SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Pregled metoda u području prianjanja proteina

Jura Ćurić

Voditelj: *Dr. sc. Mile Šikić*

Zagreb, lipanj 2009.

Sadržaj

1. Uvod

Kroz cijelu povijest čovječanstva ljudi se neprestano bore protiv raznih bolesti. S vremenom, izumljeno je mnoštvo lijekova koji spašavaju mnoge živote pa tako neke bolesti koje su u prošlosti bile kobne kao što su npr. sifilis i malarija, danas više ne predstavljaju problem. S druge strane, čovječanstvo je suočeno s novim smrtonosnim bolestima kao što su npr. AIDS i razni oblici gripa. Lijekove za navedene bolesti za sad još nije moguće susresti na policama ljekarni. Stoga, kako bi se omogućila daljnja borba protiv bolesti, neophodno je posvetiti mnogo pažnje i truda području proizvodnje lijekova. Jedan od načina jest utvrđivanje koji će par proteina ili pak protein i neka druga molekula međusobno tvoriti kompleks (hoće li reagirati), te pokušava predvidjeti trodimenzionalna struktura eventualnog rezultirajućeg kompleksa. Upravo struktura rezultirajućeg kompleksa govori o njegovom djelovanju te predstavlja eventualno cjepivo.

Prianjanje proteina je područje bioinformatike koje se bavi računalnim modeliranjem proteinskih interakcija. Glavno svojstvo koje se proučava u tu svrhu jest komplementarnost površina dvaju proteina, tj. u kojoj se mjeri preklapaju površine dvaju proteina. U tu svrhu razvijeno je mnoštvo matematičkih i računalnih metoda koje modeliraju trodimenzionalnu strukturu proteina te potom utvrđuju u kolikoj se mjeri njihove površine preklapaju. Najpoznatije metode koje se koriste u tu svrhu su geometrijsko raspršenje, FFT te Monte Carlo metoda. U daljnjem tekstu opisani su principi navedenih metoda te je navedeno kako se mogu koristiti u području prianjanja proteina. Važno je također napomenuti da navedene metode ne predstavljaju zatvoren sustav, tj. jedini moguć pristup problemu prianjanja proteina. Ipak, većina alata koji se koriste u tu svrhu bazirani su na geometrijskom raspršenju, FFT te Monte Carlo metodi. Iako opisi proteinskih interakcija pomoću navedenih metoda nisu savršeni, potrebno ih je dobro shvatiti kako bi daljnji razvoj ovog područja bio moguć.

2. Geometrijsko raspršenje

Jedna od alternativa iscrpnom pretraživanju prostora jest geometrijsko raspršenje (eng. *geometric hashing*) koje se koristi u računalnom vidu i kod prianjanja proteina. Budući da je zbog trodimenzionalne strukture problem prianjanja sam po sebi veoma složen, prikladnije je postupak opisati na primjeru raspoznavanja objekata na dvodimenzionalnoj slici.

Zamislimo da je potrebno konstruirati stoj koji će biti u mogućnosti prepoznati sve objekte i alate na podu tvornice. U slučaju da se na podu nalazi samo par stotinjak objekata, moguće je stvoriti bazu podataka tih objekata i pohraniti ju u memoriju stroja za prepoznavanje. Nakon što stroj pomoću senzora primi sliku, mora biti u mogućnosti da u kratkom vremenu prepozna sve objekte koji se nalaze na slici. Drugim riječima, mora u veoma kratkom vremenu iz memorije dohvatiti sve objekte koji se nalaze na danoj slici. Iako je dani problem jednostavan za čovjeka, za implementaciju računalnog rješenja potrebno je riješiti nekoliko složenih problema:

- Na slici, svi objekti koje treba prepoznati su translatirane, rotirane i skalirane verzije modela, tj. objekata koji su pohranjeni u memoriji stroja. Također, u obzir treba uzeti i projekciju trodimenzionalnih objekata na dvodimenzionalnu sliku.
- Objekti na slici mogu prekrivati jedan drugog. Također, na slici se mogu pojaviti neki objekti koji nemaju svoj model u memoriji stroja
- Postupak uspoređivanja svih modela iz baze podataka stroja i promatrane slike je računalno nedjelotvoran pa takav pristup treba izbjegavati. Npr. ako se na slici pojavljuju samo okrugli predmeti, nema ih smisla uspoređivati s pravokutnim oblicima koji se nalaze u bazi podataka stroja

Potrebno je dakle primijeniti metodu koja omogućuje pristup samo relevantnim podatcima smanjujući tako vrijeme prepoznavanja. Jedna od takvih metoda je upravo geometrijsko raspršenje.

2.1. Osnovna ideja

Kao što je prije spomenuto, da bi sustav mogao uopće prepoznavati neki objekt potrebno je prethodno konstruirati bazu podataka modela, tj. referentnih objekata. Ova faza naziva se ***faza pretprocesiranja***. U fazi pretprocesiranja se svaki model kodira te se dobivena informacija sprema u raspršenu tablicu (*hash table*) . Sadržaj raspršene tablice ne ovisi o slici na kojoj se nalaze objekti koje treba raspoznati već samo o odabranim, referentnim objektima. Dakle, konstrukcija raspršene tablice ne utječe na vrijeme raspoznavanja. Informacijama pohranjenim u raspršenoj tablici se pristupa pomoću ključa koji je geometrijskog oblika, preciznije ključ predstavljaju koordinate neke točke.

U ***fazi raspoznavanja*** sustav kao ulaz prima sliku na kojoj se nalaze objekti koje treba raspoznati. Pomoću koordinata pojedinog objekta se pristupa prethodno konstruiranoj *hash*-tablici, te na temelju informacija pohranjenim pod ključem kojeg predstavljaju koordinate se određuje o kojem se objektu radi.

2.2. Analiza postupka geometrijskog raspršenja

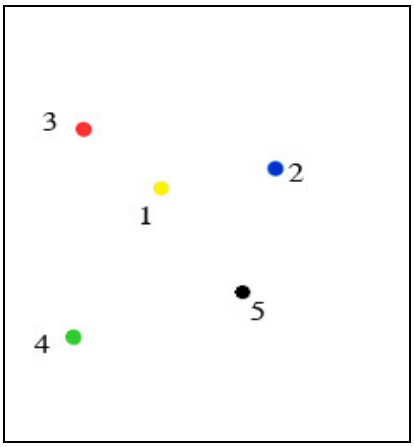
2.2.1. Ekstrakcija značajki

U svakom modelu raspoznavanja neizostavan dio predstavlja korak izlučivanja značajki. Relevantne značajke su informacije koje moraju biti karakteristične za pojedini objekt, odnosno za skup objekata koji pripadaju istom razredu. Npr. ako slika sadrži dva plava pravokutnika i dva plava kruga, onda oba pravokutnika pripadaju razredu „pravokutnik“ dok oba kruga pripadaju razredu „krug“. U ovom slučaju se boja ne može smatrati relevantnom značajkom. Naime, i pravokutnici i krugovi su iste boje pa ih ne možemo razlikovati na temelju boje. S druge strane, uzmemo li u obzir oblik rubova u obzir, lako je razlikovati krugove od pravokutnika. Stoga oblik rubova jest relevantna značajka. Kad bi umjesto krugova na slici bili kvadrati, oblik rubova više ne bi bila relevantna značajka već bi kao značajku trebalo odabrati npr. omjer duljine dvije susjedne stranice.

Sustav najprije mora u po određenom kriteriju odabrati na koji će način i koje značajke uzimati u obzir. Skup svih značajki reprezentiran je u sustavu pomoću skupa točaka gdje svaka točka predstavlja mjesto pojedine značajke. Pojedinoj točki također može biti pridijeljeno više značajki.

2.2.2. Primjer jednostavnog raspoznavanja objekta

Iako veoma pojednostavljen, sljedeći primjer prikazuje princip raspoznavanja objekata na apstraktnoj razini. Pretpostavimo da je pomoću značajki izgrađen referentni objekt, tj. model. Kao što je prije spomenuto, lokacije značajki su predstavljene točkama pa se i sljedeći model sastoji od točaka.



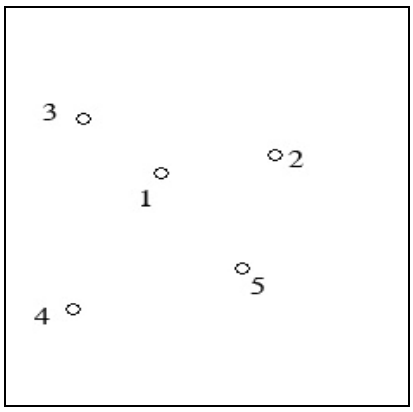
Slika 1. Model - referentni objekt

Predstavljeni model se sastoji od pet točaka. Pretpostavimo li na trenutak da svaka točka nekog modela sadrži jedinstvenu boju, kao ulaz u tablicu raspršenja može se koristiti upravo boja točke. U tom slučaju bi podatak pridružen nekoj boji u tablici raspršenja predstavljao oznaku modela kojem navedena boja (ključ) pripada. U fazi raspoznavanja sustav bi očitao boje točaka sa slike na kojoj želimo prepoznati neki objekt i pristupio tablici raspršenja na temelju pojedine očitane točke. Podatak kojem sustav pristupa pomoću ključa (boje točke) jesu svi modeli koji sadrže očitanu točku. Pri svakom pristupu nekom modelu povećava se brojač pristupa tog modela. Tada modeli s najviše pristupa imaju najveću vjerojatnost da se upravo oni nalaze na promatranoj slici.

