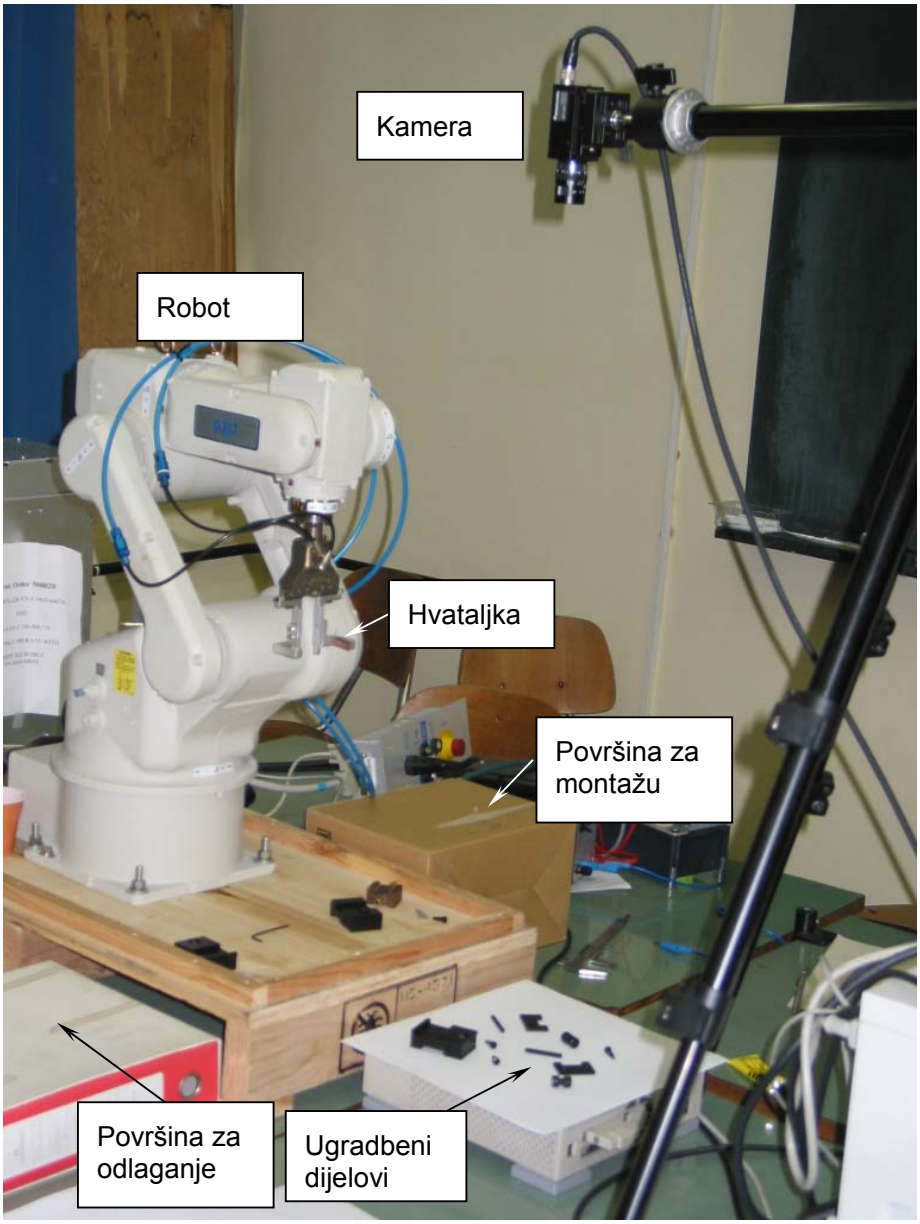
Drugi zapisnik, *put_sklap.txt,* opisuje putanje sklapanja, a njegov sadržaj je:

```
6               3               357.922         0               26
bazni           362.986         -88.538         340             0
3               364.574         0               357.922         180
179.473         40              180             -88.538         90
344.521         180             0               0               vijak
-15.358         90              252             180             3
0               0               357.922         0               350.234
180             362.986         -88.538         plocica         335.309
0               364.574         0               3               -10
179.473         -220.4          180             208.399         179.9
344.521         180             0               323.862         97.937
-96.132         90              268             26              0.1
0               0               357.922         0               350.234
180             362.986         -88.538         180             335.309
0               364.574         0               90              -204.486
179.473         40              180             208.399         179.9
344.521         180             0               323.862         97.937
-15.358         90              233.645         -81.1           0.1
0               0               357.922         0               350.234
180             zatik           -88.538         180
0               5               0               90
valjak          342.055         180             208.399
```

9.3. Postupak izvođenja montažnih operacija

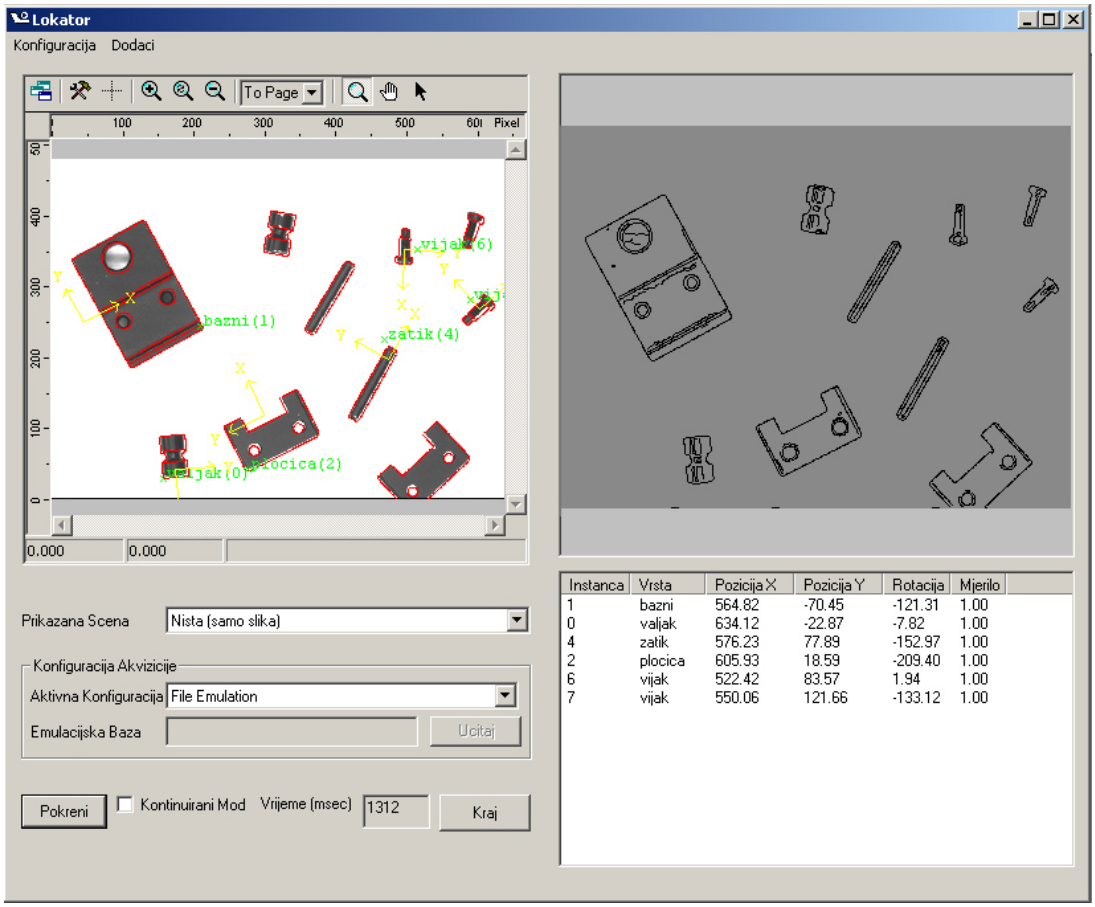
Sustav kojim će se upravljati razvijenom programskom podrškom prikazan je slikom Sl.-77. Unutar radnog područja robota nalaze se tri radne površine.

Jedna je pokrivena vizijskim sustavom i na njoj se nalaze ugradbeni dijelovi u nesređenom staju. Robot uzima ugradbene dijelove s te površine i prenosi ih do druge radne površine, odnosno mjesta gdje se izvodi sklapanje. Nakon što je sklopljen i posljednji dio, proizvod se odlaže na površinu za odlaganje.



Sl.-77. Sustav za automatsko sklapanje u nesređenoj okolini.

Izvođenje montažnog postupka započinje procesom prepoznavanja koji se ostvaruje izvođenjem razvijene programske podrške strojnog vida. Izgled sučelja programske podrške strojnog vida s rezultatima prepoznavanja prikazan je slikom Sl.-78.

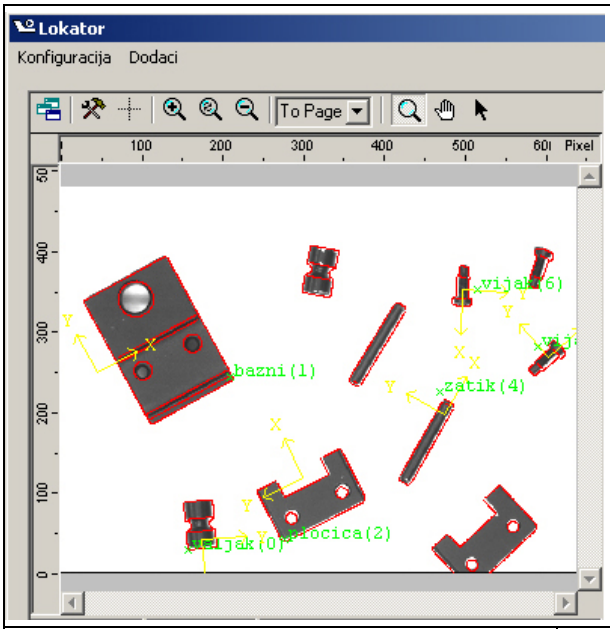
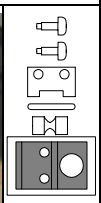
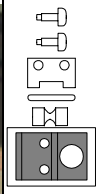


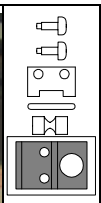
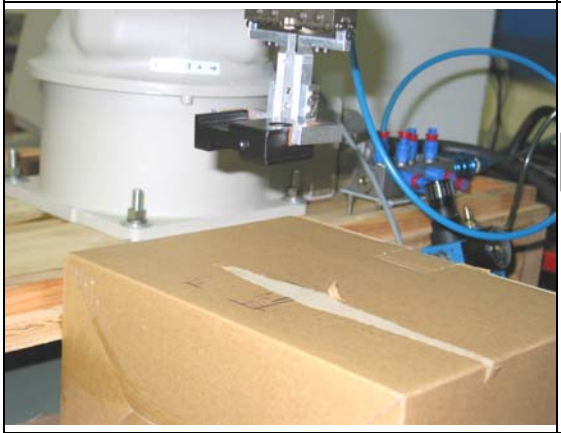
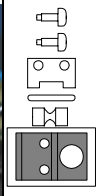
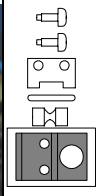
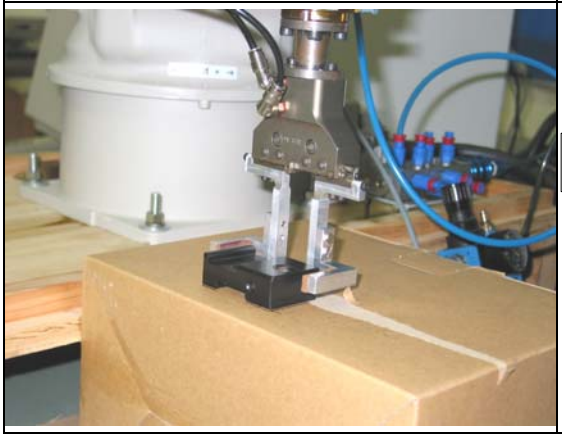
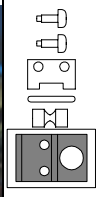
Sl.-78. Sučelje programske podrške strojnog vida sa rezultatima.

