

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1877

# **VIZUALIZACIJA STRUKTURE PROTEINA**

Toni Sente

Zagreb, lipanj 2019.

**SVEUČILIŠTE U ZAGREBU**  
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**  
**ODBOR ZA DIPLOMSKI RAD PROFILA**

Zagreb, 7. ožujka 2019.

Predmet: **Diplomski rad**

**DIPLOMSKI ZADATAK br. 1877**

Pristupnik: **Toni Sente (0036483721)**  
Studij: **Računarstvo**  
Profil: **Programsko inženjerstvo i informacijski sustavi**

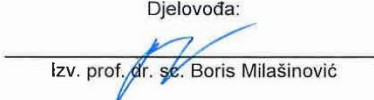
Zadatak: **Vizualizacija strukture proteina**

**Opis zadatka:**


U okviru ovoga diplomskog rada potrebno je proučiti razine strukture proteina: primarnu, sekundarnu, tercijarnu i kvaternu strukturu te formate zapisa strukture proteina. Zatim je potrebno istražiti matematičke modele te grafičke načine prikaza strukture proteina te napraviti program koji podržava prikaz sekundarne i tercijarne strukture proteina. Vlastito rješenje usporediti s postojećim sličnim alatima.

Zadatak uručen pristupniku: 15. ožujka 2019.  
Rok za predaju rada: 28. lipnja 2019.

  
Mentor:  
Doc. dr. sc. Mirjana Domazet-Lošo

  
Djelovođa:  
Izv. prof. dr. sc. Boris Milašinović

Predsjednik odbora za  
diplomski rad profila:

  
Izv. prof. dr. sc. Igor Mekterović

*Zahvaljujem mentorici doc. dr. sc. Mirjani Domazet-Lošo koja je već na 2. godini fakulteta bezuvjetno prihvatila moj zahtjev za mentorstvom. Od srca hvala na strpljivosti, susretljivosti i svojoj pomoći iskazanoj tijekom svih ovih godina studiranja.*

*Veliko hvala i prof. dr. sc. Željki Mihajlović te kolegi mag. ing. comp Marku Jagodiću za pomoć prilikom izrade ovog diplomskog rada.*

*Zahvaljujem se i svim ostalim profesorima na fakultetu koji su uvijek bili otvoreni za diskusiju i spremni pomoći bez obzira na vrstu problema.*

*Veliko hvala i svim prijateljima koji su ovo studiranje učinili nezaboravnim iskustvom.*

*Konačno, ali nikako manje važno, od srca hvala mojoj obitelji, a posebice mojim roditeljima. Hvala Vam na svojoj pomoći, žrtvi i trudu iskazanom tijekom cijelog mog obrazovanja.*

## Sadržaj

1.	Uvod.....	1
2.	Proteini.....	4
2.1.	Primarna struktura .....	5
2.2.	Sekundarna struktura .....	5
2.2.1.	$\alpha$ -zavojnica.....	6
2.2.2.	$\beta$ -naborana ploča .....	6
2.3.	Tercijarna struktura.....	7
2.4.	Kvartarna struktura .....	7
2.5.	Stabilnost proteina.....	7
2.6.	Određivanje 3D strukture proteina.....	8
3.	Zapisi proteina .....	9
4.	Program za vizualizaciju proteina .....	12
4.1.	Kamera.....	13
4.2.	Izrada 3D modela proteina .....	16
4.2.1.	Krivulje .....	17
4.2.2.	Izrada okosnice proteina .....	21
4.2.3.	$\alpha$ -zavojnice.....	23
4.2.4.	$\beta$ -lanci .....	26
5.	Usporedba s postojećim alatima .....	28
6.	Zaključak.....	32
7.	Literatura.....	33
	Sažetak.....	36
	Dodatak A: Upute za uporabu programa .....	38

## **Popis oznaka i kratica**

engl	engleski
API	sučelje za programiranje aplikacija
C $\alpha$	središnji atom ugljika u aminokiselini
PDB	(1) format zapisa proteina ili (2) Protein Data Bank repozitorij

## Popis slika

Slika 1. Prikaz proteina lizozima (253L) sa kuglama i štapićima u alatu Jmol .....	2
Slika 2. Prikaz proteina lizozima (253L) korištenjem vrpce u alatu Jmol .....	2
Slika 3. Aminokiselina .....	4
Slika 4. Stvaranje peptidne veze .....	5
Slika 5. Rotacije u polipeptidnom lancu [4] .....	5
Slika 6. Struktura $\alpha$ -zavojnice [5] .....	6
Slika 7. Struktura $\beta$ -naborane ploče .....	7
Slika 8. Model projekcijske kamere [11] .....	13
Slika 9. Položaj kamere u prostoru .....	14
Slika 10. Rotacija kamere oko središta objekta .....	14
Slika 11. Rotacije oko koordinatnih osi .....	15
Slika 12. Usporedba direktnog spajanja $C\alpha$ atoma (ravna okosnica) te spajanja korištenjem krivulja (zakrivljena okosnica) .....	17
Slika 13. Razlike u C kontinuitetu kod krivulja [13] .....	18
Slika 14. Bezierove krivulje [14] .....	19
Slika 15. Catmull-Rom krivulja .....	20
Slika 16. Prikaz poligona prilikom kreiranja okosnice .....	22
Slika 17. Geometrijski prikaz smjera okosnice .....	22
Slika 18. Usporedba glatkoće cijevi .....	23
Slika 19. Izgled $\alpha$ -zavojnice korištenjem vrpce [6] .....	23
Slika 20. Neke od mogućih orijentacija pravokutnika okomitog na vektor smjera .....	24
Slika 21. Određivanje lokalnog koordinatnog sustava za $C\alpha(i)$ [2] .....	25
Slika 22. Izgled zavojnice dobivene linearnom interpolacijom lokalnih koordinatnih sustava $C\alpha$ atoma .....	26
Slika 23. Ispravljanje $\beta$ -lanca .....	27
Slika 24. Usporedba proteina 6NIV .....	29
Slika 25. Usporedba proteina 5ZSY .....	29
Slika 26. Usporedba proteina 6NUK .....	30
Slika 27. Usporedba proteina 1A3N .....	31

## **Popis tablica**

Tablica 1. Neke od ključnih riječi PDB formata .....	9
Tablica 2. Struktura retka sa zapisa atoma u PDB formatu .....	10

# 1. Uvod

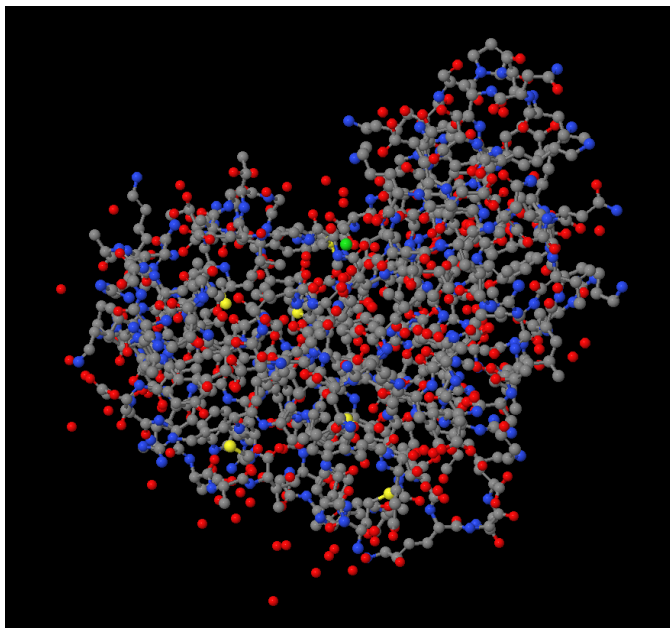
Proteini su jedni od najvažnijih makromolekula prisutnih u organizmu koje sudjeluju u gotovo svakom biološkom procesu kao što su prijenos i spremanje kemijskih spojeva, kataliziranje kemijskih reakcija, prevođenje i signalizacija informacija od strane drugih proteina, očuvanje strukture stanica i tkiva, pretvorba kemijske energije u mehaničku (pokretanje mišića) te mnogi drugi. Funkcija proteina striktno je određena njegovom 3D strukturom, te je područje istraživanja utjecaja 3D strukture na funkcionalnost proteina trenutno vrlo aktivno područje u bioinformatici [1]. Jednom kada nam je jasan utjecaj 3D strukture na funkciju proteina, otvara se mogućnost ručnog dizajniranja proteina koji bi na točno određeni način djelovali na druge proteine što bi u konačnici dovelo do lakšeg i jednostavnijeg razvoja raznih lijekova [2].

Strukturu (pozicije atoma) i izgled proteina je moguće odrediti korištenjem nekoliko različitih metoda: rendgenskom kristalografijom, NMR spektroskopijom, elektron mikroskopijom [1]. Te informacije je zatim moguće iskoristiti za 3D rekonstrukciju proteina u nekom od alata za računalnu vizualizaciju proteina što nam u konačnici omogućava da detaljnije proučavamo proteine, ali i paralelno ih uspoređujemo s drugim proteinima. Ovaj način istraživanja je vrlo čest jer poznate funkcionalnosti jednog proteina možemo preslikati na slične ili identične dijelove drugog proteina te na kraju djelomično ili u potpunosti shvatiti funkciju novog proteina. Dobar alat za vizualizaciju je ovdje od velike važnosti jer mala vizualna razlika može uvelike utjecati na konačnu funkcionalnost proteina.

Prije računalnih programa za vizualizaciju, molekule su se najčešće proučavale kao fizički modeli sa kuglama i štapićima (Slika 1), što je vrlo nepraktično za velike molekule poput proteina, međutim razvoj 3D računalne grafike omogućio je puno sofisticiraniju vizualizaciju u kojoj korisnik može interaktivno pregledavati i analizirati pojedinačne dijelove kompliciranijih molekula. Uz to, programi za vizualizaciju najčešće omogućuju različite stilove vizualnog prikaza proteina koji olakšavaju proučavanje funkcionalnosti. Jedan od najčešće korištenih stilova je tzv. vrpčasti prikaz (ili prikaz korištenjem



vrpca) (engl. *ribbons*) koji olakšavaju prikaz i analizu tercijarne strukture proteina (Slika 2)



Slika 1. Prikaz proteina lizozima (253L) sa kuglama i štapićima u alatu Jmol



Slika 2. Prikaz proteina lizozima (253L) korištenjem vrpce u alatu Jmol

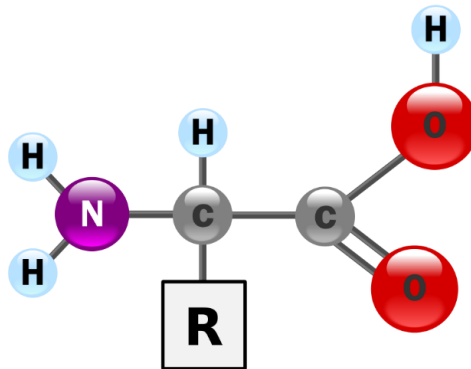
Tema ovog rada bit će prikazati izvedbu programa koji na ulazu prima zapis strukture proteina na temelju kojeg generira interaktivni 3D model tog proteina.

U 2. poglavlju će za lakše razumijevanje najprije biti objašnjene biološke i kemijske osnove proteina, a potom će u poglavlju 3. ukratko biti opisani različiti zapisi strukture proteina od kojih će detaljnije biti objašnjen najpopularniji, .PDB format. U poglavlju 4. bit će razrađeni složeniji dijelovi programa potrebni za generiranje i prikaz modela proteina, nakon čega će u 5. poglavlju biti prikazana usporedba dobivenih rezultata sa rezultatima već postojećih alata. Zaključak, literatura i sažetak dani u poglavljima 6., 7. i 8., dok se u dodatku A nalaze upute za instalaciju i korištenje programa.

## 2. Proteini

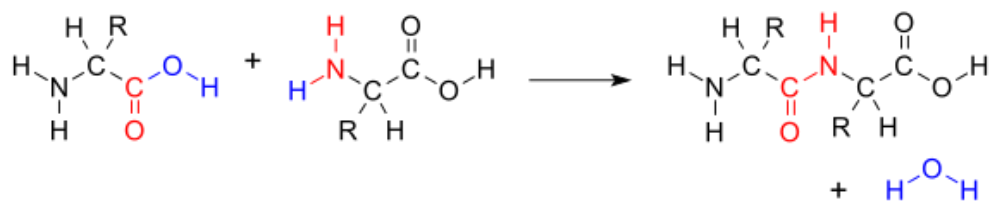
Proteini ili bjelančevine su makromolekule koje se sastoje od niza aminokiselina, međusobno povezanih poput karika u lancu. Protein se može sastojati od više takvih lanaca, a redoslijed i broj aminokiselina u pojedinom lancu određuje specifične osobine i funkcije svakog proteina. Njihova prva i osnovna zadaća je proces rasta i razvoja, a odgovorni su i za nadomještanje oštećenih i odumrlih stanica te služe kao enzimi za ubrzavanje biokemijskih procesa [3].