2.2.3. Općeniti primjer raspoznavanja objekta

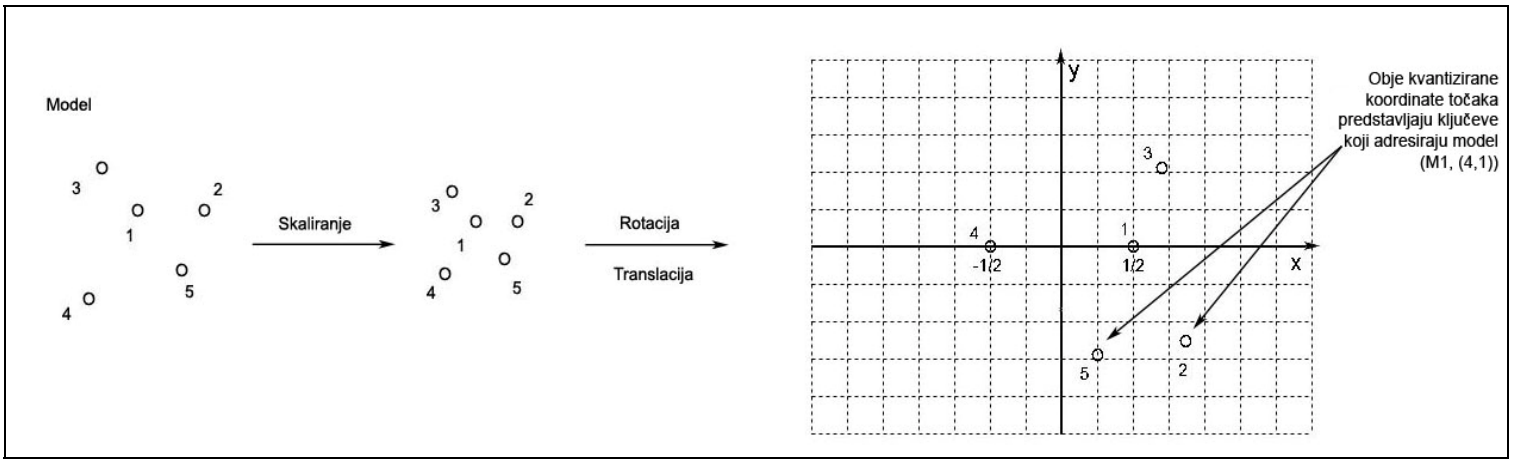
U prethodnom primjeru je pretpostavljeno da svaka točka sadrži jedinstvenu boju. No kako riješiti problem raspoznavanja u slučaju kada točke koje predstavljaju model nemaju nikakav atribut osim njihove geometrijske konfiguracije? Postoji li u tom slučaju karakteristična, jedinstvena „geometrijska boja“ ? Da, „geometrijska boja“ pojedine točke je upravo skup koordinata te točke. Opet, postavlja se pitanje kako pridijeliti skup koordinata nekoj točki, odnosno kako odrediti referentne točke u koordinatnom sustavu? Jedan od načina jest izabrati jedan par točaka koje jednoznačno određuju referentni koordinatni sustav.

Promotrimo sljedeći model:



Slika 2 - Model M_1

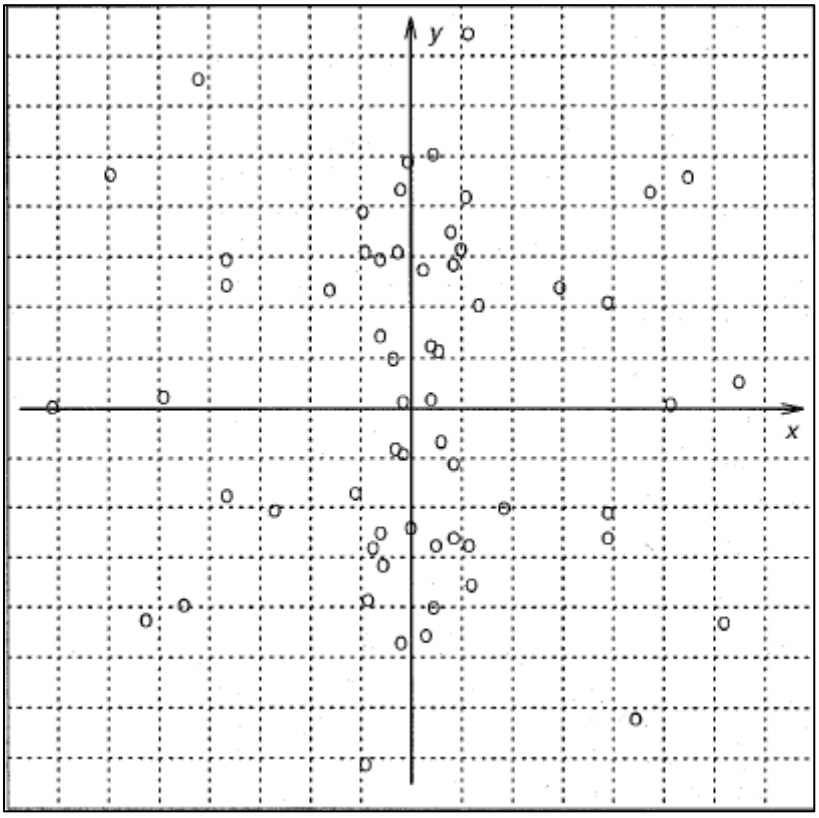
Definirajmo pomoću para točaka $\mathbf{p_4}$, $\mathbf{p_1}$ uređenu bazu koja određuje referentni koordinatni sustav. Također potrebno je skalirati model M_1 tako da veličina vektora baze $\overrightarrow{p_4p_1}$ u Oxy koordinatnom sustavu bude jedinična. Središnja točka vektora $\overrightarrow{p_4p_1}$ određuje središte koordinatnog sustava, a vektor $\overrightarrow{p_4p_1}$ ima orijentaciju u smjeru pozitivne x-osi. Na taj je način određen referentni koordinatni sustav za jednu inačicu modela M_1. Naime, mogli smo odabrati neki drugi par točaka i pomoću njih odrediti drugi referentni koordinatni sustav.



Slika 3. Određivanje referentnog koordinatnog sustava

Definiranjem referentnog koordinatnog sustava, ostale točke modela M_1 zauzimaju jednoznačno određene lokacije. Upravo te koordinate služe kao ključ tablice raspršenja. No, potrebno ih je prethodno kvantizirati. Svaki kvadratić sa gornje slike predstavlja jedan ključ tablice raspršenja. Vrijednost koja se nalazi pod ključem svih preostalih točaka koje ne čine bazu, tj. vrijednost čiji su ključevi kvantizirane koordinate preostalih točaka jest oznaka modela i baze koja definira referentni koordinatni sustav, dakle $(M_1,(4,1))$.

Budući da neki od objekata sa slike mogu biti prekriveni, ne možemo biti sigurni da se obje točke koje čine bazu nalaze u svakom dijelu slike gdje je prisutan model M_1. Stoga je potrebno kodirati model M_1 pomoću svih mogućih uređenih parova baza na isti način kao što je učinjeno za uređeni par (p_4, p_1). Za model M_1 postoji ukupno 20 mogućih odabira baza. Budući da za svaki uređeni par baza preostaju još tri točke koje ne čine bazu, postoji ukupno 60 mogućih ključeva koji adresiraju oznaku pripadnosti modelu M_1. Npr. odaberemo li kao bazu uređeni par (p_2, p_3), kvantizirane koordinate preostalih točaka p_1, p_4 i p_5 će adresirati oznaku pripadnosti modelu M_1 s pripadnom bazom (p_2, p_3), tj. vrijednost pod ključem kvantiziranih koordinata jest $(M_1,(2,3))$.



Slika 4. Sve moguće nekvantizirane koordinate, tj. budući ključevi tablice raspršenja. Točke koje određuju bazu nisu prikazane budući da njihove koordinate ne čine ključeve tablice raspršenja

2.2.4. Sažetak faze pretprocesiranja

Za svaki model M izvrši sljedeće operacije:

1. Izluči značajke modela te točkama označi njihove lokacije. Pretpostavi da je određeno *n* takvih točaka.

2. Za svaki uređeni par točaka, tj. bazu izvrši sljedeće operacije:
   a. Izračunaj koordinate (u,v) svih preostalih točaka definiranih bazom
   b. Nakon odgovarajuće kvantizacije koordinata (u,v), iskoristi kvantizirane koordinate (u_q,v_q) kao ključ tablice raspršenja i umetni u odgovarajući pretinac informaciju o pripadnosti $(model,baza)$, koja sadrži oznaku modela i bazu.

U suštini, definirane su ortonormalne baze koje određuju koordinatni sustav Oxy koristeći par vektora $\overrightarrow{p_x^S} \triangleq \left(\overrightarrow{p_{\mu_1}} - \overrightarrow{p_{\mu_1}}\right)$ i $\overrightarrow{p_y^S} \triangleq Rot_{90}\left(\overrightarrow{p_x^S}\right)$, gdje μ_1 i μ_2 predstavljaju točke koje su korištene za definiranje baze. Za svaki izbor baze, ostale točke p modela M_1 određene su jednadžbom $\vec{p} - \overrightarrow{p_0^S} = u\overrightarrow{p_x^S} + v\overrightarrow{p_y^S}$, gdje je $\overrightarrow{p_0^S} = \left(\overrightarrow{p_{\mu_1}} + \overrightarrow{p_{\mu_2}}\right)/2$ središnja točka između $\overrightarrow{p_{\mu_1}}$ i $\overrightarrow{p_{\mu 2}}$. Skalari u i v su invarijantni na translaciju, rotaciju i skaliranje sustava i njihovom kvantizacijom određuju se ključevi tablice raspršenja. Pomoću ključa (u_q,v_q) dohvaća se podatak o pripadnosti $\left(m,\left(\overrightarrow{p_{\mu_1}},\overrightarrow{p_{\mu_2}}\right)\right)$.

2.2.5. Faza raspoznavanja

Prvi korak faze raspoznavanja identičan je prvom koraku faze pretprocesiranja. Na ulaznoj slici se odredi skup točaka koje sadrže značajke objekata, odaberu se dvije točke koje potom čine bazu koordinatnog sustava, postave se na odgovarajuća mjesta, a sve preostale točke zauzimaju jednoznačno određeno mjesto u stvorenom koordinatnom sustavu. *Upravo se koordinate preostalih točaka koriste kao ključevi za tablicu raspršenja koja je konstruirana u fazi pretprocesiranja. Pomoću jednadžbe* $\vec{p} - \overrightarrow{p_0^S} = u\,\overrightarrow{p_x^S} + v\,\overrightarrow{p_y^S}$ *se odrede kvantizirane koordinate točaka sa ulazne slike. U slučaju da se kvantizirane koordinate neke točke sa ulazne slike podudaraju s nekim ključem tablice raspršenja, svi modeli koje označavaju kvantizirane koordinate sa slike dobivaju „glas“, tj. brojač pristupa tim modelima povećava se za jedan.*

Naime, neki ključ može označavati više različitih modela s pripadnim bazama. Npr. dva različita modela (objekta) mogu za neke određene baze sadržavati istu točku. Prema tome ta točka predstavlja ključ koji označava modele koji tu točku sadrže.

Dakle, za svaki pristup nekom modelu, njegov brojač se uvećava za jedan, tj. broje se pristupi tom modelu. Nakon što se za neki uređeni par baza koji čine dvije točke izlazne slike konstruira novi koordinatni sustav i ostale točke se mapiraju na njega, sve kvantizirane koordinate točaka se mogu podudarati s ključevima tablice raspršenja te tako doprinijeti jedan glas modelu, odnosno modelima koje označavaju. Nakon svih mogućih mapiranja točaka izlazne slike, odnosno svih mogućih kombinacija uređenih parova baza koje čine točke izlazne slike, poznat je broj pristupa svim modelima koji se nalaze u tablici raspršenja. Što je broj pristupa nekom modelu veći, to je veća vjerojatnost da se upravo taj model nalazi na ulaznoj slici. Općenito, smisao ovog postupka *nije* određivanje modela s najvećim brojem pristupa već samo značajno smanjiti broj modela za koje se smatra da se nalaze na ulaznoj slici. To se postiže postavljanjem *praga* pristupanja. Samo oni modeli kojima se pristupilo više puta od zadanog praga ulaze u daljnje razmatranje gdje se vrši transformacija točaka svih modela kandidata i uspoređuje s točkama ulazne slike.