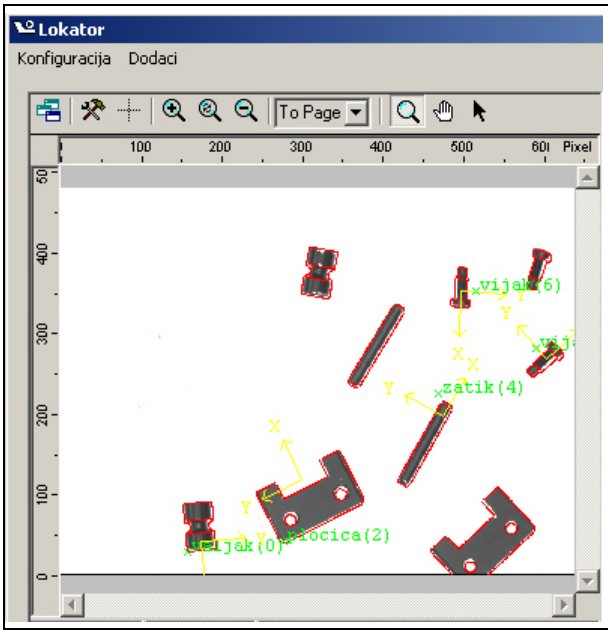
Upravljački program robota uzima rezultate prepoznavanja i lociranja ugradbenih elemenata. Tablica 3 ilustrira jedan ciklus sklapanja sklopa za primjenu i verifikaciju sustava strojnog vida.

U prvom stupcu tablice su fotografije montažnih operacija. Osim toga, prikazan je i integrirani prizor objekta na svakom početku sklapanja novog ugradbenog dijela. Drugi stupac pokazuje koji ugradbeni dio se sklapa. U trećem je stupcu ukratko opisana slika.

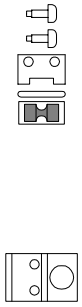
Tablica 3. Prikaz montažnog postupka sklopa za primjenu i verifikaciju sustava strojnog vida.

	*Integrirani prizor objekta na početku sklapanja baznog dijela.*
 	*Robot je pozicioniran direktno iznad točke hvatanja baznog dijela.*
 	*Položaj robota prilikom hvatanja baznog dijela.*

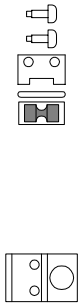
		*Nakon hvatanja baznog dijela robot se udaljava od radne površine s dijelovima u nesređenom stanju.*
		*Robot je pozicioniran direktno iznad mjesta gdje će postaviti bazni dio.*
		*Položaj robota prilikom postavljanja baznog dijela.*
		*Otvaranje hvataljke prilikom postavljanja baznog dijela na površinu za sklapanje.*



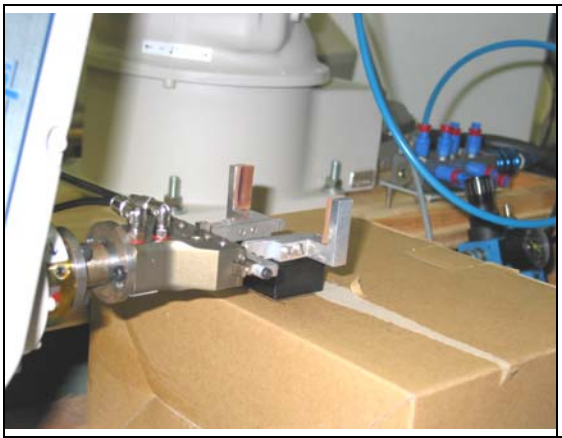
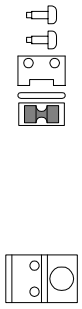
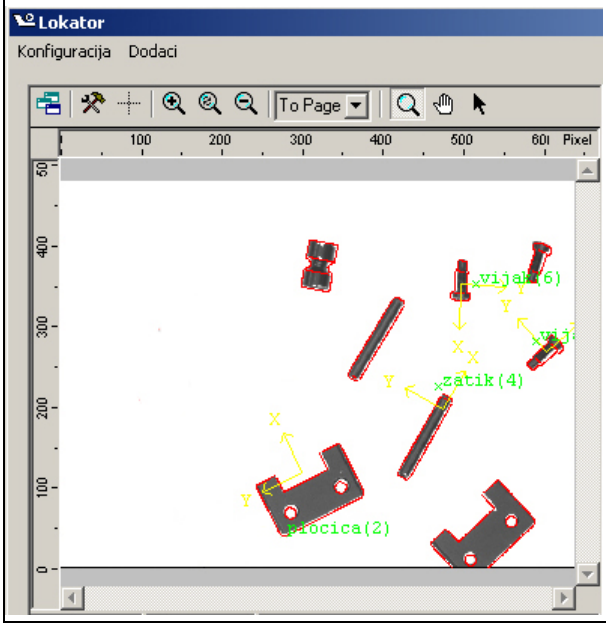
Integrirani prizor objekta nakon sklapanja baznog dijela, a prije početka sklapanja valjka.

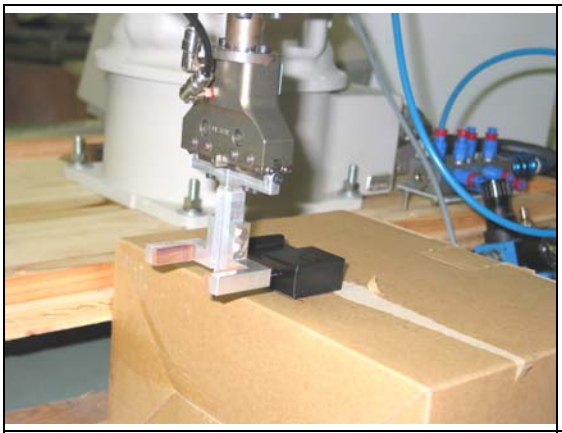
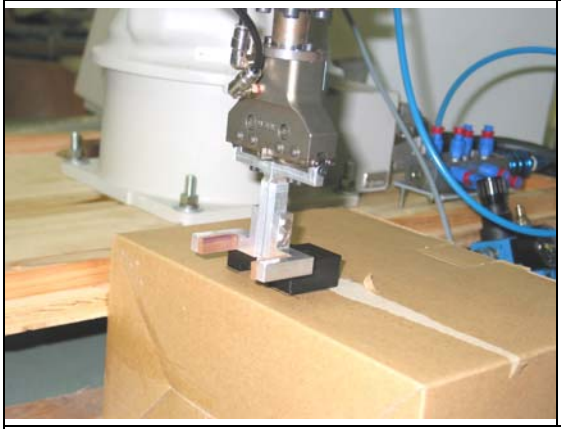
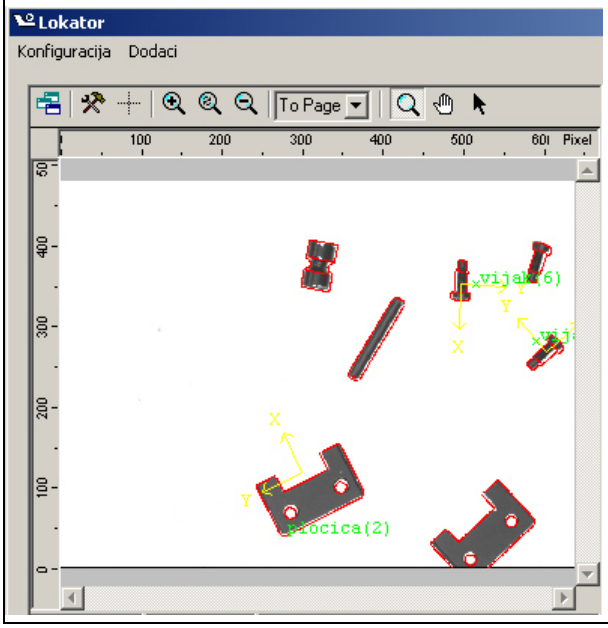


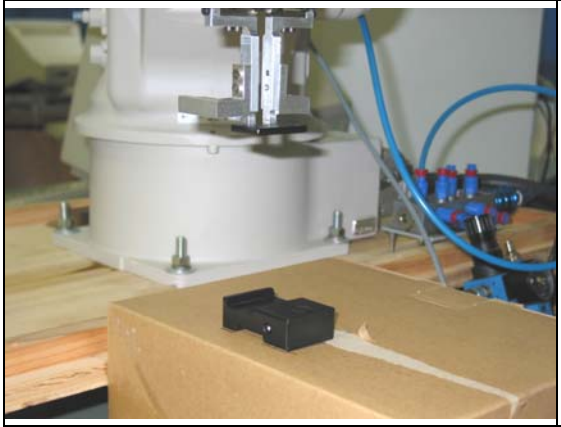
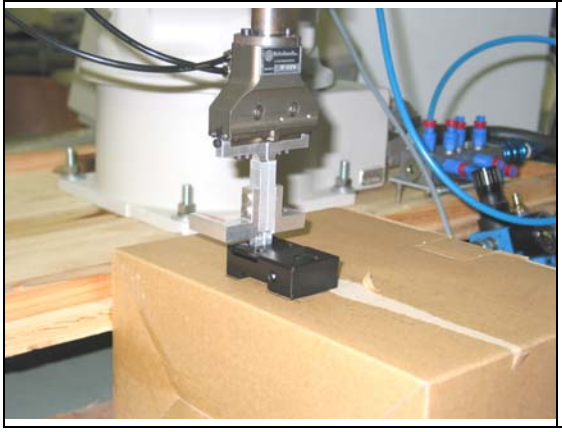
Robot je pozicioniran direktno iznad točke hvatanja drugog ugradbenog dijela po redoslijedu sklapanja – valjka.