Aminokiseline su molekule koje imaju slobodnu amino grupu ( $\text{NH}_2$ ) i slobodnu karboksilnu skupinu ( $\text{COOH}$ ) gdje su obje skupine vezane na središnji, alfa atom ugljika ( $\text{C}_\alpha$ ) na koji je još vezan i bočni lanac R koji određuje svojstva svake aminokiseline (Slika 3).



Slika 3. Aminokiselina

Aminokiseline se mogu međusobno povezati formiranjem peptidne veze te tako stvoriti polipeptidni lanac - protein. Peptidna veza koja spaja dvije aminokiseline je kovalentna veza koja nastaje između karboksilne skupine jedne i amino skupine druge aminokisline te se pri tome oslobađa jedna molekula vode.

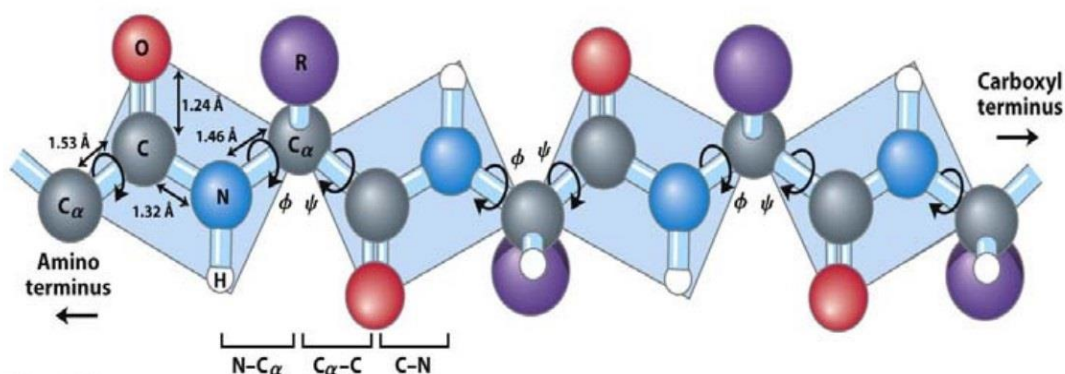


Slika 4. Stvaranje peptidne veze

Zbog lakšeg proučavanja, struktura proteina podijeljena je na 4 različite razine: primarnu, sekundarnu, tercijarnu te kvartarnu.

## 2.1. Primarna struktura

Primarna struktura je redoslijed aminokiselina u pojedinom lancu proteina. Lanac je usmjeren i započinje amino krajem, a završava karboksilnim krajem peptida. Tri veze odvajaju susjedne  $C\alpha$  atome. Veze oko  $C\alpha$  ( $C\alpha - C$  i  $N - C\alpha$ ) mogu rotirati, dok peptidna veza  $CO = NH$  ne (Slika 5) [4]. Ove rotacije omogućavaju proteinima da se nabiru na različite načine što omogućava velik broj različitih oblika [4].



Slika 5. Rotacije u polipeptidnom lancu [4]

## 2.2. Sekundarna struktura

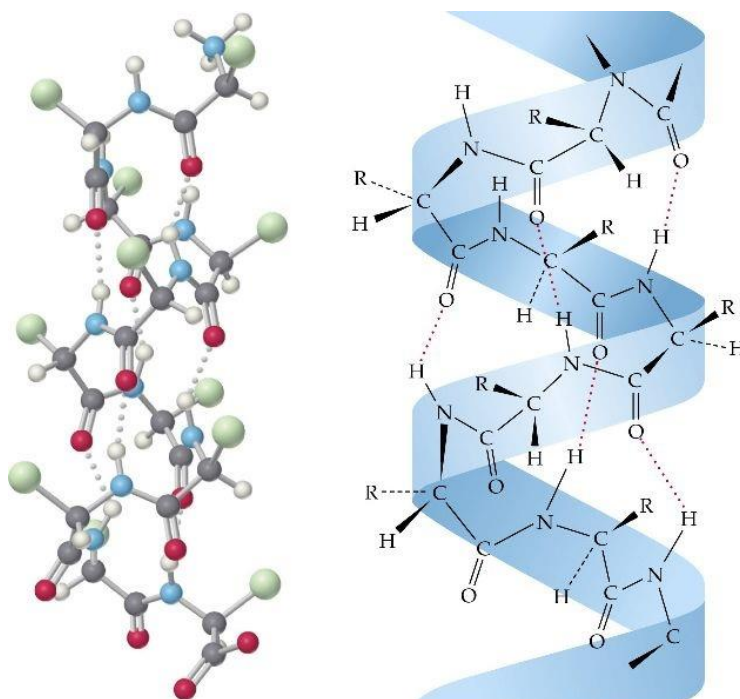
Sekundarna struktura je prostorni raspored  $C\alpha$  atoma koji čine okosnicu, a da se pri tome ne uzima u obzir konformacija bočnih ogranaka aminokiselina [5].

Veliki broj načina rotacija peptidnog lanca pridonosi stvaranju nekih pravilnih tvorevina među kojima su najznačajnije  $\alpha$ -zavojnice te  $\beta$ -lanci koji

čine  $\beta$ -naboranu ploču. Uz njih, u sekundarnu strukturu se još ubrajaju i petlje te oštri zavoji ( $\beta$ -zavoji) [1].

### 2.2.1. $\alpha$ -zavojnica

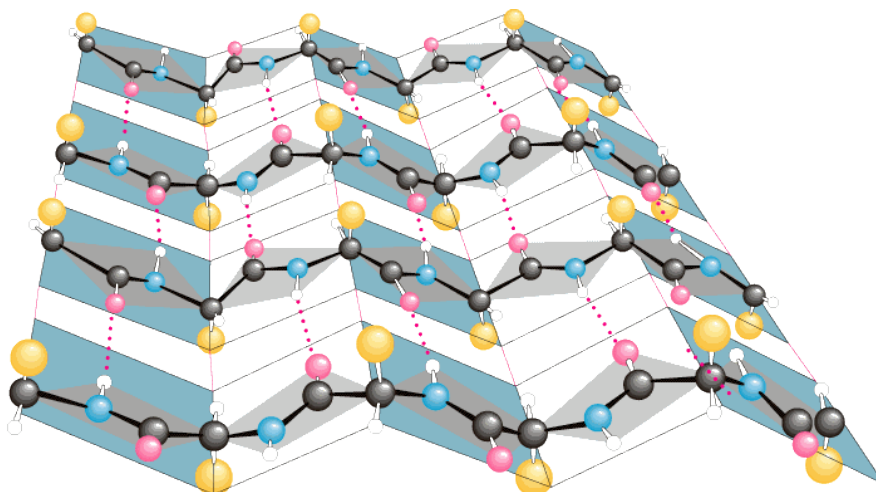
U  $\alpha$ -zavojnici, polipeptidni lanac zavija u lijevu ili desnu stranu i stvara strukturu čvrsto pakiranog valjka (Slika 6) [5]. Do takve strukture dolazi zbog stvaranja vodikovih veza između svakog četvrtog C $\alpha$  atoma [1]. Smjer zavojnice može biti i lijevi i desni, međutim desni smjer je energetski puno povoljniji te su zavojnice sa lijevim smjerom iznimno rijetke [5].



Slika 6. Struktura  $\alpha$ -zavojnice [5]

### 2.2.2. $\beta$ -naborana ploča

Za razliku od  $\alpha$ -zavojnice,  $\beta$ -naborana ploča je izdužena tvorevina koju čine dva ili više  $\beta$ -lanaca, međusobno povezanih vodikovom vezom između amino i karboksilne skupine (Slika 7) [5]. Lanci aminokiselina koji su međusobno povezani u  $\beta$ -naboranoj ploči mogu biti paralelni, antiparalelni ili mogu biti kombinacija paralelnih i antiparalelnih lanaca [1].



Slika 7. Struktura  $\beta$ -naborane ploče

## 2.3. Tercijarna struktura

Cjelokupni raspored i utjecaj svih sekundarnih struktura jednog polipeptidnog lanca definira terciarnu strukturu te se uz to još i promatra interakcija između R-grupa svih aminokiselina koje čine taj lanac [6]. U tu se interakciju ubrajaju vodikove veze, ionske veze, dipol-dipol interakcije te Londonove disperzijske sile. Također, ovdje su vrlo bitne i hidrofilne te hidrofobne interakcije koje određuju na koji će način protein reagirati s vodom [6].

## 2.4. Kvartarna struktura

Proteini koji se sastoje od samo jednog polipeptidnog lanca imaju samo prve tri strukture, dok proteini koji su sačinjeni od dva ili više lanca imaju još i kvartarnu strukturu. Općenito, u kvartarnoj strukturi se gledaju iste interakcije i stvaranje veza kao i u terciarnoj strukturi, no umjesto na jednom lancu, ovdje se promatraju međudjelovanja jednog lanca na drugi [6].

## 2.5. Stabilnost proteina

Kako su sekundarna, terciarna i kvartarna struktura proteina određena slabim silama, proteini su vrlo osjetljivi te su lako podložni poremećaju strukture ili denaturaciji [3]. Ukoliko se naruši njihova struktura, tada oni više nisu u stanju obavljati svoju osnovnu funkciju što dovodi do poremećaja stanične aktivnosti te moguće smrti stanice.

## 2.6. Određivanje 3D strukture proteina

Određivanje strukture proteina najčešće se vrši dvjema popularnim metodama: rendgenskom kristalografijom (engl. *x-ray crystallography*) te nuklearnom magnetskom rezonancijom (engl. *Nuclear Magnetic Resonance Spectroscopy, NMR*) [1].

Analiza rendgenskom kristalografijom radi se na način da se protein najprije zamrzne kako bi se fiksirao položaj atoma. Potom taj uzorak bombardiramo rendgenskim zrakama koje se odbijaju od elektronskih oblaka obližnjih atoma te prilikom difrakcije proizvode regularne uzorke koje potom zabilježimo. Iz tih podataka, korištenjem matematičkih alata te heurističkih metoda, možemo iz dobivenih rezultata očitati položaj atoma u uzorku [1].

Drugi najčešći način određivanja strukture je spektroskopija nuklearnom magnetskom rezonancijom. Ova metoda mjeri promjene magnetskog momenta jezgre u vanjskom magnetskom polju na temelju kojih se mogu dobiti podaci o strukturnim i dinamičkim svojstvima molekula bilo da su slobodne ili vezane [7].

### 3. Zapisi proteina

Jednom kada se odredi struktura proteina, sve sakupljene informacije potrebno je zapisati u nekom formatiranom obliku. Trenutno su u uporabi dva formata: stariji PDB te noviji mmCIF [1].

PDB format osmišljen je ranih 1970ih kao format u kojem bi bile zapisane strukture proteina koje se spremaju u Protein Data Bank repozitorij [1]. Inicijalno je bio napravljen da bude kompatibilan s FORTRAN programskim jezikom te ima strogu strukturu od 80 znakova u jednoj liniji pri čemu je svaka linija poseban zapis. Datoteka započinje zaglavljem u kojem su opisane proizvoljne informacije o proteinu kao što su ime molekule, podrijetlo organizma, način određivanja strukture, kristalografski parametri i rezolucija i sl [1]. Svaki redak započinje nekom od ključnih riječi nakon kojeg slijede podaci koji su strukturirani ovisno o toj ključnoj riječi. Neke od ključnih riječi i njihova objašnjenja dana su u Tablici 1, dok je u Tablici 2 prikazana struktura retka koji sadrži informacije o jednom atomu.