2.2.6. Sažetak faze raspoznavanja:

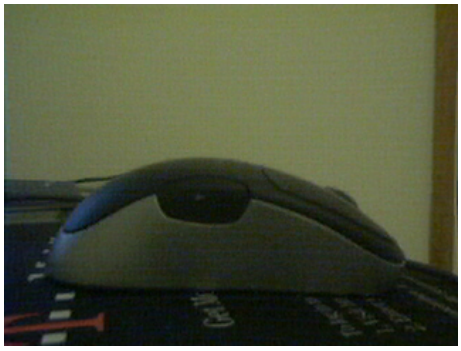
Za svaku ulaznu sliku izvrši sljedeće operacije:

1. Izluči razne interesantne točke (tj. one koje određuju lokaciju značajki koje određuju karakteristike pojedinog objekta). Neka je S skup svih izlučenih točaka, a *S* neka označava kardinalnost skupa S.
2. Odaberi dvije točke koje će stvoriti uređeni par, tj. bazu
3. Odredi koordinate svih ostalih točaka u skladu s odabranom bazom
4. Sve koordinate određene u prethodnom koraku prikladno kvantiziraj. Ako se neka od kvantiziranih koordinata poklapa sa ključem tablice raspršenja konstruirane u fazi pretprocesiranja, povećaj brojač pristupa za sve modele koje označava ta koordinata.
5. Sortiraj sve modele koji su ostvarili barem jedan pristup tokom izvođenja koraka 4. Odredi sve one kojima je pristupljeno više od postavljenog praga, ostale odbaci. Svaki od preostalih modela je kandidat da se upravo on nalazi na ulaznoj slici.
6. Za sve modele kandidate dobivene u prethodnom koraku pronađi transformaciju *T* takvu koja će transformirati koordinate pojedinog modela kandidata tako da se model najbolje moguće poklapa sa objektom koji se nalazi na izlaznoj slici. Kao mjera kvalitete uzima se srednja kvadratna pogreška.
7. Pomoću pronađene transformacije *T* transformiraj sve modele, tj. točke svih modela i usporedi ih s točkama ulazne slike. Kao mjera kvalitete uzima se srednja kvadratna pogreška. U slučaju da je pogreška dovoljno mala, identificiran je objekt sa ulazne slike. U protivnom, idi natrag na korak 2.

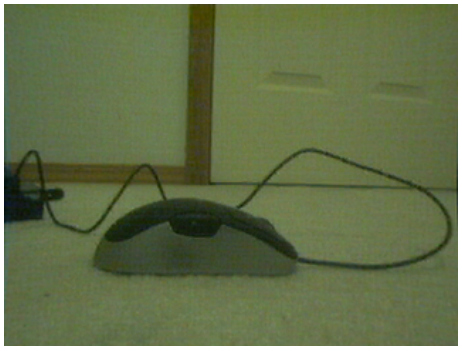
U trodimenzionalnom sustavu raspoznavanja transformacija *T* uključuje translaciju, rotaciju, skaliranje kao i kombinacije svih triju transformacija. U dvodimenzionalnom sustavu raspoznavanja moguće je još primijeniti Afine transformacije kao i dvodimenzionalnu projekciju.

2.3. Primjer traženja objekta na slici

U ovom primjeru potrebno je ispitati nalazi li se određeni objekt na ulaznoj slici.



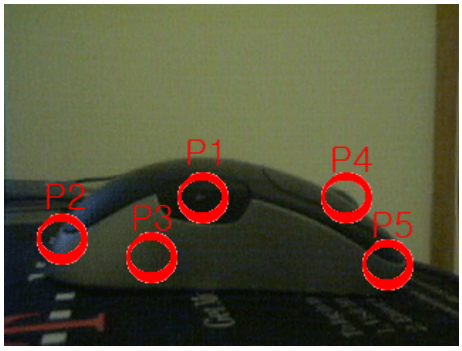
Slika 5. Traženi model - miš



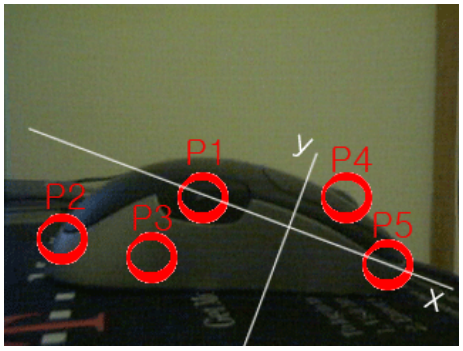
Slika 6. Ulazna slika

2.3.1. Faza pretprocesiranja

Kako bi mogli odrediti ključne točke modela, potrebno je utvrditi o kojim se značajkama radi, odrediti njihovu lokaciju u modelu, te svako pojavljivanje relevantne značajke označiti točkom. Pretpostavimo da je na modelu pronađeno pet točaka sa ključnim značajkama koje jednoznačno opisuju model. Nakon toga je potrebno odabrati sve moguće kombinacije točaka za stvaranje baze. Budući da je određeno pet ključnih točaka modela, ukupan broj mogućih baza je 20.

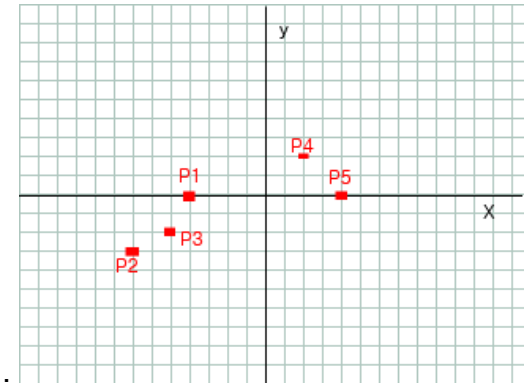


Slika 7. Ključne točke modela



Slika 8. Jedan mogući odabir baze

Za odabranu bazu se stvara koordinatni sustav i određuje koordinate svih ostalih točaka s obzirom na definiranu bazu.

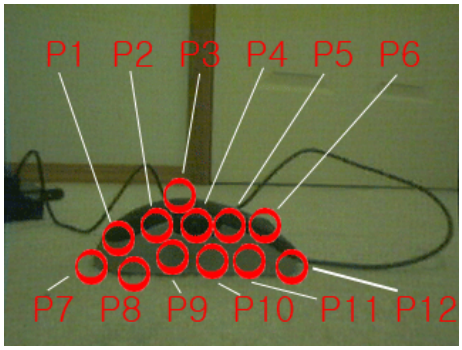


Slika 8. Raspored svih točaka uz definiranu bazu (1,5)

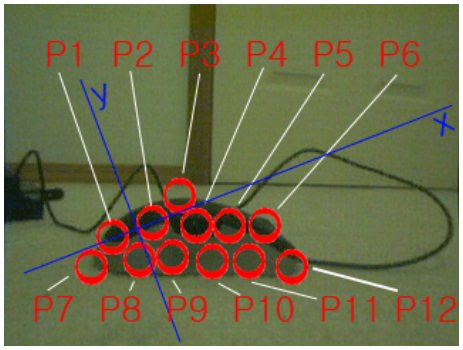
Potrebno je konstruirati položaje točaka za sve moguće odabire baza, a takvih ima 20. Budući da se koordinate baze ne spremaju u tablice raspršenja, za svaki koordinatni sustav postoje tri preostale točke koje ne čine bazu i upravo njihove kvantizirane koordinate čine ključeve tablice raspršenja. Dakle, za 20 mogućih koordinatnih sustava postojat će ukupno 60 ključeva za dani model.

2.3.2. Faza raspoznavanja

Kao i na modelu, potrebno je odrediti ključne točke na ulaznoj slici. Ne temelju ulazne slike sustav mora biti u mogućnosti odrediti ključne značajke objekta sa slike i potom ih predstaviti točkama.

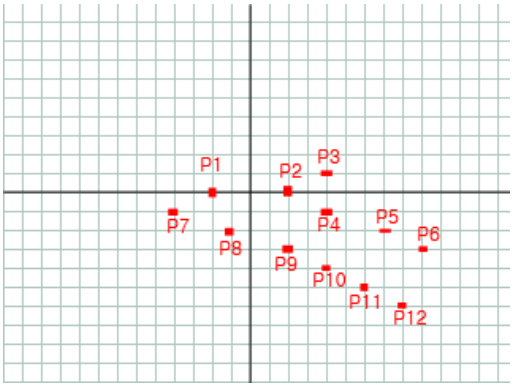


Slika 9. Ključne točke objekta na ulaznoj slici



Slika 10. Jedan mogući odabir baze

Za odabranu bazu se stvara koordinatni sustav i određuje koordinate svih ostalih točaka s obzirom na definiranu bazu.



Slika 11. Raspored svih točaka uz definiranu bazu (1,2)

Kao i kod modela, ovaj prikaz rasporeda točaka ovisi o definiranoj bazi. Budući da je objekt sa slike definiran s ukupno 12 točaka, moguće je odabrati ukupno 132 različite baze. Uloga koordinata sad više nije stvaranje ključeva radi pohrane identifikacije modela, već one služe kao ulaz u tablicu raspršenja konstruiranu u fazi pretprocesiranja. Nakon što se odredila baza i koordinate svih ostalih točaka s obzirom na tu bazu, potrebno je dobivene koordinate kvantizirati. Upravo te kvantizirane koordinate služe kao ulaz u tablicu raspršenja. Općenito, svaka od kvantiziranih točaka može adresirati jedan ili više modela. Budući da za ovaj primjer tablica raspršenja sadrži samo jedan model, kvantizirane koordinate svake točke iz ulazne slike mogu adresirati zadani model. Kad bi tablica raspršenja sadržavala više modela, bilo bi potrebno brojiti pristupe svakom modelu, te nakon toga izabrati one koji su ostvarili broj pristupa veći od zadanog praga. Budući da tablica raspršenja sadrži samo jedan model, navedeni postupak nije potrebno provoditi već je potrebno naći transformaciju koja će transformirati sve točke modela na način da je srednja kvadratna pogreška između objekta sa slike i modela minimalna. U slučaju da je srednja kvadratna pogreška dovoljno mala, model se uistinu nalazi na slici. U protivnom je potrebno izabrati sljedeću bazu u skupu točaka objekta sa slike i ponoviti cijeli postupak. Ako se niti nakon odabira svih mogućih baza ne dobije zadovoljavajuće mala pogreška, model se ne nalazi na slici.