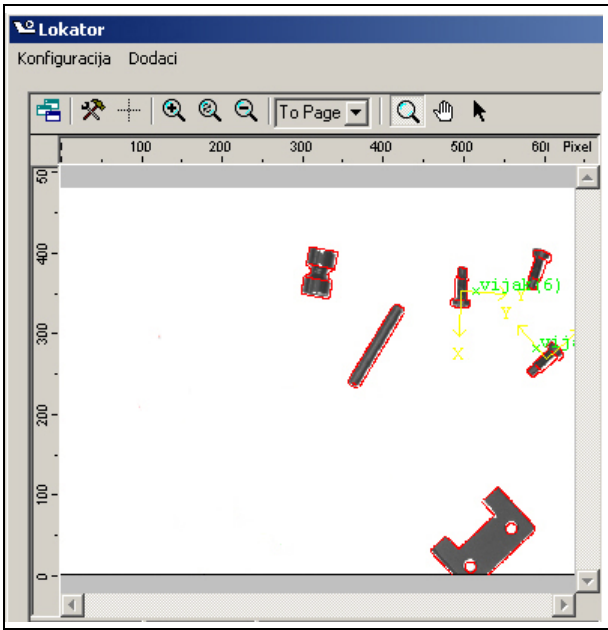


Položaj robota prilikom hvatanja valjka.

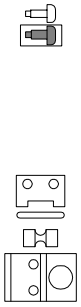
		*Položaj robota na kraju ugradnje valjka u sklop.*
		*Integrirani prizor objekta nakon sklapanja valjka, a prije početka sklapanja zatika.*
		*Položaj robota prilikom hvatanja zatika.*

		*Umetanje zatika.*
		*Položaj robota na kraju ugradnje zatika u sklop.*
		*Integrirani prizor objekta nakon sklapanja zatika, a prije početka sklapanja pločice.*

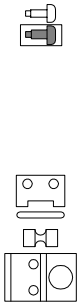
		*Položaj robota pri hvatanju pločice.*
		*Robot je pozicioniran na početku završnog (pravocrtnog) dijela operacije ugradnje pločice.*
		*Položaj robota pri ugradnji pločice u sklop.*



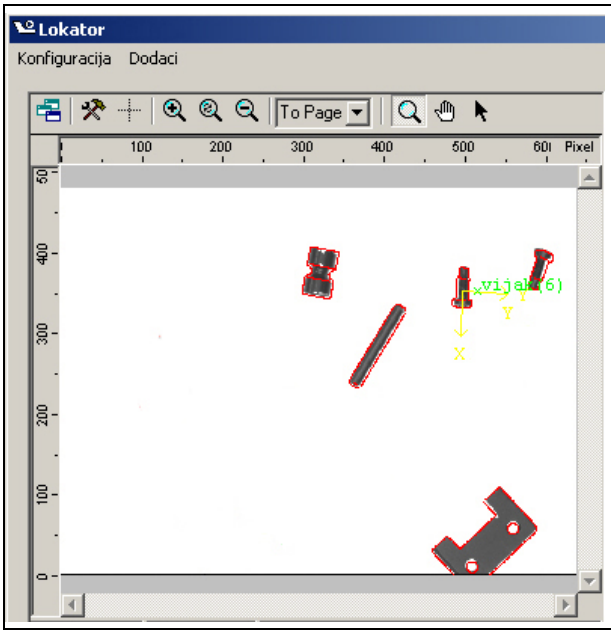
Integrirani prizor objekta nakon sklapanja pločice, a prije početka sklapanja prvog vijka.



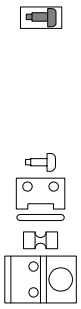
Položaj robota pri hvatanju prvog vijka.



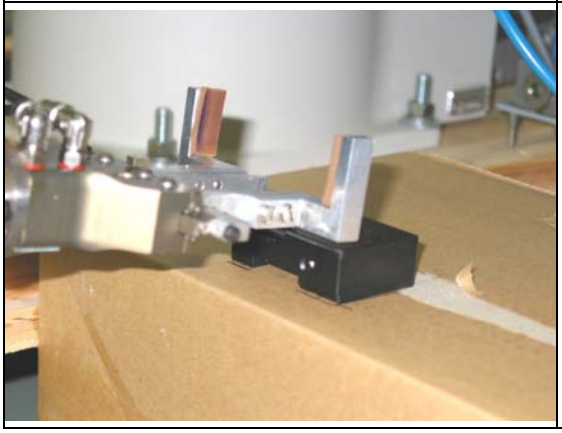
Položaj robota pri ugradnji prvog vijka u sklop.

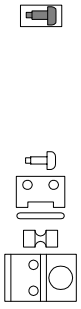


Integrirani prizor objekta nakon sklapanja prvog vijka, a prije početka sklapanja drugog vijka, posljednjeg dijela prema planu sklapanja.

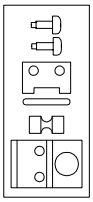
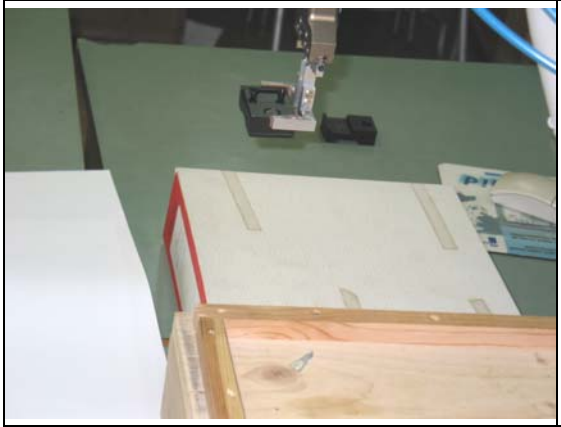
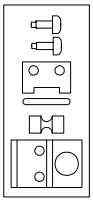
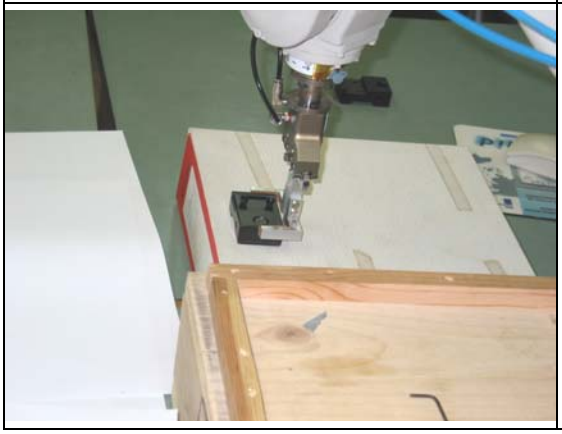
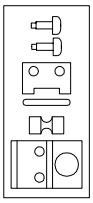


Položaj robota kod hvatanja drugog vijka.





I njegovo postavljanje u sklop.

		*Hvatanje sklopljenog proizvoda.*
		*Prenošenje sklopa do površine za odlaganje.*
		*Odlaganje sklopljenog proizvoda.*

10 ZAKLJUČAK

Kod automatske/robotske montaže informacije o pozicijama i orijentacijama ugradbenih dijelova u radnoj okolini predstavljaju esencijalne podatke za upravljanje sustavom. Direktno utječu na učinak operacija kod kojih dolazi do promjene položaja ugradbenog dijela.

Upravljanje prostornim položajima ugradbenih dijelova, odnosno rukovanje, je jedan od najtežih zadataka kod automatske montaže. Iz toga proizlazi sljedeći zaključak: ako se mogu operacije rukovanja realizirati pomoću automatske opreme, čitav proces montaže može biti automatiziran.

Problem rukovanja kod automatskih/robotskih montažnih sustava se tradicionalno rješava tako da se pretpostavi poznato stanje ugradbenih elemenata. To znači da je radni prostor robota i stanje pomoćne opreme determinirano. Nedostatak ovakvog pristupa jest da mala odstupanja od očekivanih položaja dijelova ili male promjene njihove geometrije uzrokuju znatne poteškoće u montažnom procesu. Općenito, usprkos digitalnom determinizmu automatskih sustava, neizbježan utjecaj determinističkog kaosa proizvodi nepredvidive uvjete te se mora prihvatiti kao prirodni fenomen, nasuprot aktualnim naporima kojima se želi stvoriti (tehnički) savršen svijet [51]. Stoga je ključan pristup pri automatizaciji montaže razvoj autonomnih robotskih montažnih sustava, sposobnih da uspješno djeluju u nesređenoj okolini.

U radu je razvijen model autonomnog fleksibilnog montažnog sustava (Sl.-37). Osnovna ideja koncepta je maksimalno korištenje znanja o proizvodu strukturiranog, konceptualiziranog i operacionaliziranog u vidu CAD modela, te integracija procesa oko njega. CAD model sadrži opis geometrije proizvoda, iz kojeg se izvode parametri proizvodnog i montažnog procesa. Ostala znanja se moraju prikupiti pomoću senzora.

Sustav strojnog vida je jedan od fundamentalnih preduvjeta u ovom konceptu. Stoga je, u duhu predloženog koncepta, realiziran autonomni robotski montažni sustav sposoban za djelovanje u nesređenim radnim uvjetima. Sustav se sastoji od industrijskog robota i sustava strojnog vida s odgovarajućom programskom podrškom. Programska podrška razvijenog sustava uključuje programsku podršku strojnog vida i upravljački program robota.

Programskom podrškom strojnog vida prepoznaju se ugradbeni elementi koji se trebaju sklopiti prema definiranom planu montaže. Ugradbeni elementi su u radnom prostoru robota, a njihove pozicije i orijentacije nisu determinirane fizičkim ograničavanjem pomoću skupih nefleksibilnih mehaničkih perifernih montažnih uređaja. Sustav strojnog vida utvrđuje položaje ugradbenih elemenata u sustavu, čime njihove pozicije postaju određene, a sustav informacijski sređen.

Utvrđivanje položaja ugradbenih elemenata slijedi nakon postupka prepoznavanja ugradbenih elemenata. Proces prepoznavanja je baziran na procesiranju i usporedbi dva izvora informacija: CAD modela proizvoda i slike sa kamere.

CAD informacija je 3D matematička/analitička reprezentacija fizičkog proizvoda koja se mora transformirati u oblik pogodan za sustav strojnog vida. Za razliku od CAD-a iz kojeg se kreiraju slike virtualnih ugradbenih dijelova, kamerom se kreiraju slike stvarnih ugradbenih dijelova. Slike sa oba izvora moraju biti kalibrirane i formatirane za esencijalni dio programske podrške strojnog vida, program za identificiranje i lociranje.

Razvijena programska podrška je utemeljena na komercijalnim *Adept HEXSIGHT* vizijskim alatima. U programsku podršku su uključeni alati/programi za:

- identificiranje i lociranje
- upravljanje podacima
- kreiranje modela za prepoznavanje
- vizualizaciju rješenja.

Upravljački program robota integrira rezultate sustava strojnog vida i informacije o procesu sklapanja. Time su potpuno određena gibanja robotske prihvatnice te se može izvršiti montaža.