Tablica 1. Neke od ključnih riječi PDB formata

Ključna riječ	Značenje
HEADER	Zaglavlje
TITLE	Naziv molekule
REMARK	Općenite informacije u slobodnom obliku
MODEL	Označava početak jednog modela proteina (format dozvoljava više zapisa iste molekule)
ATOM	Informacije o jednom atomu u molekuli
HETATM	Heterogeni atom koji ne pripadaju standardnim grupama atoma, npr. za opis atoma molekule vode koja ne pripada molekuli proteina
TER	Kraj zapisa atoma jednog lanca aminokiselina



ENDMDL	Kraj informacija za jedan model proteina (dolazi u paru sa MODEL oznakom)
--------	---

Tablica 2. Struktura retka sa zapisa atoma u PDB formatu

Stupac	Sadržaj	Tip podatka
1-4	„ATOM“	Znakovi
7-11	Serijski broj atoma	Cijeli broj (integer)
13-16	Ime atoma	Znakovi
17	Indikator alternativne lokacije	Znak
18-20	Ime člana (aminokiseline)	Znakovi
22	Identifikator lanca	Znak
23-26	Redni broj člana (aminokiseline)	Cijeli broj (integer)
27	Kod za umetanje člana	Znak
31-38	X koordinata atoma	Realni broj (float 8.3)
39-46	Y koordinata atoma	Realni broj (float 8.3)
47-54	Z koordinata atoma	Realni broj (float 8.3)
55-60	Zauzeće	Realni broj (float 6.2)
61-66	Temperaturni faktor	Realni broj (float 6.2)
73-76	Identifikator segmenta	Znakovi
77-78	Simbol elementa	Znakovi
79-80	Naboj atoma	Znakovi

PDB format već je dugo u uporabi te ga je lako čitati i koristiti, međutim, format nije dizajniran za efikasno izvlačenje podataka koje je sve češće potrebno prilikom pretraživanja baze podataka [1]. Nadalje, pojedine restrikcije su podosta zakomplicirale njegovo korištenje, npr. u formatu se navode samo koordinatne pozicije atoma, ali ne i njihove veze što stvara problem prilikom

određivanja disulfidnih veza koje nisu zapisane u datoteci već ih program mora sam odrediti (neki programi ni ne uspiju) [1]. Još neke negativne strane formata su npr. veličina stupca za serijski broj atoma koji maksimalno dopušta peteroznamenasti broj, čime se stavlja ograničenje da jedan model proteina može imati maksimalno 99 999 atoma. Također, identifikator lanca je definiran samo jednim znakom što znači da cjelokupna molekula proteina može imati najviše 26 lanaca (jer je identifikator lanca određen velikim slovom engleske abecede [1]. Sva ova ograničenja stvaraju probleme prilikom rada s velikim proteinskim kompleksima koji zbog tih ograničenja moraju biti podijeljeni na više datoteka, stoga je daljnji razvoj ovog formata zaustavljen 2012 godine te se podaci u PDB formatu više ni ne prihvaćaju u Protein Data Bank bazi podataka [8].

Ograničenja PDB formata dovela su do razvoja novih formata od kojih je najpopularniji *macromolecule crystallographic information file* ili kraće mmCIF. Format se temelji na CIF (engl. *crystallographic information file*) formatu koji se koristi za zapis manjih molekula te je proširen informacijama potrebnim za prikaz većih makromolekula [9]. mmCIF podržava razne vrste informacija koje su definirane u mmCIF rječniku, te atributi i njihove vrijednosti, za razliku od PDB formata, mogu biti dugački i deskriptivni.

Jednom kada je struktura proteina određena i zapisana u mmCIF format, datoteka se može objaviti na *Protein Data Bank* (skraćeno PDB) repozitorij – jedini svjetski repozitorij za procesiranje i distribuciju podataka o trodimenzionalnim strukturama makromolekula i nukleinskih kiselina. Repozitorij u trenutku pisanja broji više od 153 000 struktura, te su svi podaci besplatno dostupni javnosti [10].

## 4. Program za vizualizaciju proteina

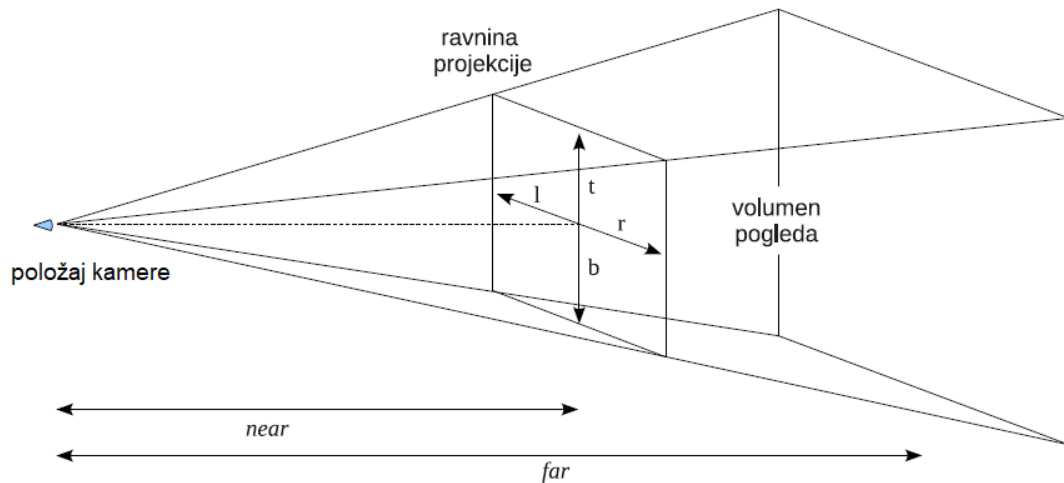
Izrada programa za vizualizaciju proteina zahtijeva široki spektar znanja. Potrebno je poznavati biološke osnove proteina, strukturu zapisa proteina, matematičke osnove o krivuljama i linearnoj algebri, te naravno, poznavanje nekih od grafičkih API-ja za 3D grafiku.

U sklopu ovog rada razvijen je alat ProteinVisualizer čija je svrha prikazivanje tercijarne i kvartarne strukture proteina. Alat je podržan na Windows platformama a pisan je u C++ programskom jeziku uz korištenje DirectX11 API-ja. Cjelokupni program sastoji se od mnogo dijelova i razreda poput razreda za generiranje krivulja, posebne matematičke knjižnice, razreda za upravljanje kamerom, razreda za generiranje svjetla, razreda za generiranje cijevi, razreda za generiranje vrpce i još mnogih drugih. U nastavku će biti opisani samo najvažniji i najsloženiji dijelovi programa, dok je slijed izvođenja ključnog dijela programa u kojem se generira kompletan model sljedeći:

1. *Parsiranje i spremanje podataka iz ulazne PDB datoteke*
2. *Priprema (indeksiranje) i sortiranje podataka prema poziciji pojavljivanja svih sekundarnih struktura ( $\alpha$ -zavojnica i  $\beta$ -lanaca) za svaki lanac u proteinu*
3. *Za svaki lanac u proteinu ponoviti:*
  - 3.1. *Za svaku sekundarnu strukturu u lancu ponoviti:*
    - 3.1.1. *Generiraj poligone za okosnicu do kraja prošle (ili početka lanca), do početka trenutne sekundarne strukture*
    - 3.1.2. *Generiraj poligone za trenutnu sekundarnu strukturu*
  - 3.2. *Generiraj poligone za okosnicu od kraja zadnje sekundarne strukture, do kraja lanca*
4. *Sve generirane vrhove poligona sa pripadajućim indeksima spremi u spremnik vrhova (engl. vertex buffer), odnosno spremnik indeksa (engl. index buffer) kako bi se generirani poligoni mogli prikazati na zaslonu.*

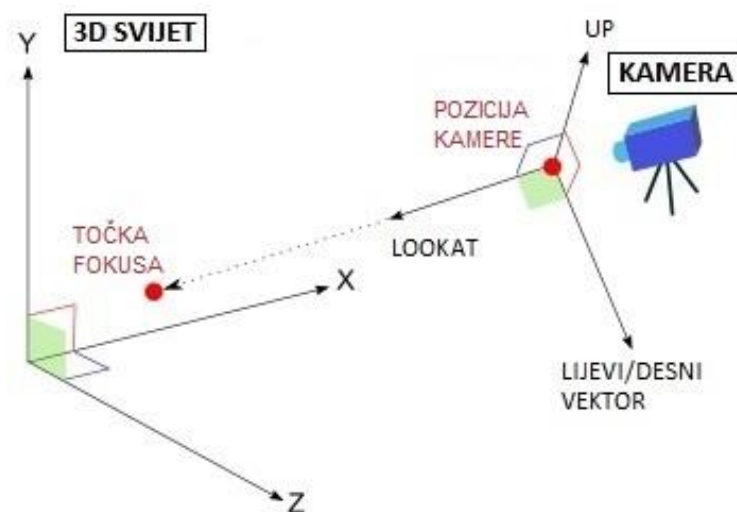
## 4.1. Kamera

Kamera je u 3D grafici jedan od ključnih dijelova. Isto kao i ostali objekti, kamera ima svoj položaj ali i orijentaciju te neke druge, trenutno manje važne, parametre. Osnove 3D kamere u računalnoj grafici prikazane su na slikama 8 i 9.



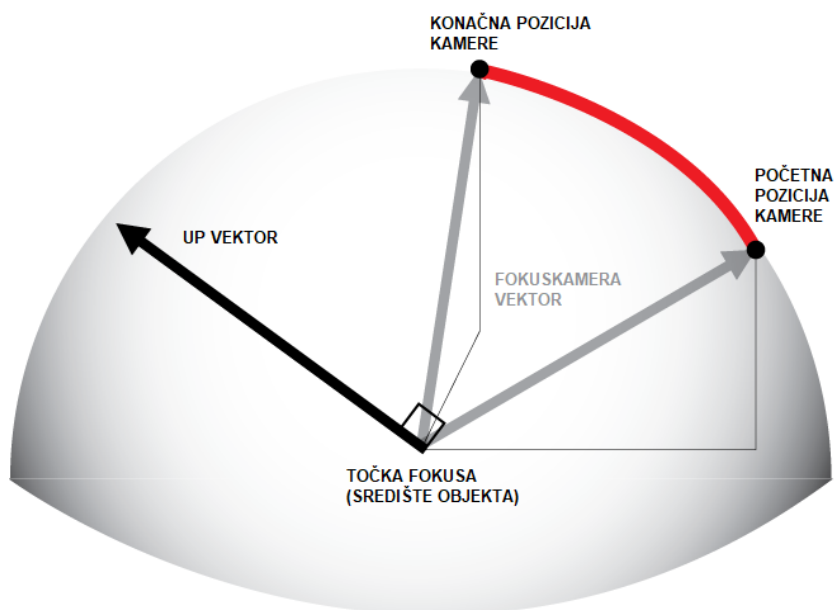
Slika 8. Model projekcijske kamere [11]

Orijentacija kamere je jednako je važna kao i pozicija, a određena je pomoću dva vektora. Prvi vektor je *lookAt* vektor koji je usmjeren od položaja kamere prema točki fokusa. Taj vektor prolazi kroz središte bliže i daljnje ravnine odsijecanja koje određuju međuprostor koji se u konačnici prikazuje na zaslonu. Drugi vektor je takozvani *up* vektor čija je svrha dati pravilnu orijentaciju kamere u prostoru. Vektorskim produktom *up* i *lookAt* vektora dobije se još i treći, bočni vektor koji je usmjeren bočno od kamere. Ukoliko se u programu koristi lijevi koordinatni sustav, bočni vektor će pokazivati desno od kamere, a ukoliko se koristi desni koordinatni sustav, bočni vektor će pokazivati lijevo od kamere.



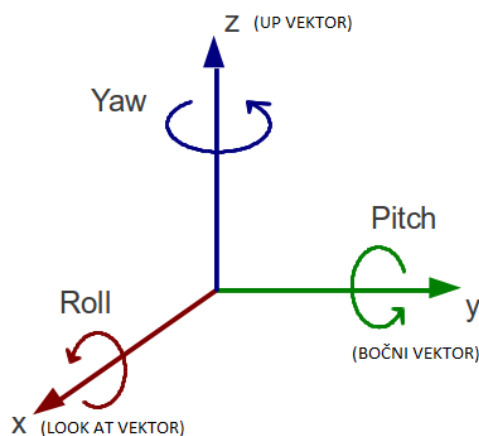
Slika 9. Položaj kamere u prostoru

Za detaljniju analizu i proučavanje pojedinačnog objekta, kao što su u ovom slučaju proteini, najčešće se koristiti tehnika u kojoj se središte objekta istraživanja stavlja u točku fokusa kamere te se pri tom kamera rotira oko njega po zamišljenoj sferi (Slika 10). Ekvivalentan prikaz mogli bismo dobiti da, umjesto pozicije kamere, rotiramo model oko točke fokusa



Slika 10. Rotacija kamere oko središta objekta

Kamera se u programu najčešće i najlakše kontrolira korištenjem miša. Kako bi realizirali ranije opisanu tehniku, program prvo treba registrirati određen pomak miša na temelju kojeg se potom određuje smjer i količina rotacije. Kada miš pomičemo lijevo ili desno, očekujemo da se kamera zarotira ulijevo odnosno udesno oko proteina (tj. da se protein zarotira u suprotnom smjeru, oko točke fokusa). Međutim, kako je kamera objekt u prostoru koji može imati proizvoljnu poziciju i koordinate, moramo biti pažljivi da ne radimo rotaciju korištenjem globalnih koordinatnih osi. Ispravan način je zapravo koristiti koordinatne osi kamere (*up*, *lookAt* i bočni vektor). Tako horizontalnu rotaciju ulijevo i udesno dobijemo na sljedeći način: kreiramo *fokusKamera* vektor koji pokazuje od fokusa kamere prema poziciji kamere (Slika 10). Taj vektor je kolinearan sa *lookAt* vektorom (te je ujedno i okomit na *up* vektor). Rotacijom *fokusKamera* vektora oko *up* vektora postavljenog u točku fokusa, dobiva se nova pozicija kamere. Ova rotacija naziva se još i yaw rotacija (Slika 11).