2.4. Primjena geometrijskog raspršenja u prianjanju proteina

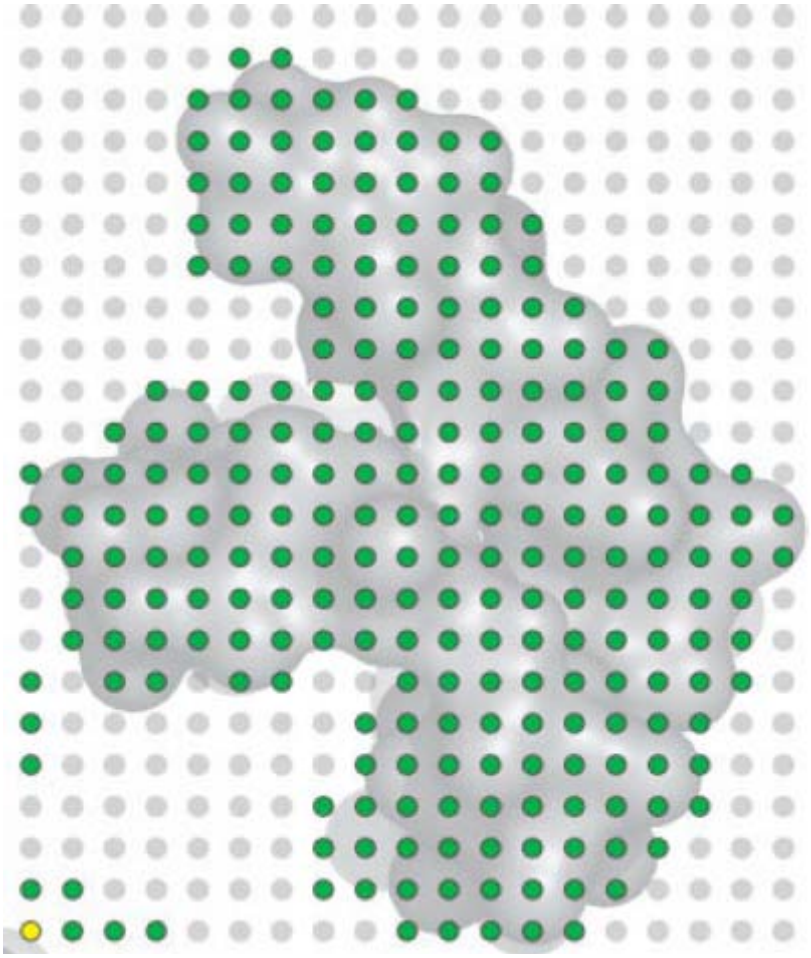
Kao što je pokazano, u računalnom je vidu potrebno usporediti dva objekta na slici koji su predstavljeni skupom točaka. Što je veća podudarnost točaka koje opisuju objekt, veća je vjerojatnost da se traženi objekt nalazi na slici. Problem prianjanja proteina je nešto drugačiji u tome što se ne traži identičnost dijelova dvaju proteina, već njihova komplementarnost. Dakle, poželjna situacija nastupa u slučaju kad se *izbočina* prvog proteina podudara s *udubljenjem* drugog proteina.

Proteini su u računalu predstavljeni skupom voksela (trodimenzionalnih piksela) u trodimenzionalnom sustavu. Trodimenzionalnost upućuje na daleko veću složenost algoritma nego što je to bio slučaj raspoznavanja objekta na dvodimenzionalnoj slici. Ipak, moguće je prostor pretraživanja reducirati na način da se prije samog postupka prianjanja odrede područja vezanja na način da se na svakom od proteina odrede područja izbočina i udubljenja.

Vokseli koji se nalaze u prostoru mogu ali ne moraju pripadati proteinu. Kako bi se protein uopće prikazati pomoću voksela, potrebno je odrediti njihovu rezoluciju, tj. gustoću u prostoru. Nakon što je trodimenzionalni prostor diskretiziran, sastoji se od niza voksela. Na temelju PDB-ova se za svaki od voksela određuje pripada li proteinu. Na tak način se u prostoru nalaze vokseli koji pripadaju proteinu i tzv. prazni vokseli. Postoje dva specifična slučaja:

- u slučaju da je skupina voksela koja pripada proteinu okružena sa zadanim pragom ili *manje* praznih voksela, dio proteina kojeg čine promatrani vokseli označava se kao *udubljenje*.
- u slučaju da je skupina voksela koja pripada proteinu okružena sa zadanim pragom ili *više* praznih voksela, dio proteina kojeg čine promatrani vokseli označava se kao *izbočina.*

Nakon što su određene regije udubljenja i izbočina za oba proteina, područje pretraživanja se svodi na određivanje komplementarnosti izbočina prvog i udubina drugog proteina te obratno. Na temelju lokalnih značajki kao što je zakrivljenost određuju se ključne točke za geometrijsko raspršenje.



Slika 12. Izbočine i udubljenja. Sa slike se lako uočava da su regije udubljenja okruženije vokselima koji čine protein. S druge strane, regije izbočina su okruženije praznim vokselima.

3. Monte Carlo Markovljevi lanci

Simulacija Monte Carlo Markovljevih lanaca (*MCMC simulation*) predstavlja još jednu alternativu iscrpnom pretraživanju prostora. Glavna karakteristika MCMC simulacije jest stohastički pristup, odnosno uporaba slučajnih brojeva u rješavanju različitih problema. Kako bi se najlakše shvatila bit Monte Carlo simulacije, navodi se primjer rješavanja jednostavnog problema. Pretpostavimo da želimo naći vjerojatnost da u grupi od trideset ljudi barem dvoje ima rođendan na isti dan. U daljnjem tekstu se navodi klasični (precizan, analitički) te Monte Carlo pristup.

3.1. Problem zajedničkih rođendana - klasični pristup

Najbolje je krenuti od suprotnog problema, tj. postavlja se pitanje kolika je vjerojatnost da u grupi od 30 ljudi ne postoji niti jedan par osoba kojima rođendan pada na isti dan u godini (pretpostavimo da godina ima 365 dana):

- Promatra li se samo prva osoba, ona može imati rođendan na bilo koji dan te vjerojatnost da niti jedan par osoba nema rođendan na isti dan još uvijek iznosi 1.
- Uzme li se u obzir još jedna, druga osoba, vjerojatnost da ona i prva osoba iz prethodnog razmatranja nemaju rođendan na isti dan iznosi 364/365. Dakle, ako druga osoba ima rođendan na bilo koji dan, a da to nije dan na koji rođendan ima prva osoba, njihovi rođendani se neće podudarati. Očito, postoji 364 od 365 mogućih dana za koje se njihovi rođendani neće preklapati.
- Treća odabrana osoba može imati rođendan na isti dan kao i prva ili pak druga osoba. Dakle, da ne bi došlo do preklapanja rođendana ona ne smije imati rođendan na isti dan kao prva ili druga osoba što znači da mora imati rođendan na jedan od preostalih 365 – 2 = 363 dana. Nakon uzimanja u obzir treće osobe, uvjet na ne postojanje preklapanja rođendana se proširuje te sad vjerojatnost da niti jedna osoba (od njih mogućih tri) nema rođendan na isti dan iznosi (364/365)*(363/365).

- Analogno, uzimamo li redom osobu po osobu, vjerojatnost da u *n* osoba ne postoji niti jedan par koji ima rođendan na isti dan iznosi (364/365)*(363/365)*...((365-n+1)/365).

- Iz navedenih razmatranja slijedi da vjerojatnost da niti jedan par u skupu od 30 osoba nema rođendan na isti dan iznosi (364/365)*(363/365)*...((365-30+1)/365) = (364/365)*(363/365)*...(336/365) = 0,2937. Suprotno, vjerojatnost da barem dvije osobe u grupi od 30 ljudi ima rođendan na isti dan iznosi 1-0,2937 = 0,7063.

3.2. Problem zajedničkih rođendana - Monte Carlo pristup

Rješenje problema je veoma jednostavno, pa je stoga opisano samo algoritamski:

1. Za svaku od 30 osoba izaberi jedan slučajni cijeli broj koji predstavlja rođendan pojedine osobe.
2. Provjeri postoji li već, tj. je li obrađena osoba koja ima isti rođendan kao promatrana. Čim se naiđe na jedan takav slučaj, dalje nije potrebno provjeravati jer je tada sigurno da barem dvije osobe imaju rođendan na isti dan. Povećaj broj podudaranja rođendana za jedan.
3. Ponovi korake 1 i 2 još *M*-1 puta, gdje *M* predstavlja broj epoha, tj. ukupni broj izvođenja pokusa.
4. Nakon *M* epoha odredi kvocijent podudaranja rođendana i ukupnog broja epoha *M.* Izračunati kvocijent tada predstavlja procjenu tražene vjerojatnosti.

Prema zakonu velikih brojeva, što je broj epoha veći, omjer «uspješnih» epoha, tj. onih u kojima je došlo do podudaranja rođendana dvaju osoba i ukupnog broja epoha teži k stvarnoj vjerojatnosti.

Tablica 1. Ovisnost procijenjene vjerojatnosti o broju epoha.

Procijenjena vjerojatnost (p) u odnosu na broj epoha (M) [stvarna vjerojatnost = 0,7063]							
M	1	5	10	100	1000	10000	100000
p	1.0000	0.8000	0.6000	0.6300	0.7150	0.7019	0.7042

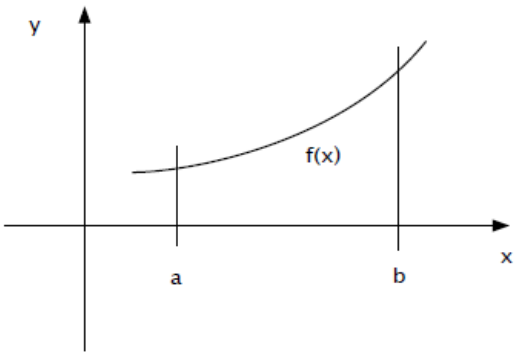
Iz tablice je vidljivo da za velik broj epoha procijenjena vjerojatnost dobro opisuje stvarnu vjerojatnost. Dakle, uz korištenje slučajnih brojeva moguće je dovoljno dobro procijeniti traženu vrijednost bez eksplicitne analitičke formulacije problema.