Razvojem i verifikacijom ovog sustava pokazalo se da je moguće učinkovito realizirati autonomno robotsko sklapanje u nesređenoj okolini. Sustav strojnog vida, na koji je posebno fokusirano ovo istraživanje, u velikoj mjeri povećava fleksibilnost montažnog sustava budući da se sada mogu izbjeći periferni montažni uređaji za orijentiranje koji najviše doprinose krutosti sustava.

U razvijenom sustavu direktno su korištene informacije o virtualnom (CAD) proizvodu za prepoznavanje objekata. Informacije kojima su definirani parametri montažnog procesa u skladu su s rezultatima sustava za generiranje plana sklapanja. Plan sklapanja je kreiran na temelju CAD modela sklopa, pa su stoga i parametri montažnog procesa u razvijenom sustavu indirektno definirani CAD modelom.

Time je potvrđeno da je moguće informacijski integrirati razvoj, odnosno oblikovanje proizvoda, projektiranje procesa sklapanja i izvođenje montaže. To je, u duhu koncepta simultanog inženjerstva, i još jedna potvrda da je moguće realizirati fleksibilnu montažu koristeći inženjerska znanja strukturirana, konceptualizirana i operacionalizirana u prethodnim fazama razvoja proizvoda.

Daljnja istraživanja nameću se u tri smjera. Prvi je unapređivanje vizijskog sustava. To se u prvom redu odnosi na implementaciju prepoznavanja i određivanja položaja na temelju 3D modela za prepoznavanje. Prepoznavanje na temelju 3D modela omogućuje prepoznavanje objekata kada se ovi nalaze u bilo kojoj orijentaciji. Model sustava na temelju kojeg se zaključuje o stanju sustava je sličniji realnom sustavu te je ukupna vizualna percepcija cjelovitija. Nadalje, pretpostavlja se da bi 3D prepoznavanje omogućilo vizualnu percepciju konfiguracije montažnih uređaja, odnosno stanja čitavog montažnog sustava. Ovakav pristup prepoznavanju bi opravdao ili čak nametnuo postavljanje kamere na ruku robota.

Drugi pravac istraživanja se odnosi na daljnju integraciju razvijenog sustava s alatima pomoću kojih se realiziraju ostale faze razvoja proizvoda, odnosno povećanje razine automatizma procesa razmjene informacija između ovih sustava.

Na kraju, treći se pravac daljnjih istraživanja odnosi na pokušaj implementacije inteligentnih upravljačkih programa spregnutih sa sustavom za vizualnu percepciju. Na taj se način proširuje mogućnost automatizacije montaže i zamjene ljudskog rada u takvim i sličnim djelatnostima.

11 LITERATURA

[1] B. Vranješ, B. Jerbić, Z. Kunica;"*Inženjerski priručnik*"; u tisku.
[2] B. Lotter; "*Wirtshaftliche Montage*"; VDI Verlag; Düsseldorf; 1986.
[3] Z. Kunica; "*Doprinos generiranju planova automatske montaže - Doktorska disertacija*"; Sveučilište u Zagrebu; Zagreb; 1996.
[4] DIN 8593; "*Fertigungsverfahren Fuegen; Einordnung, Unterteilung, Begriffe; T0-T8*"; 1985.
[5] B. Vranješ, B. Jerbić, Z. Kunica; "*Doprinos projektiranju automatskih montažnih sistema*"; Strojarstvo; Vol. 33, No. (2-3), pp. 97-110; 2002.
[6] T. Owen; "*Assembly with robots*"; Prentece-Hall, Inc.; Englewoog Cliffs; New Jersey; 1985.
[7] P. Dewhurst, W. Knight, G. Boothroyd; "*Product Design for Manufacture & Assembly Revised & Expanded*"; Marcel Dekker; New York; 2001.
[8] C. R. Asfahl; "*Robots and Manufacturing Automation*"; John Wiley & Sons; New York; 1985.
[9] A. C. McDonald; "*Robot Technology; Theory, Design and Aplications*"; Prentice-Hall, Inc.; Englewood Clifs; New Jersy; 1986.
[10] B. Batchelor; "*Intelligent Sensing Systems for Industry*", http://bruce.cs.cf.ac.uk/bruce/#anchor3048002.
[11] M.L. Minsky; "*Steps Toward Artificial Intelligence*"; u: J. Feldmann, E.A. Feigenbaum (Eds.); "*Computers and Thought*"; pp. 406-450; McGraw-Hill; New York; 1963.
[12] R. Jain, R. Kasturi, B. G. Schunk; "*Machine Vision*"; McGraw-Hill, Inc.; New York; 1995.
[13] Computer User High-Tech Dictionary; http://www.computeruser.com/resources/dictionary/.
[14] Image Labs International; "*Machine Vision and Imaging Library*"; http://www.vision1.com/gl.shtml.
[15] R.B. Fisher; "*From Surfaces to Objects. Computer Vision and Three Dimensional Scene Analysis*"; John Wiley; Chichester; 1989.

[16] D. M. Szaflarski; "*How We See: The First Steps of Human Vision*"; http://www.accessexcellence.com/AE/AEC/CC/vision_background.html.
[17] Institute for Innovative Blind Navigation; "*Vision, Navigation, Mobility and the Brain*"; http://www.saginaw.k12.mi.us/~mobility/visbrain.htm.
[18] N. Zuech; "*Understanding and Applying Machine Vision*"; M. Dekker; New York; 2000.
[19] L.G. Roberts; "*Machine perception of three-dimensional solids*"; u J.P. Tippet et. al. (Eds.); "*Optical and electro-optical information processing*"; MIT Press; Cambridge; 1965.
[20] National Instruments; "*Anatomy of camera*"; http://zone.ni.com/.
[21] Linos Photonics; "*CCD-Camera Technology*"; http://www.spindlerhoyer.co.uk/pdf/en/prod/x1-e-06.pdf.
[22] G.Z. Yang, D.F. Gillies; "*Region Based Segmentation*"; http://www.doc.ic.ac.uk/~gzy
[23] M. Jiang; "*Digital Image Processing*"; http://www.math.pku.edu.cn/teachers/jiangm/%CD%BC%CF%F3%B4%A6%C0%ED%BF%CE%B3%CC%BD%B2%D2%E5/html/dip.html.
[24] D. Eppstein; "*Geometry in action, Quadtrees and Hierarchical Space Decomposition*"; http://www1.ics.uci.edu/~eppstein/gina/quadtree.html.
[25] D. Marr, H. K. Nishihara; "*Representation and recognition of the spatial organization of threedimensional shapes*"; Proceedings of the Royal Society in London, B 200; pp. 269-294; 1977.
[26] J. F. Reid; "*Applied Machine Vision*"; http://www.age.uiuc.edu/age315/inf315.html.
[27] Association for Computing Machinery, Multimedia Research group; http://www.acm.org/sigmm/.
[28] R.M. Haralick; "*Digital step edges from zero crossing of second directional derivates*"; IEEE Trans. Patern Analysis and Machine Intelligence; 6(1):58-68; 1984.
[29] R.M. Haralick, L.G. Shapiro; "*Computer and Robot Vision*"; Adison-Wesley; Reading; 1992.
[30] R. Fisher (ed); "*The Marble Project*"; http://www.icbl.hw.ac.uk/marble/vision/.
[31] M. Sonka, V. Hlavac, R. Boyle; "*Image Processing, Analysis, and Machine Vision*"; Brooks and Cole Publishing; 1998.
[32] J. M. Buhmann, J. Malik, P. Perona; "*Image Recognition: Visual Grouping, Recognition and Learning*"; Proc. of National Academy of Sciences; 96(25), 14203-14204; December 1999.
[33] B. Jerbić; "*Računalom podržano inženjerstvo*"; u tisku.
[34] Z. Kunica, B. Vranješ; "*Towards automatic generation of plans for automatic assembly*"; International Journal of Production Research; Vol. 37, No. 8, pp. 1817-1836; 1999.
[35] S. Chakrabarty, J. Wolter; "*A Structure-Oriented Approach to Assembly Sequence Planning*"; IEEE Transactions on robotics and automation; Vol. 13, No. 1, pp. 14-29; 1997.
[36] R.E. Jones, R. H. Wilson, T. L. Calton; "*On Constraints in Assembly Planning*"; IEEE Transactions on Robotics and Automation; Vol. 14, No. 6, pp. 849-863; 1998.
[37] T. Šurina, M. Crneković; "*Industrijski roboti*"; Školska knjiga Zagreb; Zagreb; 1990.
[38] TeachTargetVB.com; "*ActiveX control*"; http://searchvb.techtarget.com/.
[39] VayTek Inc.; "*Cameras*"; http://www.vaytek.com/cameras1.htm.
[40] Matrox Imaging; "*Matrox Meteor-II/Multi-Channel*"; http://www.matrox.com/.
[41] PC Guide; "*Peripheral Component Interconnect (PCI) Local Bus*"; http://www.pcguide.com/ref/mbsys/buses/types/pci.htm.
[42] Adept Technology Inc.; "*Adept MV Controler User's Guide*"; Adept Technology; San Jose;. 2001.
[43] H. Gilbert; "*Introduction to TCP/IP*"; http://www.yale.edu/pclt/COMM/TCPIP.HTM.
[44] 1394 Trade Association; "*1394 TA Specifications*"; http://www.1394ta.org/Technology/Specifications/specifications.htm.

[45] Adept Technology Inc.; "*V+ Operating System User's Guide*"; Adept Technology; San Jose;. 2001.
[46] Panasonic; "*GP-MF602 - 1/2" CCD Machine Vision Black and White Industrial Camera*"; http://www.panasonic.com/medical_industrial/.
[47] Adept Technology Inc.; "*HexSight User Guide*"; Adept Technology Canada; Santa Foy; 2001.
[48] Dassault Systemes; "*The CATIA Users' Page*"; http://www.catia.com/.
[49] Adobe Systems Incorporated; http://www.adobe.com/.
[50] Adept Technology Inc.; "*V+ Language User's Guide, Version 14.0*"; Adept Technology; San Jose; 2001.
[51] B. Jerbic; "*Autonomous robotic assembly using collaborative behavior based agents*"; International journal of smart engineering system design; Vol. 4, No. 1, pp. 11-20, Jan-Mar, 2002.

KRATKI ŽIVOTOPIS

Miljenko Hrman rođen je 16. siječnja 1975. godine u Varaždinu gdje je završio Srednju strojarsku školu. Fakultet strojarstva i brodogradnje, Sveučilišta u Zagrebu upisao je 1993. godine. Diplomirao je 1998. na smjeru Proizvodno strojarstvo.