Slika 11. Rotacije oko koordinatnih osi

Prilikom te bočne, odnosno yaw, rotacije, smjer *up* vektora se neće promijeniti jer on čini rotacijsku os, tj. kolinearan je s osi rotacije, no bočni vektor hoće. Kako bočni vektor u svakom trenutku možemo dobiti vektorskim produktom *up* i *lookAt* vektora, nema potrebe da radimo dodatnu rotaciju.

Vertikalnu rotaciju dobijemo isto rotacijom *fokusKamera* vektora, ali ovaj put oko bočnog vektora kamere postavljenog u točku fokusa. Ova rotacija se još i

naziva pitch rotacija. Međutim, prilikom ove rotacije moramo biti oprezni jer, osim rotacije *fokusKamera* vektora, moramo rotirati i *up* vektor korištenjem iste rotacijske matrice kako bi osi kamere ostale konzistentne i okomite jedne na drugu.

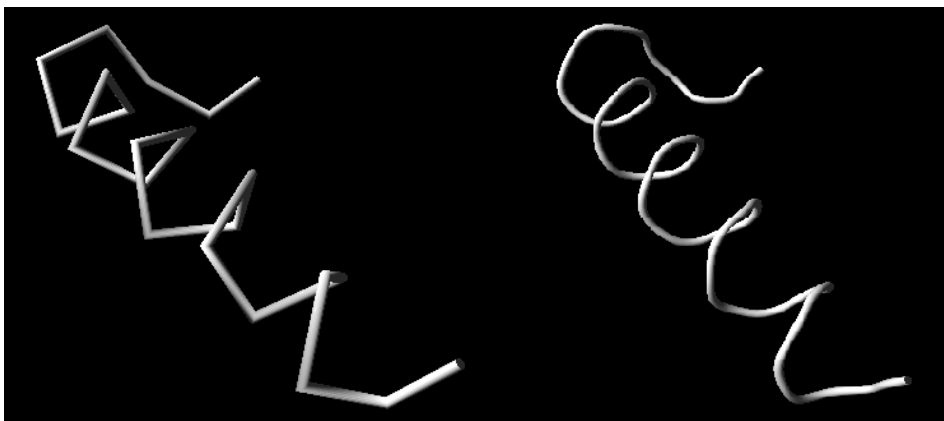
Kako su rotacije u 3D grafici poprilično opširna i kompleksna tema, da ne ulazimo preduboko, spomenut ćemo samo kako gotovo sva API sučelja za računalnu grafiku imaju ugrađene metode kojima je moguće dobiti potrebnu rotacijsku matricu za rotaciju jednog vektora oko drugog.

Osim rotacije, koja čini najkompleksniji dio kamere, potrebno je još i realizirati zumiranje te pomicanje, odnosno translaciju objekta. Zumiranje je vrlo jednostavno izvesti. Kako je ranije rečeno, kamera se miče po zamišljenoj sferi čiji je središte u središtu objekta, a polumjer određen duljinom *fokusKamera* vektora. Ukoliko se želimo približiti ili udaljiti, potrebno je jednostavno smanjiti odnosno povećati polumjer sfere, drugim riječima, dovoljno je samo skalirati *fokusKamera* vektor kako bi približili ili udaljili poziciju kamere.

Na početku pokretanja programa, središte objekta ujedno je i točka fokusa kamere, međutim ukoliko cijeli objekt želimo pomaknut u stranu, tada se središte objekta više neće poklapati sa fokusom kamere. Ako cijeli objekt želimo pomaknuti horizontalno, to možemo učiniti tako da točku fokusa transliramo korištenjem bočnog vektora kamere, a ako želimo pomaknut vertikalno, translaciju radimo korištenjem *up* vektora.

## 4.2. Izrada 3D modela proteina

Kada bi prikazivali svaki pojedini atom proteina korištenjem metode kuglica i štapića, prikaz bi vrlo brzo postao prekompleksan (Slika 1), stoga se danas najčešće koristi prikaz okosnice zajedno sa vizualno izraženijim sekundarnim strukturama. Okosnica je sačinjena od  $C\alpha$  atoma svake aminokiseline u lancu, a kako bi cijela struktura izgledala glađe i prirodnije, umjesto direktnog spajanja atoma okosnice, kroz njih se često provlači krivulja (Slika 12). Osim za generiranje okosnice, krivulja kroz  $C\alpha$  atome pomaže i prilikom generiranja  $\alpha$ -zavojnica i  $\beta$ -lanaca



Slika 12. Usporedba direktnog spajanja Ca atoma (ravna okosnica) te spajanja korištenjem krivulja (zakrivljena okosnica)

#### 4.2.1. Krivulje

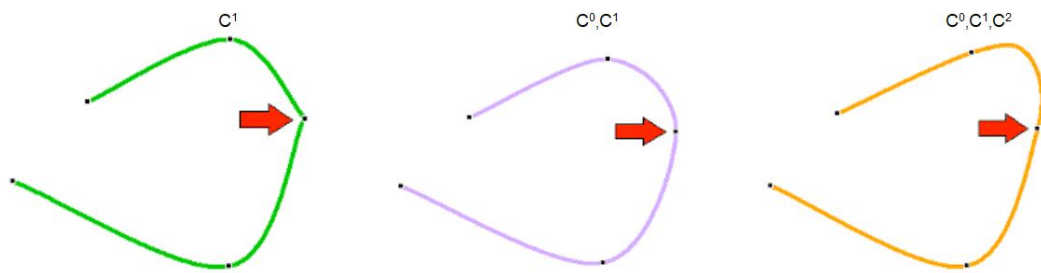
Matematički gledano, krivulja je neprekidna crta, ili točnije rečeno, jednodimenzionalni skup točaka[11]. Krivulje se obično zadaju analitički te imamo tri osnovne kategorije zapisa [12] :

- Eksplicitnom jednačbom  $y = f(x)$
- Implicitnom jednačbom  $f(x, y) = 0$
- Parametarskim zapisom, npr.  $x = \cos(t)$ ,  $y = \sin(t)$

Pojedini zapisi imaju određene probleme pa tako, na primjer, eksplicitnom jednačbom  $y = ax + b$  ne možemo prikazati pravac paralelan s y-osi te nije moguće prikazati funkciju s višestrukim vrijednostima. Implicitne jednačbe rješavaju ove nedostatke, no one pak imaju problem što ne mogu prikazivati djelomičan prikaz na određenom intervalu [12].

Jedno od važnijih svojstava krivulja je svojstvo neprekinutosti pa tako postoje C kontinuiteti krivulja:  $C^0$  kontinuitet zahtjeva neprekinutost u koordinatama,  $C^1$  kontinuitet zahtjeva neprekinutost prve derivacije, drugim riječima, krivulja ne smije imati šiljke, već mora biti glatka,  $C^2$  zahtjeva neprekinutost druge derivacije u svim točkama. To nam osigurava neprekinutost zakrivljenosti,  $C^3$  zahtjeva neprekinutost treće derivacije u svim točkama itd. (Slika 13) [12].





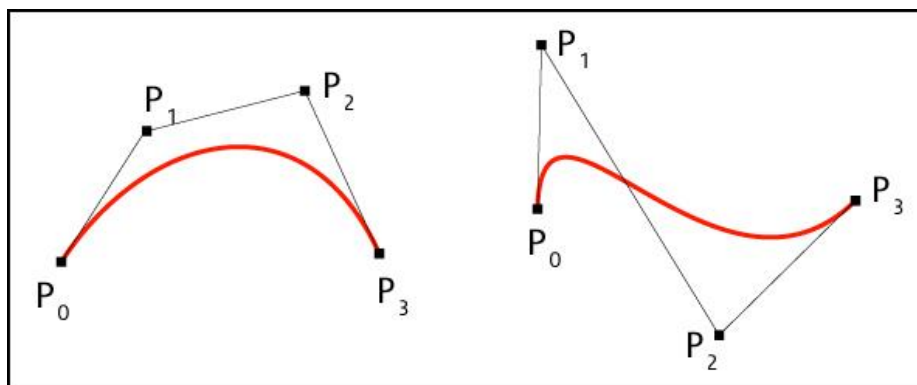
Slika 13. Razlike u C kontinuitetu kod krivulja [13]

Krivulje se u računalnoj grafici najčešće konstruiraju korištenjem kontrolnih točaka i težinskih polinoma. Ideja je sljedeća: svaka kontrolna točka doprinosit će ukupnom obliku krivulje svojim koordinatama pomnoženim s pripadajućom težinskom funkcijom koja daje vrijednost od 0 do 1 [12] :

$$T_K(t) = \sum_{i=0}^n f_i(t) \cdot T_i$$

gdje je  $T_K$  točka krivulje,  $T_i$   $i$ -ta zadana kontrolna točka,  $f_i(t)$  težinska funkcija za  $i$ -tu točku, a  $t \in [0, 1]$  [12].

Ovisno o potrebi, postoje razni načini kako definirati težinske funkcije, međutim ovako zadana krivulja neće prolaziti kroz kontrolne točke. Na primjer, na slici 14, crvenom bojom prikazane su dvije Bezierove krivulje koje se temelje na toj formuli, dok  $P_0$ ,  $P_1$ ,  $P_2$  i  $P_3$  predstavljaju kontrolne točke. Na slici se jasno vidi kako kontrolne točke „privlače“ smjer krivulje u određenu stranu, no krivulja nikad ne prolazi točno kroz točke (izuzev prve i posljednje točke iz kojih mora krenuti, odnosno stati). Težinske funkcije pak služe da definiraju jačinu utjecaja pojedinih kontrolnih točaka.

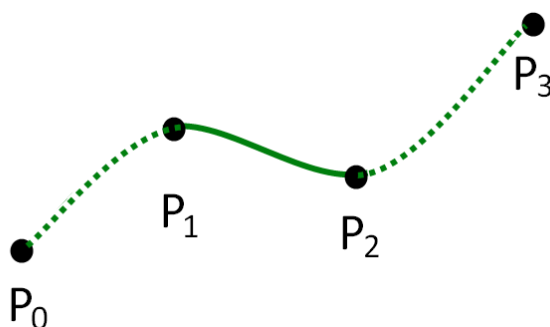


Slika 14. Bezierove krivulje [14]

Krivulje koje ne prolaze kroz dane točke nazivaju se aproksimacijskim krivuljama [12] te nisu dobar odabir za naše potrebe. Još jedna velika mana ovako zadanih krivulja je u tome što su težinske funkcije polinomi čiji stupanj ovisi o broju kontrolnih točaka. Ukoliko imamo četiri kontrolne točke, kao na slici 14, tada težinske funkcije trebaju biti trećeg stupnja, odnosno ukoliko imamo  $n$  kontrolnih točaka, težinske funkcije trebaju biti  $n-1$  stupnja što može biti vrlo nepovoljno ako imamo puno točaka.

Uz pomoć algebarskih operacija, moguće je iz ranije navedenog zapisa doći do funkcije koja generira krivulju koja prolazi danim točkama. Općenito, takve krivulje se nazivaju interpolacijskim [12] te su upravo ono što trebamo za generiranje okosnice proteina. Međutim, i dalje ostaje problem gdje je potrebno imati polinom stupnja za jedan manje od broja kontrolnih točaka koje definiraju krivulju. Problem bi bilo najbolje riješiti kada bismo mogli spojiti više manjih dijelova krivulje u jednu veću krivulju. Međutim, tu treba biti pažljiv da spajanje krivulja bude glatko tako da krivulja zadrži (barem)  $C^1$  kontinuitet. Srećom, postoji upravo takva vrsta krivulja, a radi se o Catmull-Romovom splajnu.

Catmull-Romov splajn definirali su Edwin Catmull i Raphael Rom [13]. Radi se o interpolirajućoj krivulji definiranoj sa 4 kontrolne točke  $P_0$ ,  $P_1$ ,  $P_2$  i  $P_3$  gdje je krivulja definirana samo između  $P_1$  i  $P_2$  (Slika 15) [13].