3.3. Problem podudaranja rođendana – Matlab kod

```
brEpoha = input('Unesi broj epoha: ');
brOsoba = input('Unesi broj osoba: ');

podudaranja = 0;  % broj "uspješnih" epoha, tj. onih u kojima vrijedi da barem
                  % dvije osobe imaju rođendan na isti dan

for epoha = 1 : brEpoha  % brEpoha == broj pokusa

    obradjeni = [];  % polje koje pohranjuje rođendane promatranih osoba

    for osoba = 1 : brOsoba            % promatra se svaka od osoba
        dan = floor( rand*365 + 1 );   % slucajno se za svaku osobu odabire njen rođendan

        if ( ~isempty( find( obradjeni == dan ) ) ) % u slučaju da već neka osoba ima isti rođendan

            podudaranja = podudaranja + 1;  % epoha je "usješna" jer sigurno barem dvije osobe imaju
            break;   % rođendan na isti dan pa dalje nije potrebno provjeravati

        end

        obradjeni(end+1) = dan;  % pamte se rođendani obrađenih osoba

    end

end

fprintf('\nVjerojatnost da barem jedan par od %d osoba ima rođendan na isti dan je %f\n',...
          brOsoba, podudaranja/brEpoha);
```

3.4. Računanje integrala pomoću Monte Carlo metode

Veoma ilustrativan i jedan od najpoznatijih primjera Monte Carlo metode jest računanje integrala. Pretpostavimo da želimo izračunati integral funkcije $f(x)$ na intervalu $[a,b]$.



Slika 13. Funkcija poprima pozitivne vrijednosti na intervalu $[a,b]$

3.4.1. Numerički pristup

Integral funkcije jednak je površini ispod funkcije. Također, integral funkcije $f(x)$ jednak je umnošku duljine intervala $[a,b]$ i srednje vrijednosti funkcije $f(x)$:

$$\int_a^b f(x)dx = (b-a)\, f_{sred} = (b-a)\langle f \rangle \tag{3-1}$$

Dakle, kako bi izračunali integral, potrebno je procijeniti srednju vrijednost kontinuirane funkcije $f(x)$. U tu svrhu koristi se generator slučajnih brojeva. Budući da se integral računa na intervalu $[a,b]$, potreban je generator slučajnih brojeva x_i koji će posjedovati uniformnu distribuciju na tom istom intervalu. Pomoću takvog generatora moguće je procijeniti srednju vrijednost funkcije $f(x)$ pomoću:

$$\langle f \rangle_N = \frac{1}{N}\sum_{i=1}^{n} x_i \tag{3-2}$$

i vrijedi $\lim_{N\to\infty}\langle f\rangle_N = \langle f\rangle$. Očito, odabere li se veći *N*, bolja će biti procjena srednje vrijednosti funkcije $f(x)$. Sada se integral funkcije može procijeniti numerički pomoću procijene srednje vrijednosti funkcije:

$$\int_a^b f(x)dx \simeq (b-a)\langle f\rangle_N = \frac{b-a}{N}\sum_{i=1}^{n} x_i \tag{3-3}$$

3.4.2. Monte-Carlo pristup

Alternativa numeričkom izračunu integrala jest Monte Carlo integracija koja se kao i problem zajedničkih rođendana temelji na korištenju slučajnih brojeva. Kako bi se sam pristup lakše shvatio, dobro je prisjetiti se geometrijske vjerojatnosti. Npr. pretpostavimo da se nasumice bira točka unutar kvadrata površine *P* koji u sebi sadrži manji kvadrat površine *P/4*. Prema geometrijskoj vjerojatnosti lako je vidjeti da je vjerojatnost da se nasumično odabrana točka nađe unutar manjeg kvadrata jednaka omjeru površina manjeg i većeg kvadrata. U konkretnom slučaju tražena vjerojatnost iznosi 1/4.

Pretpostavimo sad da nam površina manjeg kvadrata nije poznata. Također, pretpostavimo da posjedujemo generator slučajnih brojeva s uniformnom razdiobom u području velikog kvadrata te da pritom za svaku točku znamo je li pala unutar ili izvan malog kvadrata. Na temelju slučajno odabranih točaka moguće je procijeniti vjerojatnost se slučajno odabrana točka nađe unutar manjeg kvadrata:

$$p \approx \frac{M}{N} \tag{3-4}$$

gdje je *M* broj točaka unutar manjeg kvadrata, a *N* ukupni broj slučajno odabranih točaka. Procjena vjerojatnosti *p* biti će to bolja što je broj ukupno slučajno odabranih točaka *N* veći. Budući da je prema geometrijskoj vjerojatnosti vjerojatnost *p* jednaka omjeru površina manjeg i većeg kvadrata:

$$p = \frac{S_m}{S_v} \tag{3-5}$$

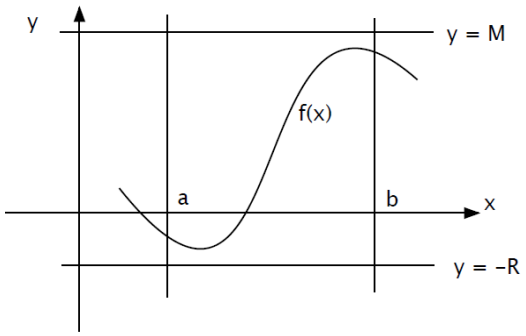
moguće je nepoznatu površinu S_m odrediti uz pomoć procijenjene vjerojatnosti *p* kao

$$S_m = pS_v \approx \frac{M}{N} S_v \tag{3-6}$$

Analogno, ovaj se pristup može iskoristiti za računanje određenih integrala. Pritom je potrebno poznavati funkciju $f(x)$ čiji se integral traži, interval integracije $[a,b]$ te referentnu površinu uz pomoć koje se na temelju procijenjene vjerojatnosti računa traženi određeni integral. Referentna površina je uvijek pravokutnik površine $(a+b)(|M|+|R|)$, tj. umnožak intervala integracije i zbroja minimalne i maksimalne granice unutar kojih se nalaze sve vrijednosti funkcije koja se integrira.

Procjena vjerojatnosti vrši se pomoću sljedećih koraka:

1. odredi vrijednost M takvu da vrijedi $f(x) \leq M$ na intervalu $[a,b]$
2. odredi vrijednost R takvu da vrijedi $f(x) \geq -R$ na intervalu $[a,b]$
3. inicijaliziraj $S = 0$
4. *N* puta izvrši korake 5. - 7.
5. odaberi slučajni broj x iz intervala $[a,b]$
6. odaberi slučajni broj y iz intervala $[-R,M]$
7. ako vrijedi $y \leq f(x)$ povećaj S za jedan



Slika 14. Funkcija koja poprima pozitivne i negativne vrijednosti na intervalu [a,b]

Na temelju ukupnog broja točaka S, tj. točaka (x,y) koje su "pale" u područje ispod pozitivnog dijela funkcije $f(x)$, odnosno iznad negativnog dijela, moguće je odrediti procjenu integrala funkcije. Naime, pomoću broja točaka S i ukupnog broja slučajno

odabranih točaka, a takvih je izračunato ukupno *N*, moguće je procijeniti omjer površina ispod krivulje i ukupnog kvadratnog područja svih mogućih vrijednosti točaka, dakle omjer između integrala funkcije $f(x)$ i površine pravokutnika koja predstavlja skup svih mogućih točaka:

$$\frac{S}{N} \simeq \frac{\text{područje ispod funkcije } f(x)}{\text{ukupno kvadratno područje}} = \frac{\int_a^b f(x)dx}{(M-R)(b-a)} \tag{3-7}$$

iz čega slijedi:

$$\int_a^b f(x)dx = (M-R)(b-a)\frac{S}{N} \tag{3-8}$$

Budući da je kao donja granica pri računanju točaka koje su pale ispod funkcije $f(x)$ uzeta vrijednost *-R ,* a ne *x*-os (funkcija $f(x)$ sadrži i negativne vrijednosti), potrebno je od izračunatog integrala oduzeti kvadratno podruje ispod nule koje iznosi $R(b-a)$ pa je konačan izraz za određeni integral funkcije $f(x)$ koja je ograničena odozgora s *M* i odozdol s *-R* na intervalu $[a,b]$ jednak:

$$\int_a^b f(x)dx = (M-R)(b-a)\frac{S}{N} - R(b-a) \tag{3-9}$$

Prema zakonu velikih brojeva, što je broj slučajno odabranih točaka *N* veći, integral funkcije i površina područja svih mogućih točaka bolje su opisani (pomoću više točaka) pa je i procjena integrala bolja.

Kao primjer, pokažimo Monte-Carlo integraciju funkcije $f(x) = \exp(-x^2)$ na intervalu $[0,10]$.

Tablica 2. Ovisnost procijenjene vjerojatnosti integrala o broju slučajno odabranih točaka.

Procijenjena vjerojatnost integrala (I) u odnosu na broj sluč. odabranih točaka (N) [stvarna vjerojatnost integrala = 0,8862]							
N	1	5	10	100	1000	10000	100000
I	0.0000	2.0000	2.0000	0.7000	1.0200	0.9210	0.8832

3.5. Računanje integrala funkcije - Matlab kod

```
function I = mcIntegracija ( f, x, a, b, M, R, N )

    % M >= max( f(x) ) na intervalu [ a, b ]
    % R <= min( f(x) ) na intervalu [ a, b ]
    % N = broj iteracija, tj. racunanja slucajnih tocaka
    %   -> sto je N veci, koristi se veci skup tocaka za reprezentaciju
    %   povrsina pa je rezultat precizniji

    S = 0;

    for i = 1 : N

            x_ = a + rand * ( b - a ) ;
            y_ = -R + rand * ( M + R );

            if y_ <= subs( f, x, x_ )
                S = S + 1;
            end

    end

    I = ( M + R )*( b - a )* (S/N) - R*( b - a );

end
```

3.6. Markovljevi lanci

Neka je $\mathrm{X} = \{\mathrm{X}_n\}_{n=0}^{\infty}$ niz koreliranih slučajnih varijabli, gdje svaka varijabla X_n može poprimiti vrijednost iz diskretnog prostora stanja Ω. Za slučajni proces X kaže se da je *Markovljev lanac* ako zadovoljava Markovljev uvjet:

$$P\left(\mathrm{X}_{n+1} = j \mid \mathrm{X}_n = i, \mathrm{X}_{n-1} = x_{n-1,\ldots,} \mathrm{X}_0 = x_0\right) = P\left(\mathrm{X}_{n+1} = j \mid \mathrm{X}_n = i\right) \tag{3-7}$$

gdje *P* označava uvjetnu vjerojatnost prijelaza iz stanja i u *n*-tom koraku u stanje j u (*n*+1)-om koraku. Gore navedeni uvjet kaže da uvjetna vjerojatnost prijelaza između dva susjedna stanja u lancu ne ovisi o načinu na koji je lanac došao do određenog stanja iz početnih uvjeta. Markovljev lanac je u potpunosti određen raspodjelom vjerojatnosti početnog stanja $P\left(\mathrm{X}_0 = i, i \in \Omega\right)$ i uvjetnom vjerojatnošću prijelaza između dva stanja u svakom promatranom koraku $P\left(\mathrm{X}_{n+1} = j \mid \mathrm{X}_n = i\right)$.