Od 1998. godine zaposlen je pri Katedri za projektiranje izradbenih i montažnih sustava pri Zavodu za robotiku i automatizaciju proizvodnih sustava Fakulteta strojarstva i brodogradnje u Zagrebu u svojstvu znanstvenog novaka. Aktivno sudjeluje u znanstveno-istraživačkom radu te je do sada objavio 9 znanstvenih radova te sudjelovao na brojnim međunarodnim znanstvenim konferencijama. Osim u istraživačkom radu, sudjeluje u

izvođenju nastave iz slijedećih kolegija: Primjena računala-P, Računalom podržano inženjerstvo, Projektiranje proizvodnih sustava, Montaža i zaštita, Automatizacija montaže i Automatska montaža.

SHORT BIOGRAPHY

Miljenko Hrman was born on January 16, 1975 in Varaždin, where he also attended Secondary School of Mechanical Engineering. He enrolled at the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb in 1993 and graduated in 1998 majoring in Production Engineering.

In 1998 he started working as a junior researcher at the Chair of Manufacturing and Assembly System Planning of the Department of Robotics and Automation of Production Systems at the Faculty of Mechanical Engineering and Naval Architecture. He has been actively involved in the research and has published nine scientific papers and took part in numerous international conferences. In addition to the research, he has also been involved in teaching the following courses: Computer Applications, Computer Aided Engineering, Production Systems Design, Assembly and Safety, Assembly Automation, and Automatic Assembly.

PRILOG

P.1. Kod programske podrške za upravljanje strojnim vidom.

P.1.1. PartSort.hpp

```
#ifndef
AFX___INCLUDEPARTSORT_HPP__09464807_296C_11D3_A0E0_0050046EF497__INCLUDED_
#define
AFX___INCLUDEPARTSORT_HPP__09464807_296C_11D3_A0E0_0050046EF497__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"
#include "KordDijl.h"


class CPartSortApp : public CWinApp
{
public:

	CPartSortApp( void );

	//{{AFX_VIRTUAL(CPartSortApp)
	public:
	virtual BOOL InitInstance( void );


	DECLARE_MESSAGE_MAP()
```

```
};

// * Inline functions definitions

//{{AFX_INSERT_LOCATION}}

#endif //
//AFX___INCLUDEPARTSORT_HPP__09464807_296C_11D3_A0E0_0050046EF497__IN
//CLUDED_
```

P.1.2. PartSort.cpp

```
#include "stdafx.h"
#include "PartSort.hpp"
#include "PartSortDlg.hpp"

CPartSortApp theApp;
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////
// * Maps definitions
// //////////////////////////

BEGIN_MESSAGE_MAP(CPartSortApp, CWinApp)
       //{{AFX_MSG_MAP(CPartSortApp)
       //}}AFX_MSG
       ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

// ////////////////////////
// * Functions definitions
// //////////////////////////

// //////////////////////////
// Function....: CPartSortApp
//
// Description.: Contructor for the CPartSortApp object
// ****

CPartSortApp::CPartSortApp( void )
{
       // TODO: add construction code here,
       // Place all significant initialization in InitInstance
}

// /////////////////////////////////////////////////
// Function....: InitInstance
//
// Return......: BOOL (True if initialisation succeed)
// ////////////////////////////////////////////////////////////////////
BOOL
CPartSortApp::InitInstance( void )
{
       AfxEnableControlContainer();

#ifdef _AFXDLL
       Enable3dControls();              // Call this when using MFC //in a
       shared DLL
#else
       Enable3dControlsStatic();  // Call this when linking to MFC //statically
#endif

       CPartSortDlg lDlg;
       m_pMainWnd = &lDlg;
       int lResponse = lDlg.DoModal();

       if ( lResponse == IDOK )
       {
              // TODO: Place code here to handle when the dialog is
              //            dismissed with OK
       }
       else if ( lResponse == IDCANCEL )
       {
              // TODO: Place code here to handle when the dialog is
              //            dismissed with Cancel
       }

       return ( FALSE );
}
```

P.1.3. PartSortDlg.hpp

```
// ************************************************************************
// PartSort
//
************************************************************************
// Platforms....: WIN32
*****************************************************************************
//{{AFX_INCLUDES()
#include "HSApplication.h"
#include "HSDisplay.h"
#include "HSListCtrl.hpp"
//}}AFX_INCLUDES

#ifndef
AFX___INCLUDEPARTSORTDLG_HPP__09464809_296C_11D3_A0E0_0050046EF497__INCLUDED_
#define
AFX___INCLUDEPARTSORTDLG_HPP__09464809_296C_11D3_A0E0_0050046EF497__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif

class HSAcquisitionDevice;
class HSLocator;
class HSModel;

*****************************************************************************

class CPartSortDlg : public CDialog
{
public:

                    CPartSortDlg( CWnd * pParent = NULL );

protected:

      HICON mIcon;

public:

      //{{AFX_DATA(CPartSortDlg)
      enum { IDD = IDD_PARTSORT_DIALOG };
      HSApplication mApplicationControl;
      HSDisplay           mApplicationDisplay;
      HSDisplay           mApplicationDisplay2;
      HSListCtrl          mMeasurementGrid;
      CComboBox           mSceneCombo;
      CComboBox           mConfigurationCombo;
      CButton                   mSortButton;
      CButton                   mLoadButton;
      CButton                   mContinuousCheck;
      CEdit               mEmulationDatabase;
      CEdit               mElapsedTime;
      //}}AFX_DATA

      //{{AFX_VIRTUAL(CPartSortDlg)
      protected:
      virtual void DoDataExchange( CDataExchange * pDX );  // DDX/DDV support
      //}}AFX_VIRTUAL

protected:

      //{{AFX_MSG(CPartSortDlg)
      virtual BOOL OnInitDialog( void );
      afx_msg void OnPaint( void );
      afx_msg HCURSOR OnQueryDragIcon( void );
      afx_msg void OnButtonQuit( void );
      afx_msg void OnButtonLoad( void );
      afx_msg void OnButtonSort( void );
      afx_msg void OnSelendokComboScene();
      afx_msg void OnSelendokComboConfiguration();
      afx_msg void OnDropdownComboConfiguration( void );
      afx_msg void OnPropertiesChangeControlApplication(LPCTSTR ProcessName);
      afx_msg void OnConfigurationExploreHexSightApplication( void );
```

```
	afx_msg void OnConfigurationAcquisitionProperties( void );
	afx_msg void OnConfigurationModelProperties( void );
	afx_msg void OnConfigurationLocatorProperties( void );
	afx_msg void OnConfigurationConfigurationFileLoad( void );
	afx_msg void OnConfigurationConfigurationFileSave( void );
	afx_msg void OnConfigurationCameraCalibrationNew( void );
	afx_msg void OnConfigurationCameraCalibrationLoad( void );
	afx_msg void OnConfigurationCameraCalibrationSave( void );
	afx_msg void OnWindowRealTimeDisplay( void );
	afx_msg void OnWindowLocatorViewer( void );
	afx_msg void OnClose();
	afx_msg void OnDestroy();
	afx_msg void OnDijalogKordinat( void );
	DECLARE_EVENTSINK_MAP()
	//}}AFX_MSG

	DECLARE_MESSAGE_MAP()

private:

	bool   mFillingConfigurationCombo;

	void UpdateElapsedTime( DWORD const pStart, DWORD const pStop );
	void FillConfigurationCombo( void );
	void WaitForCompletion( void );
	void SetAcquisitionConfiguration( void );
};

// ****************************************************************************
// * Inline functions definitions
// ****************************************************************************

//{{AFX_INSERT_LOCATION}}
//MDS will insert additional declarations immediately before the previous line.

#endif //
AFX___INCLUDEPARTSORTDLG_HPP__09464809_296C_11D3_A0E0_0050046EF497__INCLUDED_
```