Slika 15. Catmull-Rom krivulja

Krivulja je definirana sljedećom funkcijom [14] :

$$P(t) = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} -S & 2-S & S-2 & S \\ 2S & S-3 & 3-2S & -S \\ -S & 0 & S & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}, t \in [0,1]$$

gdje parametar  $S \in [0,1]$  predstavlja napetost krivulje. Za  $S = 0$ , krivulja izgleda kao ravan pravac između dvije točke dok za  $S = 1$  krivulja postaje vrlo zaobljena. Za optimalne rezultate, najbolje je koristiti vrijednost između 0.5 i 1.

Odlično svojstvo Catmull-Rom krivulje je to što dvije susjedne krivulje zadržavaju  $C^1$  kontinuitet što u konačnici omogućuje postupnu izgradnju proizvoljno velike krivulje koja na cijelom području zadržava  $C^1$  kontinuitet [14]. Međutim, i ova vrsta krivulja ima svoje probleme. Ako koristimo samo dani skup kontrolnih točaka kroz koje krivulja treba interpolirati, tada nećemo moći generirati dio krivulje između prve i zadnje dvije točke. Ovo se može riješiti na nekoliko načina: Prvi je da korisnik ručno generira dodatne točke što često nije poželjno jer želimo da cijeli proces bude automatiziran. Drugi način je da jednostavno dupliciramo početnu i završnu točku, no to ponekad uzrokuje čudne završetke [15], te treći, najčešće korišteni način, je da napravimo refleksiju druge točke preko prve, odnosno predzadnje točke preko zadnje. Ovaj način u praksi daje najbolje rezultate [15].

Kako cjelokupna krivulja, sastavljena od više manjih Catmull-Rom krivulja, ima  $C^1$  kontinuitet, a to znači da je možemo u bilo kojoj točki derivirati i time dobiti

tangentu koja nam pokazuje trenutni smjer krivulje. To možemo odrediti ako deriviramo ranije napisanu formulu, čime se dobije sljedeći izraz:

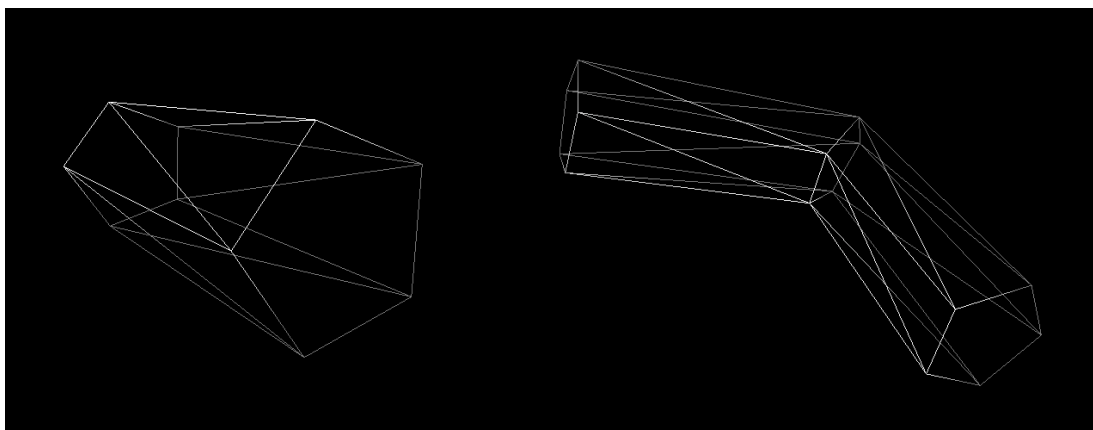
$$P'(t) = \begin{bmatrix} -3St^2 + 4St - S \\ 3(2-S)t^2 + 2(s-3)t \\ 3(S-2)t^2 + 2(3-2S)t + S \\ 3St^2 - 2St \end{bmatrix}^T \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

#### 4.2.2. Izrada okosnice proteina

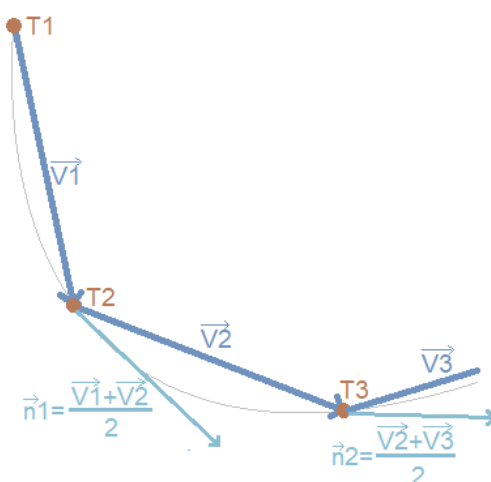
Jednom kada znamo konstruirati krivulju koja prolazi kroz točno određene točke, možemo tu krivulju iskoristiti kao bazu za konstruiranje 3D struktura koje zajedno daju cjelokupni model proteina [2]. Najvažnija od tih struktura je središnja okosnica (engl. *wireframe*) koja već sama po sebi može biti dovoljna za analizu proteina. Geometrijski gledano, okosnica se najčešće modelira kao jednostavna cijev koja prati krivulju. Kako se u grafici sve modelira korištenjem ravnih poligona, nije moguće generirati u potpunosti glatku, zaobljenu cijev, no privid glatke površine možemo ostvariti na način da na dovoljno malim razmacima konstruiramo kratke, spojene cijevi, dok te kratke cijevi generiramo kao izdužene n-terokute (Slika 16). Generiranje kratke ravne cijevi ne predstavlja veliki problem. Problem se javlja tek prilikom spajanja dviju cijevi. Kako bi direktan prijelaz iz jedne cijevi na drugu, potrebno je konstruirati model u kojem su vrhovi spoja ujedno i vrhovi obiju cijevi (drugim riječima, cijevi dijele iste vrhove na zajedničkom spoju), a za to je potrebno malo prostorne geometrije. Na početku, potrebno je generirati bazne točke početnog n-terokuta na ravnini određenoj vektorom smjera prve cijevi  $V_1$  i točkom  $T_1$  (Slika 17). Jednom kada imamo skup početnih točaka ponavljamo sljedeći postupak:

1. Odredimo vektor  $\vec{n}_i$  normale ravnine koja dijeli dva susjedna segmenta kao  $\frac{\vec{v}_{i-1} + \vec{v}_i}{2}$  (ukoliko se radi o posljednjoj točki na krivulji, za normalu samo uzmemo posljednji vektor)
2. Za svaku baznu točku  $B_{j,i-1}$  iz prijašnje iteracije, odredimo novu baznu točku  $B_{j,i}$  koja nastane kao probodište pravca određenog tom prijašnjom baznom točkom i vektora  $\vec{v}_{i-1}$ , s ravninom koja je definirana ranije određenim vektorom normale  $\vec{n}_i$  i točkom  $T_i$

Na primjer, recimo da se nalazimo u točki T2 na slici 17. Točka T2 predstavlja spoj cijevi određenih točkama T1 i T2, te T2 i T3. U prvom koraku potrebno je odrediti vektor normale ravnine  $\vec{n}_2$  u točki T2. Njega dobijemo kao  $\vec{n}_2 = \frac{\vec{v}_1 + \vec{v}_2}{2}$ . Sada kada imamo tu ravninu koja čini spoj dviju cijevi, preostaje nam kroz točke prijašnjeg spoja povući pravac, čiji je smjer određen smjerom vektora  $\vec{v}_1$ , te izračunati probodište tog pravca sa izračunatom ravninom.



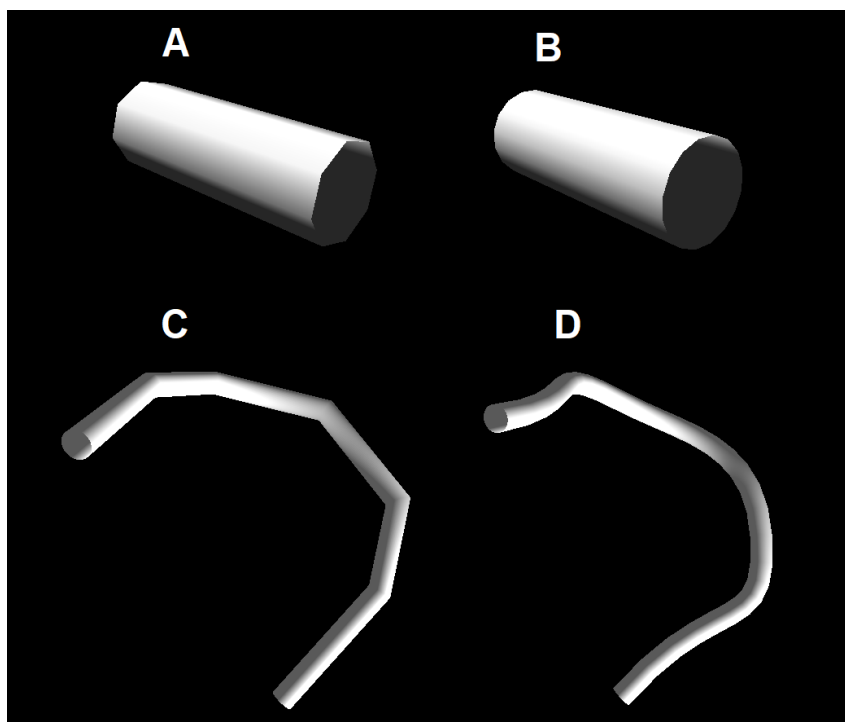
Slika 16. Prikaz poligona prilikom kreiranja okosnice



Slika 17. Geometrijski prikaz smjera okosnice

Glatkoća zakrivljenost pojedinačne, kratke cijevi određena je veličinom početnog n-terokuta, a glatkoća cjelokupne cijevi određena je brojem (gustoćom) točaka generiranih na krivulji. Na slici 18, na A dijelu slike nalazi se jedna cijev kreirana pomoću pet baznih točaka, tj. peterokuta., dok je na B dijelu slike cijev generirana pomoću 16 baznih točaka (šesnaesterokut). Na C

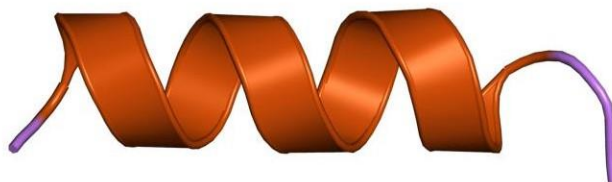
dijelu slike prikazana je cjelokupna cijevi koja prati krivulju, pri čemu je između kontrolnih točaka krivulje generirana samo jedna dodatna točka, dok je na D dijelu slike prikazana ista cijev ali ovaj put sa 10 generiranih točaka između dviju kontrolnih točaka krivulje.



Slika 18. Usporedba glatkoće cijevi

#### 4.2.3. $\alpha$ -zavojnice

Ovisno o programu, sekundarne strukture se prikazuju na različite načine, međutim, prikaz se većini slučajeva najčešće ostvaruje korištenjem vrpce (engl. *ribbons*). (Slika 19)

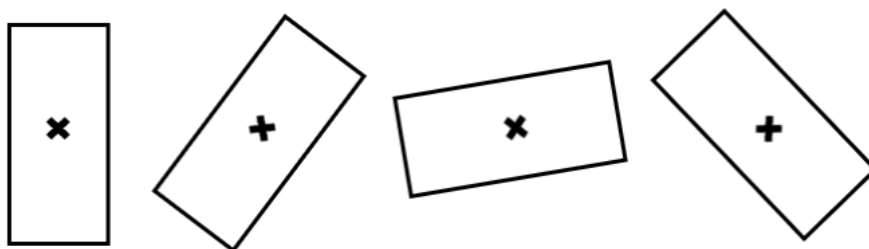


Slika 19. Izgled  $\alpha$ -zavojnice korištenjem vrpce [6]

Izrada ovakvih struktura temelji se isto na krivulji koja prolazi kroz C $\alpha$  atome svake aminokiseline. Atomi su dovoljno gusto raspoređeni tako da Catmull-

Rom krivulja, uz dovoljno velik koeficijent zakrivljenosti, čini kružnu zavojnicu, bez potrebe za dodatnim matematičkim popravcima.