Ako je prostor stanja Ω diskretan i konačan, uvjetne vjerojatnosti prijelaza stanja mogu se opisati matricom prijelaza P s elementima:

$$P_{ij} = P(\mathrm{X}_{n+1} = j \mid \mathrm{X}_n = i). \tag{3-8}$$

U slučajnim procesima čije stanje ovisi o N_p varijabli, trenutno stanje u *n*-tom koraku opisuje se *vektorom stanja* $\Theta^n \in \mathbb{R}^{N_p}$. Raspodjela pojedinih komponenti vektora stanja Θ u svakom koraku *n* opisuje se višedimenzijskom vektorskom funkcijom gustoće vjerojatnosti $\pi^{(n)}\left(\Theta\right)$:

$$\pi^{(n)}\left(\Theta\right) = \left[\pi_1^{(n)}\left(\Theta_1\right), \pi_2^{(n)}\left(\Theta_2\right), \ldots, \pi_{N_p}^{(n)}\left(\Theta_{N_p}\right)\right], \tag{3-9}$$

gdje svaki $\pi_i^{(n)}$ predstavlja gustoću vjerojatnosti komponente vektora stanja Θ_i u *n*-tom koraku simulacije. Ponašanje multivarijantnog lanca može se zapisati u kompaktnijem obliku korištenjem matrice prijelaza stanja:

$$\pi^{(n)} = \pi^{(n-1)}\mathrm{P}. \tag{3-10}$$

3.6.1. Specijalni slučajevi Markovljevih lanaca

a) *stacionarnost* - Markovljev proces je stacionaran ako vrijedi:

$$\pi = \pi \mathrm{P}\,, \tag{3-11}$$

b) *ergodičnost* - ergodički Markovljev proces je onaj za kojeg vrijedi:

$$n \to \infty \Rightarrow \pi^{(n)} \to \pi, \forall \Theta^{(0)}, \forall \pi^{(0)}\,, \tag{3-12}$$

odnosno ergodički proces asimptotski teži ka stacionarnoj distribuciji bez obzira na polazni, početni vektor stanja. Ergodički procesi izuzetno su važni u praktičnim primjenama zbog cijelog niza dobrih matematičkih svojstava, poput stacionarnosti i mogućnosti dobivanja statističkog očekivanja vremenskim usrednjavanjem realizacije slučajnog procesa.

c) *homogenost* - ako vrijedi da je $\pi^{(n)} = \pi^{(n+k)}, \forall k \in \mathbb{Z}$, tj. ako vjerojatnost prijelaza između stanja ne ovisi o koraku simulacije.

d) *ireducibilnost* - lanac je ireducibilan ako je moguć prijelaz između svakog odabranog para diskretnih stanja

e) *reverzibilnost* - lanac je reverzibilan ako je u svakom trenutnom stanju lanca moguć povratak u stanje iz prethodnog koraka.

Nužan i dovoljan uvjet da bi homogen, ireducibilan i reverzibilan Markovljev lanac bio ergodičan u literaturi se naziva *detailed balance condition*:

$$\pi_i P_{ij} = \pi_j P_{ji}\,. \tag{3-13}$$

U praksi se često pojavljuje problem simulacije višedimenzionalnog ergodičnog slučajnog procesa, čija su statistička svojstva opisana vektorskom funkcijom gustoće vjerojatnosti π, a bez eksplicitno poznate pripadne matrice prijelaza P. Nadalje, ponašanje Markovljevih lanaca kod procesa na beskonačnim ili neprebrojivim (npr. kontinuiranim) prostorima stanja Ω ne može se zapisati matricom prijelaza P konačnih dimenzija. U takvim slučajevima ponašanje Markovljevih lanaca ne opisuje

se matricom prijelaza P, već algoritamskim opisom prijelaza stanja. U tu svrhu tipično se koristi neki od dva osnovna simulacijska algoritma: *Metropolis-Hastings* algoritam i *Gibbs sampler.*

3.7. Metropolis-Hastings algoritam

Metropolis-Hastings algoritam je nastao iz praktične potrebe za numeričkim integriranjem složenih analitičkih izraza koji se pojavljuju u različitim područjima teorijske fizike, a nisu rješivi analitički ili primjenom uobičajenih postupaka numeričke integracije. Osnovna ideja pristupa jest predstaviti podintegralnu funkciju gustoćom vjerojatnosti, prema kojoj se generira veliki broj uzoraka čijom se statističkom obradom procjenjuje numerička vrijednost određenog integrala. Iako je izvorno zamišljen za rješavanje problema numeričke integracije, mogućnosti primjene Metropolis-Hastings algoritma daleko su šire, odnosno može se koristiti u svim problemima kod kojih je potrebno simulirati uzimanje uzoraka iz proizvoljno zadane funkcije gustoće vjerojatnosti.

Neka je $\pi(\Theta), \Theta \in \Omega \subseteq \mathbb{R}$ gustoća vjerojatnosti prema kojoj se žali obaviti generiranje skupa slučajnih uzoraka. Ergodički Markovljev lanac koji simulira gustoću vjerojatnosti $\pi(\Theta)$ može se dobiti Metropolis-Hastings algoritmom na sljedeći način:

1. Korak **inicijalizacije** algoritma - odabrati početno stanje $\Theta^{(0)} \in \Omega$ tako da vrijedi $\pi\left(\Theta^{(0)}\right) > 0$.
2. Korak **generiranje** kandidata - generirati novo stanje $\Theta^{(n+1)}$ iz postojećeg stanja $\Theta^{(n)}$ korištenjem prijelazna uvjetne gustoće vjerojatnosti $g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right)$ koja mora zadovoljavati:

   a) $g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right) \neq 0 \Rightarrow g\left(\Theta^{(n)} \mid \Theta^{(n+1)}\right) \neq 0$ (uvjet reverzibilnosti)

   b) $g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right)$ je ireducibilna na Ω.

3. Korak prihvaćanja kandidata - izračunati vjerojatnost α prihvaćanja novog stanja:

$$\alpha\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right) = \min\left\{1, \frac{\pi\left(\Theta^{(n+1)}\right) g\left(\Theta^{(n)} \mid \Theta^{(n+1)}\right)}{\pi\left(\Theta^{(n)}\right) g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right)}\right\}. \tag{3-14}$$

Odluku o prihvaćanju ili odbijanju novog stanja $\Theta^{(n+1)}$ najlakše je implementirati generiranjem slučajnog broja u iz uniformne raspodjele $U[0,1]$:

$$u \sim U[0,1] \Rightarrow \begin{cases} u \leq \alpha, & \text{stanje } \Theta^{(n+1)} \text{ se prihvaća,} \\ u > \alpha, & \text{stanje } \Theta^{(n+1)} \text{ se odbija.} \end{cases} \tag{3-15}$$

Ako se novo stanje $\Theta^{(n+1)}$ ne prihvati, lanac ostaje u stanju $\Theta^{(n)}$ u (*n+1*)-om koraku simulacije. Nakon odluke o prihvaćanju ili odbijanju novog stanja, algoritam se vraća na korak generiranja novog kandidata. Navedenom formulom za prihvaćanje kandidata preferira se dulje zadržavanje lanca u područjima veće vjerojatnosti, čineći tako pretraživanje prostora vjerojatnosti Ω učinkovitim.

Opisani jednodimenzionalni Metropolis-Hastings algoritam moguće je poopćiti na probleme s proizvoljnim brojem dimenzija vektora stanja $\Theta \in \mathbb{R}^{N_p}$, gdje $\pi(\Theta)$ tada predstavlja *zajedničku* gustoću vjerojatnosti svih varijabli stanja. U tom slučaju postoji više načina generiranja novih stanja lanca: istovremena promjena svih varijabli stanja sustava, slijedna djelomična promjena vektora stanja (blokovsko uzorkovanje vektora stanja) ili pojedinačna promjena svake varijable stanja, pri čemu se primjenjuje ciklička shema uzorkovanja svakog bloka ili varijable stanja.

Prednost Metropolis-Hastings algoritma leži u činjenici da se zbog načina pretraživanja prostora vjerojatnosti Ω može vrlo učinkovito obaviti pretraživanje visoko dimenzionalnih prostora stanja, izbjegavajući pritom potrebu za pretraživanjem golemih dijelova prostora koji nisu od interesa, tj. onih u kojima je $\pi(\Theta) \approx 0$. Drugo vrlo bitno svojstvo ovog algoritma jest činjenica da je funkciju $\pi(\Theta)$ dovoljno poznavati analitički samo djelomično, odnosno točno do na normalizacijsku

konstantu. Naime, da bi funkcija $\pi(\Theta)$ bila gustoća vjerojatnosti, mora biti zadovoljen uvjet $\int_{\Omega} \pi(\Theta) d\Theta = 1$. Kada to nije slučaj, izraz $\pi(\Theta)$ je potrebno pomnožiti normalizacijskom konstantom k. Proračun normalizacijske konstante kod visoko dimenzionalnih problema, pogotovo kada se ciljna gustoća vjerojatnosti $\pi(\Theta)$ formira prema specifičnom problemu i ne predstavlja neku od standardnih distribucija, u praksi je često složen i nemoguć. Međutim, problem proračuna normalizacijske konstante kod Metropolis-Hastings algoritma se u potpunosti izbjegava jer se normalizacijske konstante izraza u brojniku i nazivniku kvocijenta formule(ona duga za alfa) krate, uz pretpostavku da je analitički izraz za uvjetnu gustoću vjerojatnosti $g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right)$ u potpunosti poznat.