P.1.4. PartSortDlg.cpp

```
#include "stdafx.h"
#include "KordFloat.h"
#include "PartSort.hpp"
#include "PartSortDlg.hpp"
#include "Common.hpp"
#include "HSAcquisitionDevice.h"
#include "HSDatabase.h"
#include "HSLocator.h"
#include "HSProcessManager.h"
#include <fstream.h>
#include "kut.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// * Maps definitions
// ******************************************************

       CKordFloat ishodslike;
       CKordFloat tocapscisa;
       CKordFloat tocka;
       CKordFloat tockarobot;
       float kut = 0;
       float rotacija;
       const double pi = 3.14159265358979323;


BEGIN_MESSAGE_MAP(CPartSortDlg, CDialog)
//{{AFX_MSG_MAP(CPartSortDlg)
ON_WM_PAINT()
ON_WM_QUERYDRAGICON()
ON_BN_CLICKED(ID_BUTTON_QUIT, OnButtonQuit)
ON_BN_CLICKED(IDC_BUTTON_LOAD, OnButtonLoad)
ON_BN_CLICKED(IDC_BUTTON_SORT, OnButtonSort)
ON_CBN_SELENDOK(IDC_COMBO_SCENE, OnSelendokComboScene)
ON_CBN_SELENDOK(IDC_COMBO_CONFIGURATION, OnSelendokComboConfiguration)
ON_CBN_DROPDOWN(IDC_COMBO_CONFIGURATION, OnDropdownComboConfiguration)
ON_COMMAND(ID_CONFIGURATION_EXPLOREHEXSIGHTAPPLICATION,
OnConfigurationExploreHexSightApplication)
ON_COMMAND(ID_CONFIGURATION_ACQUISITIONPROPERTIES,
OnConfigurationAcquisitionProperties)
ON_COMMAND(ID_CONFIGURATION_LOCATORPROPERTIES, OnConfigurationLocatorProperties)
ON_COMMAND(ID_CONFIGURATION_CONFIGURATIONFILE_LOAD,
OnConfigurationConfigurationFileLoad)
ON_COMMAND(ID_CONFIGURATION_CONFIGURATIONFILE_SAVE,
OnConfigurationConfigurationFileSave)
ON_COMMAND(ID_CONFIGURATION_CAMERACALIBRATION_NEW,
OnConfigurationCameraCalibrationNew)
ON_COMMAND(ID_CONFIGURATION_CAMERACALIBRATION_LOAD,
OnConfigurationCameraCalibrationLoad)
ON_COMMAND(ID_CONFIGURATION_CAMERACALIBRATION_SAVE,
OnConfigurationCameraCalibrationSave)
ON_COMMAND(ID_WINDOW_REALTIMEDISPLAY, OnWindowRealTimeDisplay)
ON_COMMAND(ID_WINDOW_LOCATORVIEWER, OnWindowLocatorViewer)
ON_COMMAND(IDD_POK_DIJ, OnDijalogKordinat)
ON_WM_CLOSE()
ON_WM_DESTROY()
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_EVENTSINK_MAP(CPartSortDlg, CDialog)
       //{{AFX_EVENTSINK_MAP(CPartSortDlg)
       ON_EVENT(CPartSortDlg, IDC_CONTROL_APPLICATION, 4 /* PropertiesChange */,
OnPropertiesChangeControlApplication, VTS_BSTR)
       //}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

CPartSortDlg::CPartSortDlg( CWnd * pParent /* = NULL */ )
       : CDialog( CPartSortDlg::IDD, pParent ),
         mFillingConfigurationCombo( false )
```

```
{
	mIcon = AfxGetApp()->LoadIcon( IDR_MAINFRAME );
}

void
CPartSortDlg::DoDataExchange( CDataExchange * pDX )
{
CDialog::DoDataExchange( pDX );

//{{AFX_DATA_MAP(CPartSortDlg)
	DDX_Control(pDX, IDC_LIST_INSTANCE, mMeasurementGrid);
	DDX_Control(pDX, IDC_CONTROL_APPLICATION, mApplicationControl);
	DDX_Control(pDX, IDC_CONTROL_DISPLAY, mApplicationDisplay);
	DDX_Control(pDX, IDC_CONTROL_DISPLAY2, mApplicationDisplay2);
	DDX_Control(pDX, IDC_COMBO_SCENE, mSceneCombo);
	DDX_Control(pDX, IDC_COMBO_CONFIGURATION, mConfigurationCombo);
	DDX_Control(pDX, IDC_BUTTON_SORT, mSortButton);
	DDX_Control(pDX, IDC_BUTTON_LOAD, mLoadButton);
	DDX_Control(pDX, IDC_EDIT_EMULATION_DATABASE, mEmulationDatabase);
	DDX_Control(pDX, IDC_EDIT_TIME, mElapsedTime);
	DDX_Control(pDX, IDC_CHECK_CONTINUOUS, mContinuousCheck);
	//}}AFX_DATA_MAP
}

BOOL
CPartSortDlg::OnInitDialog( void )
{
	CDialog::OnInitDialog();

	SetIcon(mIcon, TRUE);		// Set big icon
	SetIcon(mIcon, FALSE);		// Set small icon


	HSAcquisitionDevice lAcquisition =
	mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );
	HSLocator lLocator = mApplicationControl.GetProcessManager().GetProcess(
COleVariant( "Locator" ) );
	HSLocator lPLocator2 =
mApplicationControl.GetProcessManager().GetProcess( COleVariant( "Locator" ) );

	mApplicationDisplay.SetImageDatabase( 0,
mApplicationControl.GetDatabase().GetHandle() );

	mApplicationDisplay.SetSceneDatabase( 0,
mApplicationControl.GetDatabase().GetHandle() );




	if ( lAcquisition.IsValid() )
	{
		mApplicationDisplay.SetImageViewName( 0,
lAcquisition.GetOutputView( HEXSIGHT_NONE ) );
		mApplicationDisplay.SetImageName( 0,
lAcquisition.GetOutputGreyScaleImage( HEXSIGHT_NONE ) );

	}

	if ( lLocator.IsValid() )
	{
		mApplicationDisplay.SetSceneViewName( 0,
lLocator.GetOutputInstanceSceneView() );
		mApplicationDisplay.SetSceneName( 0,
lLocator.GetOutputInstanceScene() );
	}

		if ( lLocator.IsValid() )
	{
		mApplicationDisplay2.SetSceneViewName( 0,
lLocator.GetOutputInstanceSceneView() );
		mApplicationDisplay2.SetSceneName( 0,
lLocator.GetOutputDetailScene() );
	}
```

```
	mApplicationDisplay.SetZoom( -1 );

	mMeasurementGrid.SetColumns( "Instanca|Vrsta	|Pozicija X
|Pozicija Y	|Rotacija|Mjerilo|" );


	mSceneCombo.InsertString( 0, "Nista (samo slika)" );
	mSceneCombo.InsertString( 1, "Instance na sceni (Transformirani model)"
);
	mSceneCombo.SetCurSel( 0 );

	FillConfigurationCombo();



	return ( TRUE );  }

// ****************************************************************************
// Function....: OnClose
//
// Description.: Window Close event. Send by Windows when the user request to
//                           close the window.
//
// Parameters..: void
//
// Return......: void
// ****************************************************************************

void
CPartSortDlg::OnClose( void )
{
	WaitForCompletion();

	CDialog::OnClose();
}

// ****************************************************************************
// Function....: OnDestroy
//
// Description.: Window Destroy event. Send by Windows to signal the
//                           destruction of the window.
//
// Parameters..: void
//
// Return......: void
// ****************************************************************************

void
CPartSortDlg::OnDestroy( void )
{
	CDialog::OnDestroy();

	WaitForCompletion();
}

// ****************************************************************************
// Function....: OnPaint
//
// Description.: If you add a minimize button to your dialog, you will need
//                           the code below to draw the icon.  For MFC
applications using
//                           the document/view model, this is automatically done
for
//                           you by the framework.
//
// Parameters..: void
//
// Return......: void
// ****************************************************************************

void
CPartSortDlg::OnPaint( void )
{
	if ( IsIconic() )
	{
		CPaintDC lDc( this ); // device context for painting
```

```
		SendMessage( WM_ICONERASEBKGND, (WPARAM) lDc.GetSafeHdc(), 0 );

		// Center icon in client rectangle

		int lCxIcon = GetSystemMetrics( SM_CXICON );
		int lCyIcon = GetSystemMetrics( SM_CYICON );
		CRect lRect;
		GetClientRect( &lRect );
		int lX = (lRect.Width() - lCxIcon + 1) / 2;
		int lY = (lRect.Height() - lCyIcon + 1) / 2;

		// Draw the icon

		lDc.DrawIcon( lX, lY, mIcon );
	}
	else
	{
		CDialog::OnPaint();
	}
}

// ********************************************************************
// Function....: OnQueryDragIcon
//
********************************************************************

HCURSOR
CPartSortDlg::OnQueryDragIcon( void )
{
	return ( (HCURSOR) mIcon );
}

// ********************************************************************
// Function....: OnButtonSort
********************************************************************

void
CPartSortDlg::OnButtonSort( void )
{

	int tmp;
	int ii = 0;
	char *dat_redos = "C:/Adept/Disks/Disk_C/redos.txt";
	ifstream izvornik(dat_redos);
	int brcit = 0;
	CString red[200];
	int potroseni[200];
	char jedanod;
	CString rijec;
	int citam = 0;
	int insertrow = 0;

	if (!izvornik)


		goto nemafajle;

	izvornik >> ii;
	while (izvornik.get(jedanod)) {
		if (jedanod == '\n')
			break;
	};


	do {

		rijec.Empty();

		while (izvornik.get(jedanod)) {
		if (jedanod == '\n')
			break;
		rijec += jedanod;

		}
```

```
		red[citam] = rijec;

		++citam;

	}
	while (!izvornik.eof());
	nemafajle:


	HSProcessManager lManager = mApplicationControl.GetProcessManager();

	if ( !lManager.IsValid() )
	{
		return;
	}

	try
	{
		HSLocator    lLocator;
		CString			lMarkerName;
		DWORD        lTimeStart;
		DWORD        lTimeStop;
		int				done=0;

		mSortButton.EnableWindow( FALSE );

		do
		{
			mMeasurementGrid.DeleteAllItems();

			lTimeStart = GetTickCount();

			// Execute processes from Acquisition to Locator

			lManager.Execute( COleVariant( "Acquisition" ), COleVariant(
"Locator" ) );
			mApplicationDisplay.RemoveAllMarker();
			mApplicationDisplay2.RemoveAllMarker();

			lLocator.AttachDispatch( lManager.GetProcess( COleVariant(
"Locator" ) ) );

			if ( lLocator.IsValid() )


			{



				char *datoteka =
"C:/Adept/Disks/Disk_C/kord_ins.txt";
				ofstream ponornik(datoteka);

				CString      UkBrojInst;
				UkBrojInst.Format( "%d",
					ii);

				ponornik << UkBrojInst;
				ponornik << endl;

				ponornik.close();

				for (int jk = 0; jk < 200; jk++){

					potroseni[jk] = 0;}

		for (int j = 0; j < ii; j++) {

			done = 0;
			for ( int i = 0; i < lLocator.GetInstanceCount(); ++i ) {
				if (done) break;

				if (red[j] == lLocator.GetInstanceModelName(i)) {

					if (potroseni[i] != 1) {
```

```
                                done = 1;
                                potroseni[i] = 1;

                                CString      lBuffer;

                                tocka.xkor = lLocator.GetInstanceTranslationX(
i );
                                tocka.ykor = lLocator.GetInstanceTranslationY(
i );

                                rotacija = -(lLocator.GetInstanceRotation( i )
+ kut/pi*180);

                                tockarobot = transh(ishodslike, tocka, kut);

                                tmp = i+1;

                                lBuffer.Format( "%d|%s|%.2f|%.2f|%.2f|%.2f|",
//                                      i, red[tmp],
                                      i, lLocator.GetInstanceModelName(i),
                                      tockarobot.xkor,
                                      tockarobot.ykor,
                                      rotacija,
                                      lLocator.GetInstanceScaleFactor( i ) );
                                mMeasurementGrid.InsertItem(insertrow++,
lBuffer );

                                // Upisivanje u file

                                CString      iname;
                                iname.Format( "%s",
                                      lLocator.GetInstanceModelName(i));

                                CString      transx;
                                transx.Format( "%f",
                                      tockarobot.xkor);

                                CString      transy;
                                transy.Format( "%f",
                                      tockarobot.ykor);

                                CString      rotz;
                                rotz.Format( "%f",
                                      rotacija);

//                              char *datoteka = "C:/test.txt";
                                ofstream ponornik(datoteka, ios::app ,
filebuf::openprot);

                                ponornik << iname;
                                ponornik << endl;

                                ponornik << transx;
                                ponornik << endl;

                                ponornik << transy;
                                ponornik << endl;

                                ponornik << rotz;
                                ponornik << endl;

                                ponornik.close();

                                lMarkerName.Format( "%s(%d)",
lLocator.GetInstanceModelName( i ), i );
                                mApplicationDisplay.AddPointMarker(
lMarkerName,
      lLocator.GetInstanceReferencePointPositionX(i, 0),
      lLocator.GetInstanceReferencePointPositionY(i, 0), TRUE );
                                mApplicationDisplay.SetMarkerColor(
lMarkerName, HSDisplay::hsGreen );
                                mApplicationDisplay.SetMarkerAnchorStyle(
lMarkerName, HSDisplay::hsCross );
```

```
                                    mApplicationDisplay.SetPointMarkerConstraints(
lMarkerName, HSDisplay::hsPointNoEdit );
                                    mApplicationDisplay.SetMarkerDisplayName(
lMarkerName, TRUE );

                                    lMarkerName.Format( "Axes%d", i );
                                    mApplicationDisplay.AddAxesMarker(
lMarkerName,
                                          lLocator.GetInstanceTranslationX( i ),
                                          lLocator.GetInstanceTranslationY( i ),
                                          30, 30, lLocator.GetInstanceRotation( i
), 90, TRUE );
                                    mApplicationDisplay.SetMarkerColor(
lMarkerName, HSDisplay::hsYellow );
                                    mApplicationDisplay.SetAxesMarkerConstraints(
lMarkerName,
                                          (HSDisplay::hsAxesMarkerConstraints)(