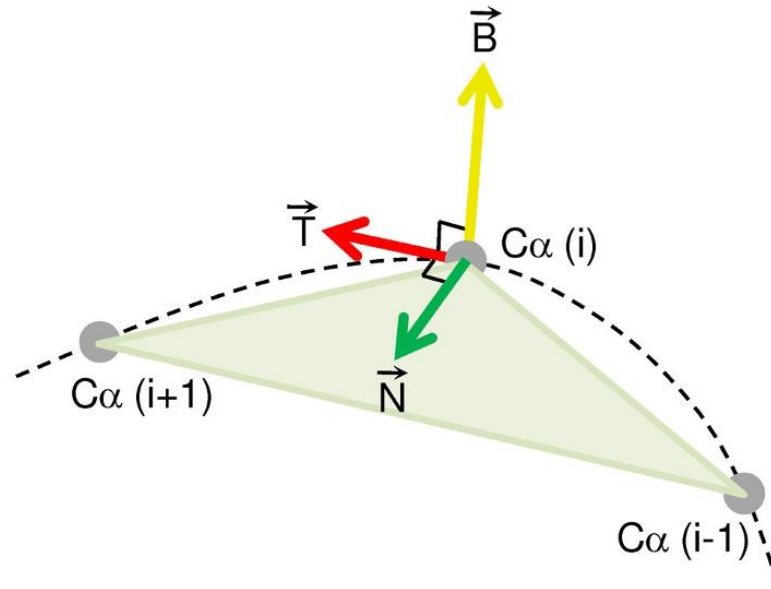
Ovakav tip strukture možemo generirati na način da provlačimo (engl. *sweeping*) određen geometrijski oblik, najčešće kvadrat ili izduženu elipsu po središnjoj krivulji. Problem tog postupka je u tome što u određenoj točki krivulje možemo dobiti samo smjer (tangentu) krivulje te ne možemo odrediti potpunu orijentaciju geometrijskog oblika kojeg provlačimo. Ako pogledamo sliku 20, znakom „x“ u sredini svakog kvadrata označen je vektor koji okomito „ulazi“ u svaki pravokutnik. Taj vektor možemo zamisliti da predstavlja tangentu krivulje u proizvoljnoj točki. Iz slike možemo vidjeti da postoji beskonačno mnogo načina rotacije tog pravokutnika, stoga nam je u točki krivulje, osim smjera, potreban još jedan vektor koji pokazuje pravilnu orijentaciju pravokutnika u proizvoljnoj točki – slično kao i *up* vektor kod kamere. Jednom kada imamo taj drugi vektor koji određuje orijentaciju, vektorskim produktom možemo dobiti i treći vektor i time u konačnici imamo kompletan lokalni koordinatni sustav u svakoj točki krivulje kojim možemo pravilno generirati točke kvadrata kojeg provlačimo po krivulji.



Slika 20. Neke od mogućih orijentacija pravokutnika okomitog na vektor smjera

Jedan način na koji možemo dobiti kompletan lokalni sustav u točki krivulje je sljedeći: za izračun lokalnog koordinatnog sustava u  $i$ -tom  $C\alpha(i)$  atomu potrebno je poznavati prethodni  $C\alpha(i-1)$  te sljedeći  $C\alpha(i+1)$  atom. Vektor smjera  $\vec{T}$  u  $C\alpha(i)$  možemo odrediti korištenjem derivirane formule za Catmull-Rom krivulju ili alternativno, možemo ga definirati kao vektor od  $C\alpha(i-1)$  prema  $C\alpha(i+1)$  atomu (Slika 21) [2]. Kao ključan drugi vektor možemo uzeti vektor

normale ravnine određene sa tim tria susjednim atomima ugljika Taj vektor je na slici označen kao žuti vektor  $\vec{B}$ . Konačno treći, bočni vektor, označen oznakom  $\vec{N}$ , na slici možemo dobiti kao vektorski produkt vektora  $\vec{B}$  i  $\vec{T}$  [2].



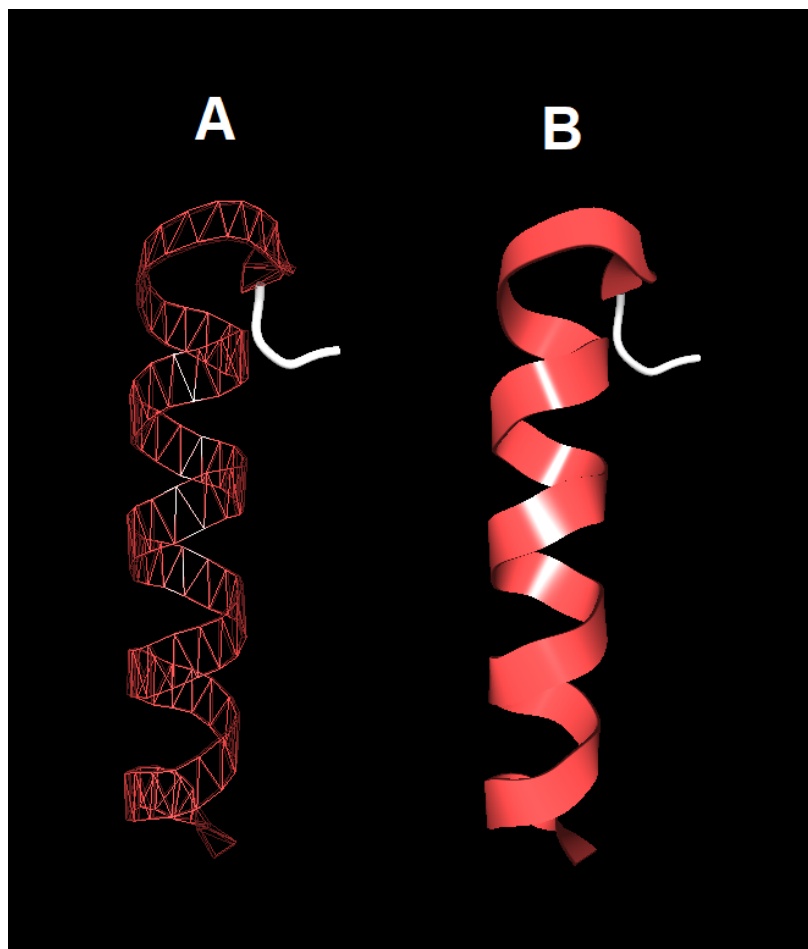
Slika 21. Određivanje lokalnog koordinatnog sustava za  $C\alpha(i)$  [2]

Kada izračunamo lokalne koordinatne sustave za dva susjedna atoma alfa ugljika, korištenjem jednostavne linearne interpolacije možemo izračunati pojedinačne lokalne koordinatni osi, i na taj način dobiti cjelokupni sustav za bilo koju točku između ta dva atoma. Linearnu interpolaciju za lokalne koordinatne sustave  $S_1$  i  $S_2$  za parametar  $t \in [0,1]$  dobijemo prema sljedećoj formuli [12] :

$$S(t) = (1 - t) \cdot S_1 + t \cdot S_2$$

Iz formule je jasno vidljivo za parametar  $t = 0$  dobijemo koordinatni sustav koji je jednak sustavu  $S_1$ , dok za vrijednost parametra  $t = 1$ , interpolirani sustav će biti jednak sustavu  $S_2$ . Ovakav pristup generira dovoljno dobre rezultate koji su vidljivi na slici 22. Na A dijelu slike je prikazan kostur (engl. *wireframe*) modela, dok je na B dijelu slike prikazan puni model zavojnice.



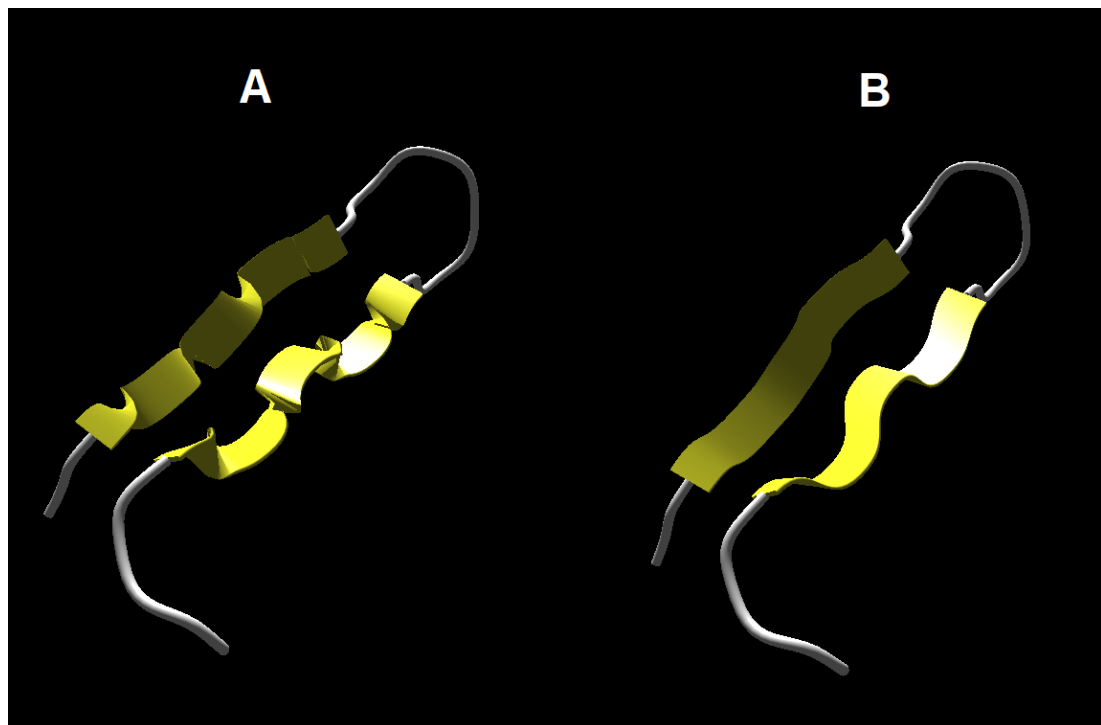


Slika 22. Izgled zavojnice dobivene linearnom interpolacijom lokalnih koordinatnih sustava C $\alpha$  atoma

#### 4.2.4. $\beta$ -lanci

$\beta$ -lanci, isto se kao i  $\alpha$ -zavojnice, najčešće prikazuju kao vrpce konstruirane pomoću krivulje koja prolazi kroz alfa atome ugljika. stoga se ranije opisana tehnika stvaranja vrpca može primijeniti i za generiranje beta lanaca, no uz jednu manju promjenu. Naime, prilikom generiranja lokalnih koordinatnih sustava za svaki C $\alpha$  atom koji čini  $\beta$ -lanac, vrlo često se dogodi da se susjedni sustavi drastično razlikuju, najčešće u tome da su koordinatne osi susjednog sustava zarotirane za 180° oko vektora smjera pružanja (z-os). [2] Prilikom generiranja zavojnice ovo nije bio problem jer se točke zavojnice cijelo vrijeme pružaju u istu (desnu) stranu. Ovaj problem generira vrpca koja je vrlo nazubljena (Slika 23, A dio). Kako bi ispravili ovaj problem, potrebno je u algoritam dodati dodatnu provjeru koja provjerava ako je veličina kuta između

bočnih osi (x-os) dvaju susjednih lokalnih koordinatnih sustava veća od  $90^\circ$  [2]. U slučaju da je, rotiramo jedan od sustava za  $180^\circ$  oko njegove z-osi [2]. Konačno, ova promjena rezultira glatkim modelom  $\beta$ -lanaca prikazanim na slici 23, B dio.



Slika 23. Ispravljanje  $\beta$ -lanca

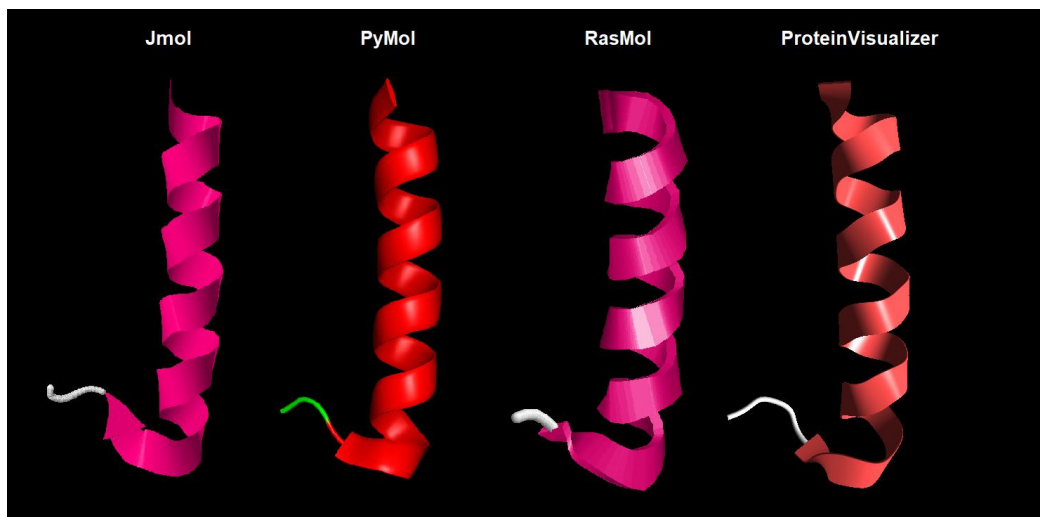
## 5. Usporedba s postojećim alatima

Popis svih alata koji mogu prikazati tercijarnu i kvartarnu strukturu proteina poprilično je dugačak – od osnovnih alata čija je namjena općeniti prikaz molekula, pa do kompleksnih alata koji su specijaliziraniji za proteine. U nastavku su navedeni i opisani neki od popularnijih alata te je na slikama 24, 25, 26 i 27 prikazana usporedba tih alata sa rezultatima koji su dobiveni korištenjem, ovdje razvijenog, ProteinVisualizer alata.