3.7.1. Formiranje prijelazne uvjetne gustoće vjerojatnosti

Kod formiranja prijelazne uvjetne gustoće vjerojatnosti $g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right)$ moguća su dva osnovna pristupa:

a) slučajni hod - vrijednost novog stanja se dobiva zbrajanjem trenutnog stanja i slučajne varijable *Z,* određene gustoćom vjerojatnosti π_Z:

$$\Theta^{(n+1)} = \Theta^{(n)} + Z, \qquad Z \sim \pi_Z\left(\lambda_1, ..., \lambda_q\right) \tag{3-16}$$

$$\Theta \sim g\left(\Theta^{(n+1)} \mid \Theta^{(n)}, \lambda_1, ...\lambda_q\right) = f\left(\Theta^{(n)}\right) \tag{3-17}$$

b) nezavisno uzorkovanje - u prijelaznoj funkciji gustoće vjerojatnosti ne uzima se u obzir vrijednost stanja u prethodnoj iteraciji:

$$\Theta^{(n+1)} \sim \pi_0\left(\lambda_1, ..., \lambda_q\right), \tag{3-18}$$

$$\Theta \sim g\left(\Theta^{(n+1)} \mid \Theta^{(n)}, \lambda_1, ...\lambda_q\right) \neq f\left(\Theta^{(n)}\right) \tag{3-19}$$

$\left\{\lambda_1, ...\lambda_q\right\}$ predstavlja skup nekih *a priori* odabranih parametara prijelazne gustoće vjerojatnosti. Odabir gustoća vjerojatnosti π_Z i π_0 za formiranje $g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right)$ je proizvoljan, a tipičan odabir predstavljaju standardne, jednostavne raspodjele (npr. uniformna, Gaussova itd.), kod kojih se ne pojavljuje problem proračuna normalizacijske konstante. Primjerice, ako se za π_Z kod slučajnog hoda odabere

Gaussova raspodjela, tada prijelazna gustoća vjerojatnosti ima oblik normalne raspodjele oko trenutne vrijednosti $\Theta^{(n)}$, s rasipanjem određenim parametrom λ. Slično vrijedi i za neovisno uzorkovanje, no tada se Gaussova raspodjela centrira oko neke *a priori* odabrane vrijednosti, a ne trenutnog stanja lanca $\Theta^{(n)}$.

Općenito, kada se odabere simetrična prijelazna gustoća vjerojatnosti $g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right)$ za koju vrijedi:

$$g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right) = g\left(\Theta^{(n)} \mid \Theta^{(n+1)}\right) \tag{3-20}$$

postupak simulacije $\pi(\Theta)$ se pojednostavljuje u *Metropolis* algoritam, kod kojega se prijelazne gustoće vjerojatnosti $g\left(\Theta^{(n+1)} \mid \Theta^{(n)}\right)$ i $g\left(\Theta^{(n)} \mid \Theta^{(n+1)}\right)$ mogu zanemariti prilikom proračuna koeficijenata prihvaćanja (opet ona duga formula).

3.8. Gibbs sampler

Gibbs *sampler* predstavlja poseban slučaj Metropolis-Hastings algoritma kod kojega je vjerojatnost prihvaćanja novog stanja α = 1. Osnovna zamisao Gibbs *samplera* jest pojednostavljenje postupka generiranja novih stanja kod višedimenzionalnih problema, tako da se u svakom koraku obavlja uzorkovanje jednodimenzionalne gustoće vjerojatnosti umjesto višedimenzionalne, kako je to bio slučaj kod Metropolis-Hastings algoritma. Za razliku od Metropolis-Hastings algoritma, Gibbs *sampler* je primjenjiv isključivo na višedimenzionalne probleme.

Gibbs *samplerom* također se simulira ergodički Markovljev lanac određen višedimenzionalnom gustoćom vjerojatnosti $\pi(\Theta)$. Algoritam se može opisati sljedećim koracima:

1. Korak **inicijalizacije** algoritma - potrebno je proizvoljno odabrati početni vektor stanja $\Theta^{(0)} \in \Omega,\ \pi\left(\Theta^{(0)}\right) > 0$.
2. Korak **generiranja** novog stanja - generirati novo stanje sustava $\Theta^{(n+1)}$ kroz N_p sekvencijalnih promjena pojedinih komponenti vektora stanja,

uzorkovanjem potpunih uvjetnih gustoća vjerojatnosti $\pi\left(\Theta_k \mid \vec{\Theta}_{-k}\right)$ prema sljedećem algoritmu:

$$\Theta_1^{(n+1)} \sim \pi\left(\Theta_1 \mid \Theta_2^{(n)}, \Theta_3^{(n)}, ..., \Theta_{N_p}^{(n)}\right)$$

$$\Theta_2^{(n+1)} \sim \pi\left(\Theta_2 \mid \Theta_1^{(n+1)}, \Theta_3^{(n)}, ..., \Theta_{N_p}^{(n)}\right)$$

...

$$\Theta_k^{(n+1)} \sim \pi\left(\Theta_k \mid \Theta_1^{(n+1)}, ..., \Theta_{k-1}^{(n+1)}, \Theta_{k+1}^{(n)}, ..., \Theta_{N_p}^{(n)}\right)$$

...

$$\Theta_{N_p}^{(n+1)} \sim \pi\left(\Theta_{N_p} \mid \Theta_1^{(n+1)}, \Theta_2^{(n+1)}, ..., \Theta_{N_p-1}^{(n+1)}\right) \qquad \text{(3-21)}$$

U svakom međukoraku se obavlja uzorkovanje jednodimenzionalne funkcije potpune uvjetne gustoće vjerojatnosti komponente Θ_k, uvjetovane na trenutno stanje vektora preostalih $\left(N_p - 1\right)$ komponenti $\vec{\Theta}_{-k}$ (uzimaju se vrijednosti $\Theta_i^{(n+1)}$ iz trenutne iteracije za komponente $i < k$, odnosno $\Theta_i^{(n)}$ iz prethodne iteracije za $i > k$). Iteracija je završena tek kada se obavi uzorkovanje svih N_p komponenti vektora stanja, pri čemu se nova parcijalna stanja komponenti u svakom međukoraku bezuvjetno prihvaćaju, za razliku od izvornog Metropolis-Hastings algoritma. Osim prikazane deterministička cikličke sheme uzorkovanja potpunih uvjetnih gustoća vjerojatnosti svih komponenti vektora stanja, mogu se primijeniti i drugačiji pristupi (npr. slučajne permutacije redoslijeda uzorkovanja komponenti Θ_k m drugačiji algoritmi ažuriranja vektora stanja $\vec{\Theta}$ i sl.).

3.9. Usporedba Metropolis-Hastings algoritma i Gibbs sampliranja

Implementacija Gibbs *samplera* često je jednostavnija od Metropolis-Hastings algoritma, posebice u slučajevima kada je moguće razdvojiti utjecaj pojedinih varijabli stanja i funkcije potpunih uvjetnih gustoća vjerojatnosti opisati jednostavnim analitičkim izrazima za neku od standardnih raspodjela. Uzorkovanje komponenti vektora stanja Θ_k svodi se na generiranje slučajnih brojeva prema standardnim jednodimenzionalnim distribucijama što je proračunski znatno manje zahtjevan postupak od proračuna izraza za višedimenzionalnu zajedničku gustoću vjerojatnosti $\pi(\Theta)$ kod proračuna koeficijenta prihvaćanja Metropolis-Hastings algoritma.

Također, pristup sekvencijalnog uzorkovanja komponenti vektora stanja podjelom jedna MCMC iteracije u N_p međukoraka povoljno utječe na smanjenje korelacije između susjednih stanja.

Ipak, postoje i određeni nedostaci Gibbs *samplera* u usporedbi s Metropolis-Hastings algoritmom. Jedan od najvažnijih je potreba za potpunim poznavanjem analitičkog oblika funkcija potpunih uvjetnih gustoća vjerojatnosti što uključuje i normalizacijske konstante koje u općem slučaju nije jednostavno odrediti. Naime ukoliko potpune uvjetne gustoće vjerojatnosti nije moguće izraziti nekom od standardnih raspodjela ili nisu raspoloživi pripadni analitički izrazi (tj. potrebno je numerički aproksimirati $\pi\left(\Theta_k \mid \vec{\Theta}_{-i}\right)$ i odrediti pripadajuće normalizacijske konstante), tada implementacija Gibbs *samplera* može postati računalno izrazito zahtjevna. Takvi se problemi mogu pojaviti primjerice kod visoko dimenzionalnih nelinearnih inverznih problema kod kojih nije moguće analitički separirati utjecaj pojedinih varijabli prilikom formiranja $\pi\left(\Theta_k \mid \vec{\Theta}_{-i}\right)$.

3.10. Uporaba Monte Carlo simulacije u prianjanju proteina

Aktualne metoda bazirane na Monte Carlo simulaciji u prianjanju proteina relativno su mlade i pokazuju određene prednosti nad ostalim metodama. Naime, velika većina metoda modelira proteine kao kruta tijela te se pritom čini gruba aproksimacija jer prilikom interakcije proteina nastaju razne konformacije proteina pa novonastale strukture više ne odgovaraju početnim strukturama proteina izvan kompleksa. Razlog tomu su fleksibilnost okosnice i bočnog lanca proteina. Također, u obzir je potrebno uzeti i van der Waalsove sile, vodikove veze, hidrofobnost površina te sile elektrostatske prirode što metode temeljene na Monte Carlo simulaciji uglavnom sve uzimaju u obzir što ih čini daleko složenijim, ali i uspješnijim od ostalih metoda. Program AutoDock baziran na Monte Carlo simulaciji je trenutno najpoznatiji i najuspješniji program u predviđanju proteinskih kompleksa.

4. FFT-metoda kod prianjanja proteina

4.1. Osnovna ideja

FFT (fast Fourier transform) predstavlja metodu iscrpnog pretraživanja prostora. U suštini, sama metoda intuitivno ne govori kako doći do ocjene preklapanja dvaju proteina već ideja leži u kros-korelacijskoj funkciji. Za najjednostavniju ilustraciju, zamislimo dvije funkcije realne varijable, $f(t) = \sin(t)$ i $g(t) = \cos(t)$. Dane funkcije su istog oblika, no funkcija $g(t) = \cos(t)$ je pomaknuta u odnosu na funkciju $f(t) = \sin(t)$ za $\pi / 2$. Njihova korelacija je dana formulom

$$xcorr(t) = \int_{-\infty}^{\infty} f(\tau) g(t+\tau) d\tau = \int_{-\infty}^{\infty} \sin(\tau)\cos(t+\tau) d\tau \qquad (4\text{-}1)$$

te je varirajući parametar t moguće odrediti njihovu maksimalnu korelaciju, odnosno njihova maksimalna korelacija nastupa za $t_{\max}$ koji predstavlja potreban iznos pomaka jedne funkciju u odnosu na drugu kako bi postale identične. Kros-korelacijska funkcija dakle pomiče funkciju $g(t)$ duž vremenske osi te računa integral za svaki odabrani pomak. Pri pomaku za $t = \pi / 2$ funkcije postaju identične te kros-korelacijska funkcija poprima maksimalnu vrijednost. Budući da se u konkretnom primjeru koriste periodičke funkcije, kros-korelacijski integral bi za definicijske granice divergirao pa je za granicu integracije dovoljno uzeti veći period danih funkcija (u ovom primjeru oba iznose 2π) .