HSDisplay::hsAxesNoEdit + HSDisplay::hsAxesFixedSize ) );
                                    mApplicationDisplay.SetAxesMarkerLabel1(
lMarkerName, "X" );
                                    mApplicationDisplay.SetAxesMarkerLabel2(
lMarkerName, "Y" );

                                    }
                                    } else {
                        }

                              }

                        }

                  if (insertrow != ii) {
                                    ofstream ponornik(datoteka);

                                    CString      UkBrojInst;
                                    UkBrojInst.Format( "%d",
                                          0);

                                    ponornik << UkBrojInst;
                                    ponornik << endl;

                                    ponornik.close();
                  }
                  }
                        lTimeStop = GetTickCount();
                        UpdateElapsedTime( lTimeStart, lTimeStop );
                        mApplicationDisplay2.Refresh();

                        DoEvents();
                  }
                  while ( mContinuousCheck.GetCheck() == 1 );

                  mSortButton.EnableWindow( TRUE );
            }
            catch( ... )
            {
                  mSortButton.EnableWindow( TRUE );

                  throw ;
            }
}

// ********************************************************************************
// Function....: OnButtonLoad
********************************************************************************

void
CPartSortDlg::OnButtonLoad( void )
{
      HSAcquisitionDevice lAcquisition =
            mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );

      if ( lAcquisition.IsValid() )
      {
```

```
            lAcquisition.LoadEmulationDatabase( HEXSIGHT_NONE, HEXSIGHT_NONE
);

          SetAcquisitionConfiguration();
      }
}

// ****************************************************************************
// Function....: OnButtonQuit
****************************************************************************

void
CPartSortDlg::OnButtonQuit( void )
{
      WaitForCompletion();

      EndDialog( IDOK );
}

// ****************************************************************************
// Function....: OnDropdownComboConfiguration
****************************************************************************

void
CPartSortDlg::OnDropdownComboConfiguration( void )
{
      FillConfigurationCombo();
}

// ****************************************************************************
// Function....: OnPropertiesChangeControlApplication
****************************************************************************

void
CPartSortDlg::OnPropertiesChangeControlApplication( LPCTSTR pProcessName )
{
      CString      const lProcessName( pProcessName );

      if (   ( lProcessName.CompareNoCase( "Acquisition" ) == 0 )
            || ( lProcessName.CompareNoCase( "HSProcessManager" ) == 0 ) )
      {
            FillConfigurationCombo();
      }
}

// ****************************************************************************
// Function....: OnConfigurationExploreHexSightApplication
****************************************************************************

void
CPartSortDlg::OnConfigurationExploreHexSightApplication( void )
{
      mApplicationControl.ShowInterface( FALSE );
}

// ****************************************************************************
// Function....: OnConfigurationAcquisitionProperties
****************************************************************************

void
CPartSortDlg::OnConfigurationAcquisitionProperties( void )
{
      HSAcquisitionDevice lAcquisition =
            mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );

      if ( lAcquisition.IsValid() )
      {
            lAcquisition.ShowProperties( FALSE , HEXSIGHT_NONE );
      }
}

// ****************************************************************************
// Function....: OnConfigurationLocatorProperties
****************************************************************************
```

```
void
CPartSortDlg::OnConfigurationLocatorProperties( void )
{
	HSLocator lLocator =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Locator" ) );

	if ( lLocator.IsValid() )
	{
		lLocator.ShowProperties( FALSE, HEXSIGHT_NONE );
	}
}

// ****************************************************************************
// Function....: OnConfigurationConfigurationFileLoad
// ****************************************************************************

void
CPartSortDlg::OnConfigurationConfigurationFileLoad( void )
{
	mApplicationControl.GetProcessManager().LoadConfiguration( 0, "" );
}

// ****************************************************************************
// Function....: OnConfigurationConfigurationFileSave
// ****************************************************************************

void
CPartSortDlg::OnConfigurationConfigurationFileSave( void )
{
	mApplicationControl.GetProcessManager().SaveConfiguration( 0, "" );
}

// ****************************************************************************
// Function....: OnConfigurationCameraCalibrationNew
// ****************************************************************************

void
CPartSortDlg::OnConfigurationCameraCalibrationNew( void )
{
	HSAcquisitionDevice lAcquisition =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );

	if ( lAcquisition.IsValid() )
	{
		if ( lAcquisition.GetInputDeviceType( COleVariant(
lAcquisition.GetInputDevice( HEXSIGHT_NONE ) ) ) !=
HSAcquisitionDevice::hsFileEmulation )
		{
			lAcquisition.ShowCalibDistortionModelDialog( TRUE,
HEXSIGHT_NONE );
		}
		else
		{
			MessageBox ( "Camera calibration cannot be modified in File
Emulation mode", "New Calibration", MB_ICONEXCLAMATION );
		}
	}
}

// ****************************************************************************
// Function....: OnConfigurationCameraCalibrationLoad
// ****************************************************************************

void
CPartSortDlg::OnConfigurationCameraCalibrationLoad( void )
{
	HSAcquisitionDevice lAcquisition =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );

	if ( lAcquisition.IsValid() )
	{
		lAcquisition.LoadCalibration( HEXSIGHT_NONE, HEXSIGHT_NONE );
	}
```

```
}

// ****************************************************************************
// Function....: OnConfigurationCameraCalibrationSave
// ****************************************************************************

void
CPartSortDlg::OnConfigurationCameraCalibrationSave( void )
{
	HSAcquisitionDevice lAcquisition =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );

	if ( lAcquisition.IsValid() )
	{
		lAcquisition.SaveCalibration( HEXSIGHT_NONE, HEXSIGHT_NONE );
	}
}

// ****************************************************************************
// Function....: OnWindowRealTimeDisplay
// ****************************************************************************

void
CPartSortDlg::OnWindowRealTimeDisplay( void )
{
	HSAcquisitionDevice lAcquisition =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );

	if ( lAcquisition.IsValid() )
	{
		lAcquisition.ShowRealTimeDisplay( FALSE, HEXSIGHT_NONE,
HEXSIGHT_NONE );
	}
}

// ****************************************************************************
// Function....: OnWindowLocatorViewer
// ****************************************************************************

void
CPartSortDlg::OnWindowLocatorViewer( void )
{
	HSLocator lLocator =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Locator" ) );

	if ( lLocator.IsValid() )
	{
		lLocator.ShowResultsViewer( FALSE );
	}
}

// ****************************************************************************
// Function....: UpdateElapsedTime
// ****************************************************************************

void
CPartSortDlg::UpdateElapsedTime( DWORD const pStart, DWORD const pStop )
{
	DWORD		lElapsed;
	CString			lBuffer;

	if ( pStop < pStart )
	{
		lElapsed = (DWORD)(-1) - pStart + pStop + 1;
	}
	else
	{
		lElapsed = pStop - pStart;
	}

	lBuffer.Format( "%lu", lElapsed );
	mElapsedTime.SetWindowText( lBuffer );
}
```

```
// ********************************************************************************
// Function....: FillConfigurationCombo
********************************************************************************

void
CPartSortDlg::FillConfigurationCombo( void )
{
	mConfigurationCombo.ResetContent();

	HSAcquisitionDevice lAcquisition =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );

	if ( !lAcquisition.IsValid() )
	{
		mLoadButton.EnableWindow( FALSE );
		mEmulationDatabase.SetWindowText( "" );
	}
	else
	{
		mFillingConfigurationCombo = true;

		try
		{
			for ( int i = 0; i < lAcquisition.GetConfigurationCount();
++i )
			{
				mConfigurationCombo.InsertString( i,
					lAcquisition.GetConfigurationName(
COleVariant( (long)i ) ) );
			}

			if ( mConfigurationCombo.GetCount() > 0 )
			{
				COleVariant lVarDefaultConfig =
lAcquisition.GetConfigurationDefault();
				LPCVARIANT   lDefaultConfig = lVarDefaultConfig;

				switch( lDefaultConfig->vt )
				{
				case VT_BSTR:
					mConfigurationCombo.SelectString( -1, CString(
lDefaultConfig->bstrVal ) );
					break;

				case VT_I2:
					mConfigurationCombo.SetCurSel( lDefaultConfig-
>iVal );
					break;
				}
			}
		}
		catch( ... )
		{
			mFillingConfigurationCombo = false;

			throw ;
		}

		mFillingConfigurationCombo = false;
	}

	SetAcquisitionConfiguration();
}

// ********************************************************************************
// Function....: OnSelendokComboScene
********************************************************************************

void
CPartSortDlg::OnSelendokComboScene( void )
{
	HSLocator lLocator =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Locator" ) );
```

```
	if ( lLocator.IsValid() )
	{
		switch( mSceneCombo.GetCurSel() )
		{
		case 0: // No scene is displayed
			mApplicationDisplay.SetScenePenWidth( 0,
HSDisplay::hsPenWidthNone );
			lLocator.SetOutputMode( HSLocator::hsNoGraphics );
			break;
		case 1: // The Instance Scene is displayed
			mApplicationDisplay.SetScenePenWidth( 0,
HSDisplay::hsPenWidthThick );
			lLocator.SetOutputMode( HSLocator::hsTransformedModel );
			lLocator.Execute();
			break;
		}
	}

	mApplicationDisplay.RefreshDisplay();
}

// ****************************************************************************
// Function....: OnSelendokComboConfiguration
****************************************************************************

void
CPartSortDlg::OnSelendokComboConfiguration( void )
{
	SetAcquisitionConfiguration();
}

// ****************************************************************************
// Function....: WaitForCompletion
****************************************************************************

void
CPartSortDlg::WaitForCompletion( void )
{
	if ( mContinuousCheck.GetCheck() != 0 )
	{
		mContinuousCheck.SetCheck( 0 );
	}
}

// ****************************************************************************
// Function....: SetAcquisitionConfiguration
****************************************************************************

void
CPartSortDlg::SetAcquisitionConfiguration( void )
{
	HSAcquisitionDevice lAcquisition =
		mApplicationControl.GetProcessManager().GetProcess( COleVariant(
"Acquisition" ) );

    if ( !lAcquisition.IsValid() )
	{
	 mLoadButton.EnableWindow( FALSE );
	 mEmulationDatabase.SetWindowText( "" );
    }
	else
	{
		if (   ( mFillingConfigurationCombo == false )
			&& ( mConfigurationCombo.GetCurSel() >= 0 ) )
		{
			lAcquisition.SetConfigurationDefault( COleVariant(
(long)mConfigurationCombo.GetCurSel() ) );
		}

		if ( lAcquisition.GetInputDeviceType(
			COleVariant( (long)lAcquisition.GetInputDevice(
HEXSIGHT_NONE ) ) )
				== HSAcquisitionDevice::hsFileEmulation )
		{
			mLoadButton.EnableWindow( TRUE );
```

```
                    mEmulationDatabase.SetWindowText(
                          lAcquisition.GetEmulationDatabaseFileName(
HEXSIGHT_NONE ) );
              }
              else
              {
                    mLoadButton.EnableWindow( FALSE );
                    mEmulationDatabase.SetWindowText( "" );
              }
       }
}

void CPartSortDlg::OnDijalogKordinat(void)
{
       CKordDijl Popap;


       Popap.m_xish = ishodslike.xkor;
       Popap.m_yish = ishodslike.ykor;
       Popap.m_xaps = tocapscisa.xkor;
       Popap.m_yaps = tocapscisa.ykor;

       float proba = kuthomo(ishodslike.xkor, ishodslike.ykor, tocapscisa.xkor,
tocapscisa.ykor);
       kut = proba;

//     CKordFloat proba1 = transh(ishodslike, tocperrob, proba);

       Popap.m_temp = proba;



       if(Popap.DoModal() == IDOK)
       {


       ishodslike.xkor = Popap.m_xish;
       ishodslike.ykor = Popap.m_yish;
       tocapscisa.xkor = Popap.m_xaps;
       tocapscisa.ykor = Popap.m_yaps;
       proba = Popap.m_temp;

       };


}
```