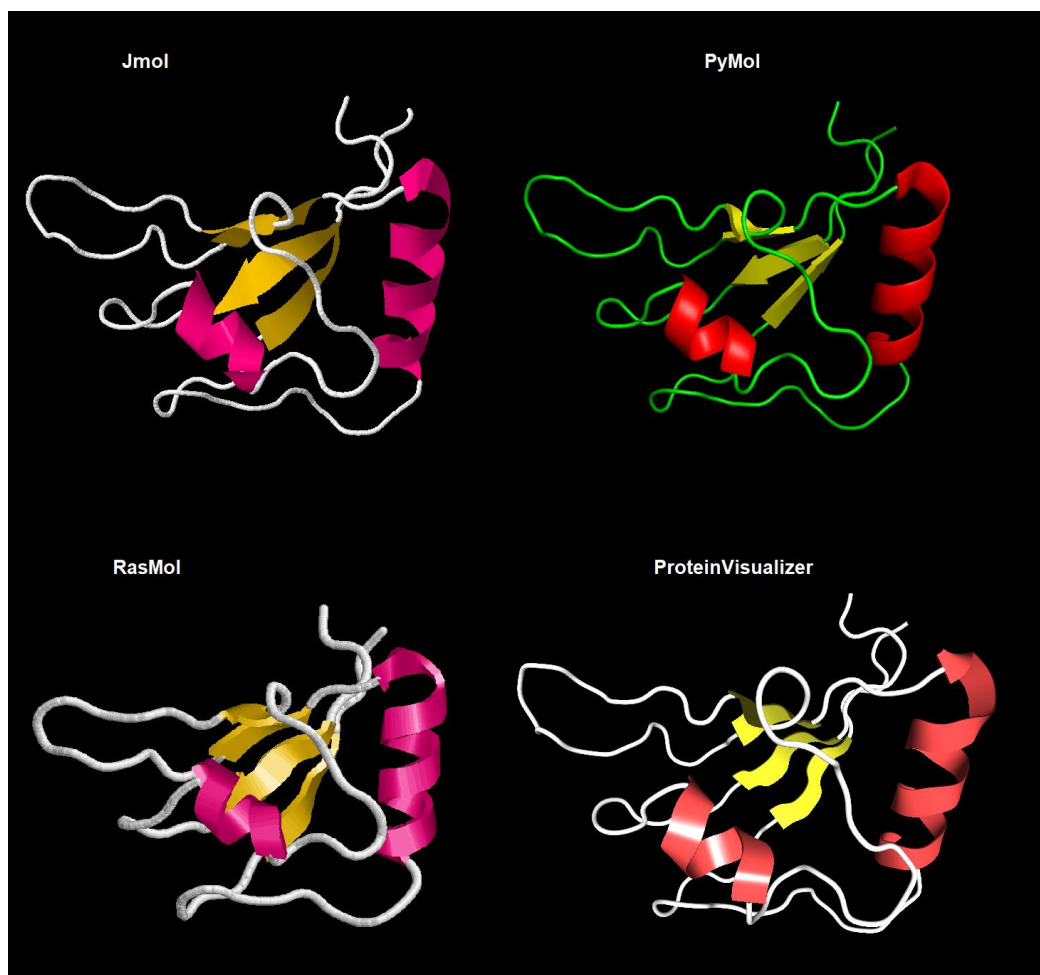
RasMol je jedan od starijih programa, razvijen još 1993. Povijesno je vrlo važan jer je svojedobno bio vrlo dobro optimiziran te je omogućio vizualizaciju na tada prosječnim računalima. Prije njega, za pokretanje bilo kojeg programa za vizualizaciju makromolekula bilo je potrebno imati radnu stanicu ili iznimno jako osobno računalo, stoga proučavanje proteina nije bilo lako dostupno kao danas. Za uporabu RasMola, razvijen je i skriptni jezik kojim se mogu obavljati razne funkcije poput odabira prikaza određenog lanca proteina, promjena boje sekundarnih struktura i slično. Isti jezik su preuzeli alati Jmol i Sirius [16].

Jmol je alat pisan u Javi inicijalno pušten 2001, no i dalje se redovito ažurira. Kako je pisan u Javi, njegova glavna odlika je podržanost na širokom rasponu računala. Kako je ranije spomenuto, Jmol koristi isti skriptni jezik za korištenje kao i RasMol, no omogućeno je korištenje preko grafičkog sučelja. Razvijeno je i proširenje koje omogućuje integraciju Jmola s web stranicama, dok je za računala koja nemaju Javu, razvijena je i verzija koja se temelji na JavaScriptu [1].

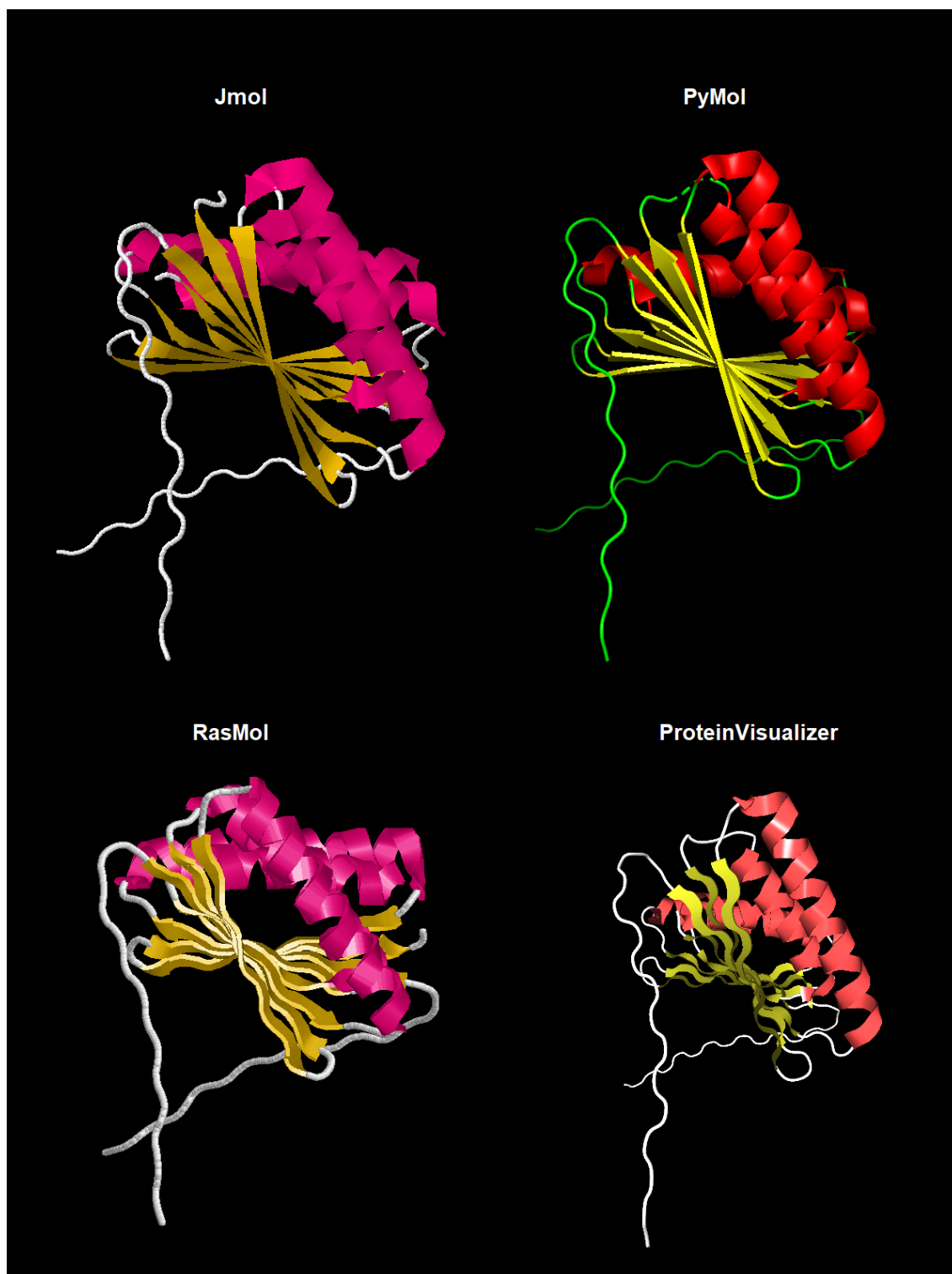
PyMOL je trenutno jedan od najpopularnijih alata za vizualizaciju strukture proteina. Otvorenog je koda (engl. *open-source*), a sponzoriran od strane korisnika. Razvijen je u Python programskom jeziku iz korištenja OpenGL API-ja [17]. PyMOL se može, osim za detaljnu analizu proteina, koristiti i za izradu kvalitetnih slika proteina, filmova, izradu i modeliranje molekula, uređivanje i promjenu postojećih molekula i još mnogo toga. Kako je alat napravljen u Python-u, te je uz to otvorenog koda, alat je, po potrebi, lako moguće proširiti nekim dodatnim Python paketima [18].



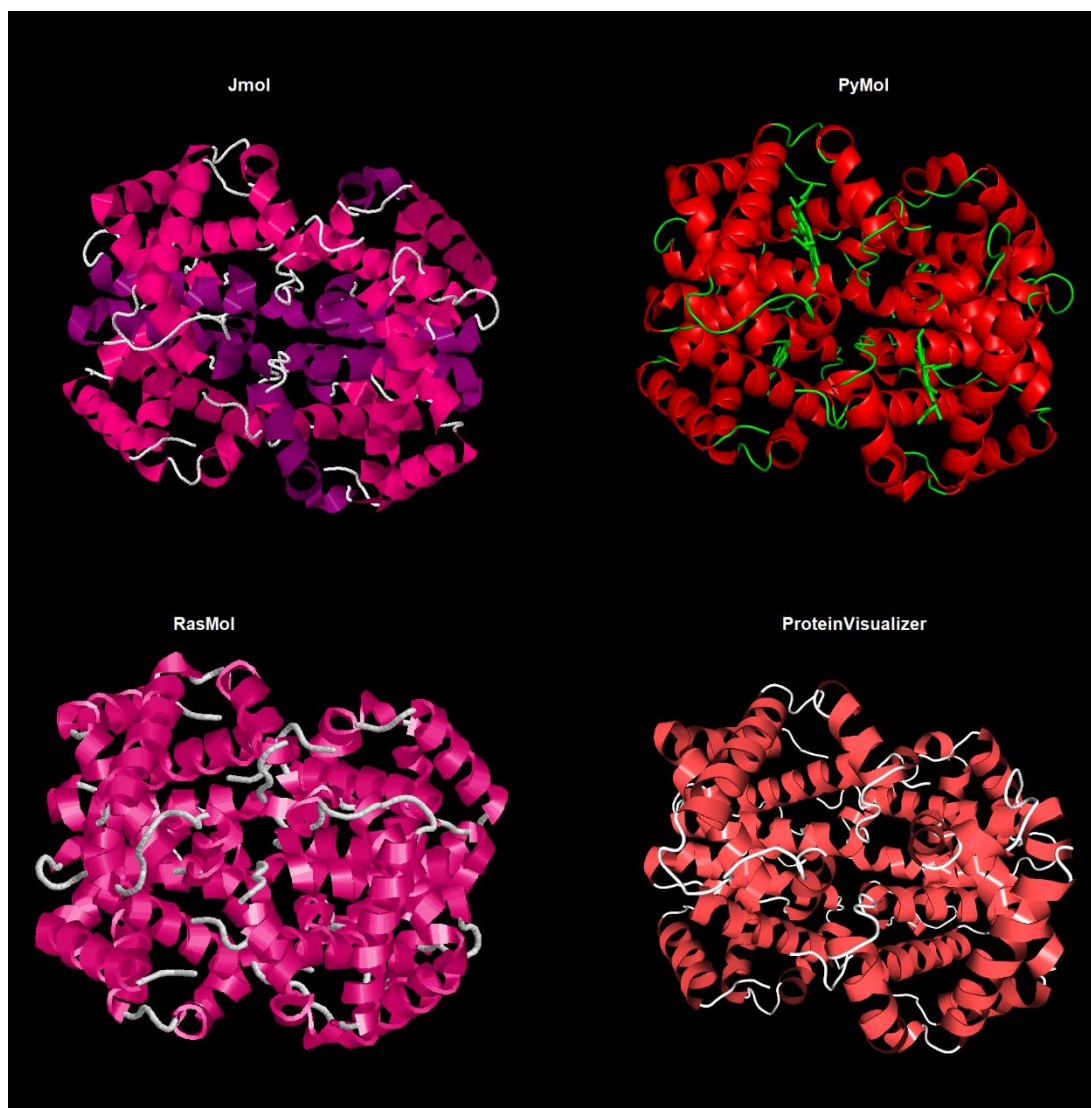
Slika 24. Usporedba proteina 6NIV



Slika 25. Usporedba proteina 5ZSY



Slika 26. Usporedba proteina 6NUK



Slika 27. Usporedba proteina 1A3N

## 6. Zaključak

Analiza i proučavanje proteina danas se u znatnoj mjeri radi na računalima te je alat za vizualni prikaz strukture proteina jedan od ključnih alata. Kako bi mogli izraditi jedan takav alat, potrebno je imati osnovna znanja iz biologije, kemije, matematike te, naravno, računalnih znanosti.