4.2. Korelacija površina proteina

Najprije je potrebno površine proteina mapirati na trodimenzionalnu rešetku. Opišemo li površinu prvog proteina funkcijom $P_1(x, y, z)$, drugog proteina funkcijom $P_2(x, y, z)$, izraz za ocjenu preklapanja odgovara upravo diskretnoj trodimenzionalnoj kros-korelacijskoj funkciji površina oba proteina:

$$S(i,j,k) = \sum_{x,y,z} P_1(x+i, y+j, z+k) P_2(x,y,z) \tag{4-2}$$

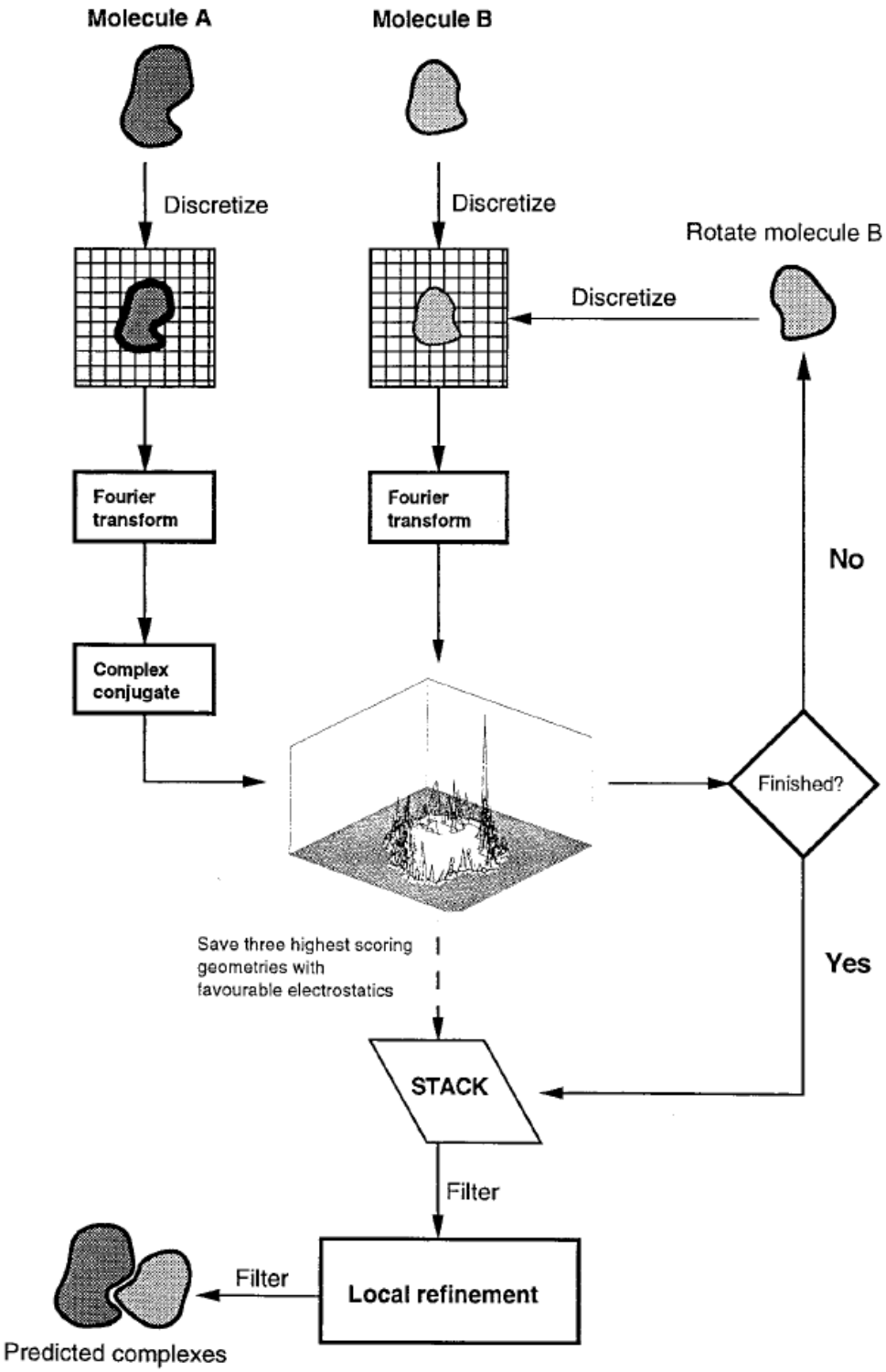
Za računanje korelacije se koristi račun u frekvencijskoj domeni pa se ocjena preklapanja računa kao inverz računa u frekvencijskoj domeni:

$$S(i,j,k) = IDFT\left(\overline{DFT\left(P_1(x,y,z)\right)} \cdot DFT\left(P_2(x,y,z)\right)\right) \tag{4-3}$$

U ovom koraku do izražaja dolazi FFT algoritam pomoću kojeg se računa DFT. Korištenjem FFT algoritma složenost korelacije umjesto $O\left(n^6\right)$ postaje $O\left(n^3 \log(n)\right)$ što mnogo ubrzava proces računanja korelacije. Pierce dolazi do formule koja se koristi u implementacijama navedenog algoritma:

$$S(i,j,k) = \frac{1}{N^3} IDFT\left(IDFT(P_1) \cdot DFT\left(P_2\right)\right). \tag{4-4}$$

Prednost FFT metode je činjenica da se odjednom računa ocjena za sve moguće translatorne pomake proteina. Nedostaci su to što se uvijek računa ocjena za sve točke rešetke, iako za većinu to nije potrebno budući da za velik broj točaka proteini prodiru jedan u drugoga što je fizikalno nemoguće. Također, protein mora biti jednoliko pravokutno otipkan, pa nije moguće npr. površinu proteina otipkati tako da svakom uzorku pripada jednaka površina. Velik nedostatak je i činjenica da je za sve moguće rotacijske pomake potrebno računati novu FFT.



Slika 15. Postupak prianjanja proteina pomoću FFT metode

5. Zaključak

Zanimljiva i obećavajuća činjenica vezana za računalno modeliranje prianjanja proteina leži u raznim mnogobrojnim mogućim matematičkim i računalnim pristupima. Trenutno su najpopularnije tri metode: geometrijsko raspršenje, FFT te Monte Carlo metoda.

Prednost metode geometrijskog raspršenja jest njena brzina i inherentna paralelnost, no u stanju je opisati samo geometrijska svojstva koja su ključna, no ne i dovoljna da bi dobro opisala proteinske interakcije. Također, veliki nedostatak predstavlja generiranje velikog broja fizikalno nemogućih kompleksa (jedan protein prodire u drugoga), koje je potrebno naknadno eliminirati.

FFT metoda je popularna zbog brzog računanja korelacije među površinama ili pak nekog drugog svojstva proteina. Brzina se postiže računanjem korelacije u frekvencijskoj domeni pomoću FFT-a, brze inačice DFT-a koja se može paralelizirati. Također, velika prednost FFT metode je činjenica da se odjednom računa ocjena za sve moguće translatorne pomake proteina. Nedostaci su to što se uvijek računa ocjena za sve točke rešetke, iako za većinu to nije potrebno budući da za velik broj točaka proteini prodiru jedan u drugoga što je fizikalno nemoguće. Velik nedostatak je i činjenica da je za sve moguće rotacijske pomake potrebno računati novu FFT.

Aktualne metoda bazirane na Monte Carlo simulaciji u prianjanju proteina relativno su mlade i pokazuju određene prednosti nad ostalim metodama. Naime, velika većina metoda modelira proteine kao kruta tijela te se pritom čini gruba aproksimacija jer prilikom interakcije proteina nastaju razne konformacije proteina pa novonastale strukture više ne odgovaraju početnim strukturama proteina izvan kompleksa. Razlog tomu su fleksibilnost okosnice i bočnog lanca proteina. Također, u obzir je potrebno uzeti i van der Waalsove sile, vodikove veze, hidrofobnost površina te sile elektrostatske prirode. Metode temeljene na Monte Carlo simulaciji uglavnom sve navedene komponente uzimaju u obzir što ih čini daleko složenijim, ali i uspješnijim od ostalih metoda. Danas se istraživački napor usmjerava prema rješavanju problema fleksibilnosti, budući da je postalo jasno da će značajnija poboljšanja postojećih rezultata uslijediti tek kad se na odgovarajući način uračunaju i konformacijske promjene.

6. Literatura

[1] Wolfson, H.J., Rigoutsos, I., Geometric Hashing: An Owerview. IEEE Computing Science and Engineering, 1997.

[2] Kihara, D., Lee, S., Ramani, K., Turuvekere, S., Agrawal, M., Li, B., Fast Protein-Protein Docking Algorithm Using Pre-Identified Binding Site Patches

[3] Džapo, H., Mjerna metodologija temeljena na modelu s primjenom na ispitivanje uzemljivačkih sustava, doktorska disertacija, Fakultet Elektrotehnike i Računarstva, Zagreb, 2007.

[4] Gray, J.J., Moughon, S., Wang, C., Schueler-Furman, O., Kuhlman, B., Rohl,C.A., Baker, D., Protein-Protein Docking with Simultaneous Optimization of Rigid-body Displacement and Side-chain Conformations, 2003.

[5] Dokmanić, I., Algoritam za predviđanje strukture proteinskih kompleksa korištenjem razvoja površine u redove funkcija, Diplomski rad, Fakultet Elektrotehnike i Računarstva, Zagreb, 2007.

[6] Ritchie, D.W., Recent progress and future directions in protein-protein docking, 2008.

[7] Kochanski, G., Monte Carlo Simulation, 2005.

[8] Gabb, H.A., Jackson R.M., Sternberg, M.J.E., Modelling Protein Docking using Shape Complementarity, Electrostatics and Biochemical Information

7. Sažetak

Definirana su i opisana tri osnovna pristupa u prianjanju proteina: geometrijsko raspršenje, FFT i Monte Carlo metoda. Detaljno se opisuje metoda geometrijskog raspršenja te se navodi jedan mogući pristup njene uporabe u području prianjanja proteina. Unatoč dobrim svojstvima kao što je velika brzina algoritma, budući da promatra isključivo geometrijska svojstva, metoda ima ograničenu uporabu kod modeliranja proteinskih interakcija. Na temelju jednostavnog primjera se pokazuje računanje kroskorelacije dvije funkcije kao mjera njihove podudarnosti te se prikazuje složeniji, trodimenzionalni oblik računanja korelacije površina proteina u frekvencijskoj domeni pomoću FFT-a. Navedeni su osnovni principi Monte Carlo metode te Metropolis-Hastings algoritma, odnosno Gibbs *samplera*. Zbog velikog broja mogućih iskorištavanja Monte Carlo metode kao što je modeliranje fleksibilnosti okosnice i bočnog lanca proteina, od svih metoda upravo Monte Carlo metoda predstavlja najplodnije područje u daljnjem modeliranju proteinskih interakcija, odnosno u prianjanju proteina.