P.1.5. KordFloat.h

```
// KordFloat.h: interface for the CKordFloat class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_KORDFLOAT_H__B64E47B9_FFC4_4D37_A71F_0021B7F387FE__INCLUDED_)
#define AFX_KORDFLOAT_H__B64E47B9_FFC4_4D37_A71F_0021B7F387FE__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CKordFloat
{
public:
	float ykor;
	float xkor;
	CKordFloat();
	CKordFloat(float,float);
	virtual ~CKordFloat();

};

#endif //
!defined(AFX_KORDFLOAT_H__B64E47B9_FFC4_4D37_A71F_0021B7F387FE__INCLUDED_)
```

P.1.6. KordFloat.cpp

```
// KordFloat.cpp: implementation of the CKordFloat class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "PartSort.hpp"
#include "KordFloat.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CKordFloat::CKordFloat()
{

	xkor = 0;
	ykor = 0;


}

CKordFloat::CKordFloat(float x, float y)
{

	xkor = x;
	ykor = y;


}



CKordFloat::~CKordFloat()
{

}
```

P.1.7. KordFloat.h

```
// KordFloat.h: interface for the CKordFloat class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_KORDFLOAT_H__B64E47B9_FFC4_4D37_A71F_0021B7F387FE__INCLUDED_)
#define AFX_KORDFLOAT_H__B64E47B9_FFC4_4D37_A71F_0021B7F387FE__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CKordFloat
{
public:
	float ykor;
	float xkor;
	CKordFloat();
	CKordFloat(float,float);
	virtual ~CKordFloat();

};

#endif //
!defined(AFX_KORDFLOAT_H__B64E47B9_FFC4_4D37_A71F_0021B7F387FE__INCLUDED_)
```

P.1.8. KordFloat.cpp

```
// KordFloat.cpp: implementation of the CKordFloat class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "PartSort.hpp"
#include "KordFloat.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CKordFloat::CKordFloat()
{

	xkor = 0;
	ykor = 0;


}

CKordFloat::CKordFloat(float x, float y)
{

	xkor = x;
	ykor = y;


}



CKordFloat::~CKordFloat()
{

}
```

P.2. Kod programske podrške za upravljanje robotom

```
.PROGRAM sklopi4()

 ; ABSTRACT:
 ;
 ;     Program za sklapanje u nesredjenoj robotskoj okolini
 ;     Program se najprije povezuje na NFSi otvara 3 zapisnika
 ;     U prvom - kord_ins.txt  su koordinate ugradbenih dijelova
 ;     s vizijskog sustava u odgovarajucem formatu
 ;     U drugom - vis_ins.txt  su visine tocke hvatanja
 ;     ugradbenih dijelova
 ;     U trecem - put_sklap.txt su definirane putanje sklapanja
 ;
 ;     Program cita lkoordinate s viijskog sustava pa ih sklapa prema zadanim
 ;     putanjama
 ;
 ;    Ugradjeneo je i nekoliko provjera konzistencije ulaznih parametara
 ;    Ukoliko je u zapisniku put_sklap.txt prvi record 0, sklapanje se ignorira
 ;
 ;                 * Copyright (c) 2002 by FSB
 ;-------------------------------------------------------------
 ;Prvo se povezujemo s NFS-om

       ATTACH (lun, 4) "NFS"
       greska = IOSTAT(lun)
       IF greska < 0 THEN
           TYPE "nemrem se zvezati na nfs", $ERROR(greska)
           GOTO 100
       END

       ATTACH (vlun, 4) "NFS"
       greska = IOSTAT(vlun)
       IF greska < 0 THEN
           TYPE "nemrem se zvezati na nfs", $ERROR(greska)
           GOTO 100
       END

       ATTACH (dlun, 4) "NFS"
       greska = IOSTAT(dlun)
       IF greska < 0 THEN
           TYPE "nemrem se zvezati na nfs", $ERROR(greska)
           GOTO 100
       END

       ATTACH (plun, 4) "NFS"
       greska = IOSTAT(dlun)
       IF greska < 0 THEN
           TYPE "nemrem se zvezati na nfs", $ERROR(greska)
           GOTO 100
       END
           ; otvori marija vrata fajlu za citanje koordinata iz vizijskog
           ;  sustava

       FOPENR (lun) "kord_ins.txt"
       mistejk = IOSTAT(lun)
       IF mistejk < 0 THEN
           TYPE "fajla se ne otvara!!!!!!", $ERROR(mistejk)
           GOTO 100
       END


       FOPENR (vlun) "vis_ins.txt"
       mistejk = IOSTAT(vlun)
       IF mistejk < 0 THEN
           TYPE "fajla se ne otvara!!!!!!", $ERROR(mistejk)
           GOTO 100
       END
```

```
      FOPENR (dlun) "put_sklap.txt"
      mistejk = IOSTAT(dlun)
      IF mistejk < 0 THEN
          TYPE "fajla se ne otvara!!!!!!", $ERROR(mistejk)
          GOTO 100
      END

      FOPENR (plun) "put_odlagaliste.txt"
      mistejk = IOSTAT(plun)
      IF mistejk < 0 THEN
          TYPE "fajla se ne otvara!!!!!!", $ERROR(mistejk)
          GOTO 100
      END

      SPEED 30 ALWAYS

      SET poc_l = TRANS(230,0,150,0,180,0)

      MOVE poc_l

      BREAK

          ; -------------------------------------------------------------
          ; broj detektiranih dijelova nd
          ; slijede deklaracije varijabli

      nd = 1
      $iddio = ""
      x = 0
      z = 0
      fi = 0

      $iddio = ""

      nvis = 1
      $ivis = ""
      vis = 0

      nplan = 1
      $iplan = ""
      noper = 0
      xp = 0
      yp = 0
      zp = 0
      fip = 0
      rotp = 0
      valp = 0
  ; citam kolko je i dali su pronadjeni dijelovi


      READ (lun) nd
      IF nd < 1 THEN
          TYPE "Nisu pronadjeni odgovarajuci dijelovi ugradbeni dijelovi"
          GOTO 100
      END
          ; provjeravam ima li putanja za plan montaze

      READ (dlun) nplan
      IF nplan < 1 THEN
          TYPE "putanje sklapanja nisu zadane"
      END

      READ (vlun) nvis
      IF nvis < 1 THEN
          TYPE "visine ugradbenih dijelova nisu zadane"
      END

      IF ((nplan <> nd) AND (nplan <> 0)) OR ((nvis <> nd) AND (nvis <> 0))
THEN

          TYPE "zadane putanje sklapanja nisu u skladu s zadanim redoslijedom
ili visinama"
          GOTO 100
      END
```

```
npoziva = 0
DO
    READ (lun) $iddio
    READ (lun) x
    READ (lun) y
    READ (lun) fi
    npoziva = npoziva+1
    IF nvis <> 0 THEN
       READ (vlun) $ivis
       READ (vlun) vis

       IF $ivis <> $iddio THEN
           TYPE "zadana imena nisu konzistentna"
           GOTO 100
       END
    ELSE
       vis = 0
    END
    z = -188.885+vis
    TYPE $iddio
    TYPE x
    TYPE y
    TYPE fi

    SET kraj_l = TRANS(x,y,z,0,180,fi)

    APPRO kraj_l, 120
    DELAY 1
    BREAK
    MOVE kraj_l
    BREAK
    DELAY 3
    DEPART 120
    DELAY 1
    BREAK

          ; sad citam putanje sklapanja

    IF nplan == 0 THEN           ;nisu zadane putanje sklapanja
       GOTO 90
    END

    READ (dlun) $iplan
    IF $iplan <> $iddio THEN
       TYPE "zadana imena nisu konzistentna"
       GOTO 100
    END
    noper = 0
    READ (dlun) noper

    xp = 0
    yp = 0
    zp = 0
    fip = 0
    rotp = 0
    valp = 0

    brojsklap = 0

    TYPE "sklapam dio "+$iddio

    DO
       brojsklap = brojsklap+1

       READ (dlun) xp
       READ (dlun) yp
       READ (dlun) zp
       READ (dlun) fip
       READ (dlun) rotp
       READ (dlun) valp
       TYPE xp
       TYPE yp
       TYPE zp
       TYPE fip
       TYPE rotp
       TYPE valp
```

```
            SET tocki = TRANS(xp,yp,zp,fip,rotp,valp)

            IF brojsklap == 1 THEN

                MOVE tocki

                BREAK

                DELAY 1

            ELSE
                MOVES tocki

                BREAK

                DELAY 1

            END

         UNTIL brojsklap == noper
   90
      UNTIL npoziva == nd

      nodl = 0

      READ (plun) nodl
      IF nodl <= 0 THEN

          TYPE "Nisu zadane za odlaganje"
          GOTO 100
      END

      tipskl = 0
      npoziva = 0

      DO

          npoziva = npoziva+1
          READ (plun) tipskl
          READ (plun) xp
          READ (plun) yp
          READ (plun) zp
          READ (plun) fip
          READ (plun) rotp
          READ (plun) valp

          SET tocki = TRANS(xp,yp,zp,fip,rotp,valp)

          IF tipskl == 0 THEN

             MOVE tocki
             BREAK
             DELAY 1

          ELSE
             MOVES tocki
             BREAK
             DELAY 1
          END


      UNTIL npoziva == nodl

  100
      MOVE poc_l

      BREAK
.END
```

P.3. Zapisnik s rezultatima obrade slike

```
2
L1
108.228394
10.720719
50.089138
T0
94.647980
107.309296
-76.746956
```

P.4. Zapisnik s definicijama na putanja sklapanja

```
3
t0
2
-21
467
-160
50
173
-53
75
523
-131
66
163
-26
t1
2
345
-400
-131
-64
163
-26
349
-392
-52
-46
145
15
t2
2
-21
467
-160
50
173
-53
75
523
-131
66
163
-26
```