Kako bi uopće mogli pristupiti problemu, prvo je potrebno upoznati se s različitim vrstama struktura proteina. Zbog jednostavnijeg proučavanja, proteini se proučavaju na četiri razine – primarnoj, sekundarnoj, tercijarnoj i kvartarnoj. Kada su nam jasne osnove strukture proteina, na *Protein Data Bank* repozitoriju možemo pronaći i preuzeti zapise brojnih proteina. Proteini se danas najčešće zapisuju u mmCIF format datoteke, dok je stariji PDB službeno napušten, ali zbog dugačke povijesti korištenja i dalje je vrlo često korišten.

Izrada programa za vizualizaciju započinje parsiranjem ulazne datoteke sa zapisom strukture iz kojeg se potom dobiju sve potrebne informacije i proteinu potrebne za daljnji rad. Generiranje okosnice jednog lanca proteina radi se na način da se kroz sve C $\alpha$  atome svake aminokiseline provuče krivulja koja potom služi kao baza za generiranje deblje zaobljenje cijevi. Sekundarne strukture poput  $\alpha$ -zavojnice i  $\beta$ -lanaca najčešće se prikazuju kao vrpce koje se također temelje na krivulji koja prolazi kroz C $\alpha$  atome, dok konačni vizualni prikaz dobijemo tako da po toj krivulji provlačimo oblik pravokutnika ili izdužene elipse.

Konačno, kada sve elemente spojimo zajedno, dobijemo konačan model proteina koji se potom stavlja u središte fokusa kamere koja se, prema korisnikovim željama može micati kako bi se mogao istražiti svaki kutak proteina.

## 7. Literatura

- [1] J. Xiong, Essential Bioinformatics, New York: Cambridge University Press, 2006.
- [2] J. R. Weber, "ProteinShader: illustrative rendering of macromolecules," BioMed Central, Cambridge, 2009.
- [3] "Bjelančevine," 17 2 2019. [Online]. Available: <https://hr.wikipedia.org/wiki/Bjelan%C4%8Devine>. [Accessed 19 6 2019].
- [4] B. Mildner, "Aminokiseline, peptidi, te primarna struktura proteina," [Online]. Available: [https://www.pmf.unizg.hr/\\_download/repository/14obk-p4-aminokiseline\\_i\\_primarna\\_struktura.pdf](https://www.pmf.unizg.hr/_download/repository/14obk-p4-aminokiseline_i_primarna_struktura.pdf). [Accessed 19 6 2019].
- [5] B. Mildner, "Proteini i njihove trodimenzionalne," [Online]. Available: [https://www.pmf.unizg.hr/\\_download/repository/14obk-p5-strukture\\_proteina.pdf](https://www.pmf.unizg.hr/_download/repository/14obk-p5-strukture_proteina.pdf). [Accessed 19 6 2019].
- [6] Khan Academy, "Orders of protein structure," Khan Academy, [Online]. Available: <https://www.khanacademy.org/science/biology/macromolecules/proteins-and-amino-acids/a/orders-of-protein-structure>. [Accessed 19 6 2019].
- [7] Zavod za fiziku i biofiziku, "Nuklearna magnetska rezonancija," Medicinski fakultet sveučilišta u Zagrebu, [Online]. Available: <http://physics.mef.hr/Predavanja/nmr/prva.html>. [Accessed 19 6 2019].



- [8] "File Formats and the PDB," wwPDB, [Online]. Available: <http://www.wwpdb.org/documentation/file-formats-and-the-pdb>. [Accessed 19 6 2019].
- [9] "mmCIF Format," [Online]. Available: [https://www.rcsb.org/pdb/static.do?p=file\\_formats/mmcif/index.html](https://www.rcsb.org/pdb/static.do?p=file_formats/mmcif/index.html). [Accessed 18 6 2019].
- [10] "About the PDB Archive and the RCSB PDB," [Online]. Available: [https://www.rcsb.org/pdb/static.do?p=general\\_information/about\\_pdb/index.html](https://www.rcsb.org/pdb/static.do?p=general_information/about_pdb/index.html). [Accessed 19 6 2019].
- [11] M. Čupić and Ž. Mihajlović, Interaktivna računalna grafika kroz primjere u OpenGL-u, Zagreb, 2016.
- [12] "Krivulja," Wikipedija, 15 4 2019. [Online]. Available: <https://hr.wikipedia.org/wiki/Krivulja>. [Accessed 19 6 2019].
- [13] I. Zeid, "Geometric Modeling - Parametric Representation of Synthetic Curves," 2018. [Online]. Available: <https://en.ppt-online.org/337205>. [Accessed 26 6 2019].
- [14] "cartography for swiss higher education," 26 1 2012. [Online]. Available: [http://www.e-cartouche.ch/content\\_reg/cartouche/graphics/en/html/Curves\\_learningObject2.html](http://www.e-cartouche.ch/content_reg/cartouche/graphics/en/html/Curves_learningObject2.html). [Accessed 26 6 2019].
- [15] "Centripetal Catmull–Rom spline," Wikipedia, 9 6 2019. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Centripetal\\_Catmull%E2%80%93Rom\\_spline&id=901014294](https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Centripetal_Catmull%E2%80%93Rom_spline&id=901014294). [Accessed 19 6 2019].
- [16] E. Catmull and R. Rom, "A class of local interpolating splines," Academic Press, New York, 1974.

- [17] J. Armstrong, "Catmull-Rom Splines," January 2006. [Online]. Available: <http://algorithmist.net/docs/catmullrom.pdf>. [Accessed 19 6 2019].
- [18] "RasMol," 23 12 2018. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=RasMol&id=875121820>. [Accessed 19 6 2019].
- [19] "PyMOL," 24 12 2018. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=PyMOL&id=875123728>. [Accessed 19 6 2019].
- [20] S. Yuan, S. H. C. Chan and Z. Hu, "Using PyMOL as a platform for," John Wiley & Sons, Ltd, 2017.

# Sažetak

## Vizualizacija strukture proteina

Proteini su sastavni dio svake stanice, a sastoje se od jednog ili više lanaca aminokiselina. Prilikom njihovog proučavanja, možemo govoriti o četiri razine struktura: primarnoj, sekundarnoj, tercijarnoj te kvartarnoj. Niz aminokiselina kao takav, bez proučavanja međusobne interakcije unutar proteina, čini primarnu strukturu. Sekundarnu strukturu proteina čine nekoliko različitih tvorevina koje nastaju na jednom lancu od kojih su najvažnije  $\alpha$ -zavojnice i  $\beta$ -lanci. Tercijarna struktura opisuje funkciju jednog lanca uz prisustvo sekundarnih struktura, dok proteini koji imaju više od jednog lanca imaju i kvartarnu strukturu koja proučava međusobni utjecaj tih lanaca.

U sklopu ovog rada razvijen je alat ProteinVisualizer čija je svrha generiranje i prikazivanje strukture proteina. Program pri pokretanju izvlači podatke o proteinu koji su zapisani u ulaznoj PDB datoteci. Za osnovu modela, kreira se krivulju koja prolazi kroz središnji atom ugljika ( $C\alpha$ ) svake aminokiseline. Korištenjem te krivulje, te uz pomoć linearne algebre, generira se cijev, koja predstavlja okosnicu proteina, i vrpce, kojima se najčešće predstavljaju  $\alpha$ -zavojnice i  $\beta$ -lanci. Konačno, spajanjem svih tih generiranih struktura, dobije se kompletan model proteina.

Rezultati koje alat generira usporedivi su s rezultatima drugih popularnih alata za vizualizaciju, što u konačnici potvrđuje ispravnost ovog postupka.

Ključne riječi: protein, vizualizacija, struktura proteina, PDB, krivulje, Catmull-Rom, modeliranje, makromolekule, okosnica proteina,  $\alpha$ -zavojnice,  $\beta$ -lanci, DirectX

# Summary

## Protein structure visualization

Proteins are part of every cell and they consist of one or more chains of amino acid residues. In protein analysis, there are four levels of protein structure: primary, secondary, tertiary and quaternary. Plain sequence of amino acids residues in a polypeptide chain forms primary structure. Secondary structure refers to local folded structures in a single chain, among which are the most important  $\alpha$ -helices and  $\beta$ -sheets. Tertiary structure represents function of a single chain in presence of secondary structures, while interaction between multiple polypeptide chains forms a quaternary structure.

In this thesis, ProteinVisualizer software has been developed which can generate and visualize protein structure,. Program starts by parsing information about protein model from PDB file. For the model basis, curve that goes through central carbon atom ( $C\alpha$ ) of every amino acid residue is created. By using this curve, and with help of linear algebra, tube that represents protein wireframe, but also the ribbons that represents  $\alpha$ -helices and  $\beta$ -sheets, can be generated. Finally, by combining all these structures, complete protein model is created.

Results of the developed program are comparable to results of other popular tools for protein visualization, which, in the end, confirms correctness of this procedure.

Keywords: protein, visualization, protein structure, PDB, curves, Catmull-Rom, modeling, macromolecules, protein wireframe,  $\alpha$ -helix,  $\beta$ -sheets, DirectX

## Dodatak A: Upute za uporabu programa

Izvorni kod programa moguće je pronaći na priloženom CD-u ili ga je moguće preuzeti s GitHub repozitorija (<https://github.com/tonisente/ProteinVisualizer>).

Program je napravljen kao Visual Studio Project te pisan u C++ programskom jeziku uz korištenje DirectX API-ja, stoga je njegova uporaba ograničena na Windows platformu. Za pokretanje programa, potrebno je korištenjem terminala, pozicionirati se u mapu „Run“ u kojoj se nalazi binarna datoteka „ProteinVisualizer.exe“ spremna za pokretanje. Prilikom pokretanja, programu je potrebno obavezno predati relativnu putanju do nekog modela proteina. Uz obavezan model proteina, programu je moguće, ali nije nužno, preko zastavica namjestiti određene parametre za vizualizaciju. Parametri koje je moguće namjestiti su sljedeći:

- `--helixRGB=x,y,z` → postavlja RGB vrijednost boje  $\alpha$ -zavojnice. x, y i z su decimalni brojevi u rasponu od [0, 1] (pretpostavljena vrijednost je 1.0,0.3,0.3)
- `--sheetRGB=x,y,z` → postavlja RGB vrijednost boje  $\beta$ -lanca. x, y i z su decimalni brojevi u rasponu od [0, 1] (pretpostavljena vrijednost je 1.0,1.0,0.3)
- `--wireRGB=x,y,z` → postavlja RGB vrijednost boje okosnice. x, y i z su decimalni brojevi u rasponu od [0, 1] (pretpostavljena vrijednost je 1.0,1.0,1.0)
- `--type=x` → postavlja tip prikaza. x je cijeli broj u rasponu od [1,3], a opis svake vrijednosti je dan u nastavku: (pretpostavljeni tip je 1)
  - `x = 1` → prikazuje se protein u tercijarnoj strukturi
  - `x = 2` → prikazuje se samo zakrivljena okosnica proteina
  - `x = 3` → prikazuje se samo ravna okosnica proteina

Primjeri pokretanja:

- `ProteinVisualizer.exe ProteinModels/2hiu.pdb --helixRGB=1.0,0.5,0.1 --wireRGB=0.2,0.2,0.2`
- `ProteinVisualizer.exe ProteinModels/2hiu.pdb --type=2`