Gymnázium Dubnica nad Váhom

Školská 387, 018 41 Dubnica nad Váhom

INVERZNÁ KINEMATIKA PRE KAŽDÉHO

Stredoškolská odborná činnosť

Odbor č. 11 – Informatika

Mesto: Kvašov

Rok: 2022

Riešiteľ: Adrián Hochla

Ročník štúdia: tretí

Gymnázium Dubnica nad Váhom

Školská 387, 018 41 Dubnica nad Váhom

INVERZNÁ KINEMATIKA PRE KAŽDÉHO

Stredoškolská odborná činnosť

Odbor č. 11 – Informatika

Mesto: Kvašov

Rok: 2022

Riešiteľ: Adrián Hochla

Ročník štúdia: tretí

Konzultant: Ing. Michal Vlk

Čestné vyhlásenie

Vyhlasujem, že prácu stredoškolskej odbornej činnosti na tému „Inverzná kinematika pre každého" som vypracoval samostatne, s použitím uvedených literárnych zdrojov. Prácu som neprihlásil a ani neprezentoval v žiadnej inej súťaži, ktorá je pod gestorstvom MŠMVVaŠ SR. Som si vedomý dôsledkov, ak uvedené údaje nie sú pravdivé.

V Kvašove dňa 24.februára 2022 .....

Adrián Hochla

Poďakovanie

Na úvod by som sa rád poďakoval skvelej komunite ľudí na internete, ktorí obklopujú svet informatiky a sú ochotní pomôcť pri každej príležitosti. Neviem si predstaviť prácu na akomkoľvek projekte bez pomoci iných, a práve komunita programátorov na internete mi pomohla zo všetkých najviac.

Ďalej patrí moja vďaka konzultantovi tejto práce Ing. Michalovi Vlkovi, ktorý, hlavne v skorých fázach projektu, poskytol svoje odborné rady.

Obsah

Zoznam tabuliek, grafov a ilustrácií

Zoznam skratiek, značiek a symbolov

IK – inverzná kinematika

DOF – stupne voľnosti (degrees of freedom)

DH – Denavit - Hartenberg

CCD – cyklický súradnicový zostup (cyclic coordinate descent)

Úvod

Táto práca popisuje fungovanie webovej stránky zameranej na riešenie problému inverznej kinematiky, dostupnej na adrese www.ikforeveryone.com. Cieľom bolo generalizovať riešenie tohto problému pre všeobecný kinematický diagram, to znamená pre akúkoľvek robotickú ruku. Poskytnutie užívateľovi voľnosť v tvorbe svojej robotickej ruky v prehľadných 3D scénach a plánovanie lineárnej trajektórie s množstvom nastavení.

Spoločnosti, ktoré robotické ruky vyrábajú a distribuujú, majú svoj softvér na ich ovládanie, ale pre bežného užívateľa nie sú využiteľné. Je to z toho dôvodu, že riešenie prakticky funguje len pre ich robotické ruky. Medzi ďalšie problémy patrí napríklad nutnosť za softvér platiť a náročnosť tvorby vlastných robotických rúk. Bolo preto potrebné vymyslieť program, ktorý by bol aplikovateľný nie len na priemyselné robotické ruky, ale aj na akékoľvek robotické zreťazenie užívateľa, ktoré môže byť prepojené napríklad mikrokontrolérom.

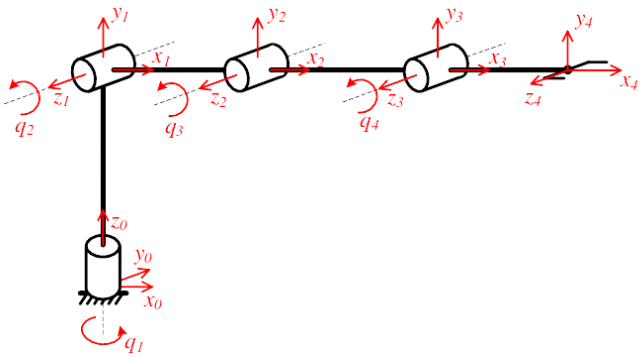
Sám som pracoval na projekte kde som potreboval vyriešiť problém inverznej kinematiky špecifickej robotickej ruky, ale žiaden z dostupných softvérov nebol nápomocný. Preto som sa rozhodol vytvoriť riešenie na webe, ktoré poskytne užívateľovi funkcionalitu, ktorú potrebuje a to kedykoľvek a kdekoľvek.

1 Problematika a prehľad literatúry

Pohyb kinematického reťazca, či už ide o robota alebo animovanú postavu, je modelovaný kinematickými rovnicami reťazca. Tieto rovnice definujú konfiguráciu reťazca z hľadiska jeho kĺbových uhlov. Dopredná kinematika používa parametre spoja na výpočet konfigurácie reťazca a inverzná kinematika obráti tento výpočet na určenie parametrov spoja, ktoré dosiahnu požadovanú konfiguráciu.

Cieľom ale nie je získať jednu konfiguráciu (uhly pre každý kĺb), ktorá reťazec dostane do požadovanej pozície. Cieľom je naplánovať takú trajektóriu, aby sa reťazec dostal do požadovanej pozície lineárne. Je potrebné vypočítať IK na množine bodov, ktorými je úsečka z aktuálnej pozície do požadovanej.

Pre výpočty inverznej kinematiky bolo zvolené geometrické riešenie. Aby sme mohli reprezentovať reťazec matematicky, potrebujeme k tomu homogénne transformačné matice. Tie opisujú transformáciu a transláciu každého kĺbu v závislosti od predošlého. Pre odvodenie transformačných matíc potrebujeme DH parametre z ktorých je to najjednoduchšie.



Obrázok 1: Príklad kinematického reťazca [12]

1.1 DH konvencia

Pre štandardizovanie súradnicových rámcov pre priestorové prepojenia zaviedli Jacques Denavit a Richard Hartenberg, v roku 1955, svoju vlastnú konvenciu. Zahŕňa pravidlá pre vytváranie referenčných rámcov a 4 parametre z ktorých vieme odvodiť homogénne transformačné matice kinematického reťazca. Postup zahŕňa nájdenie súradníc prepojenia a ich použitie na nájdenie homogénnej transformačnej matice 4×4 zloženej zo štyroch samostatných podmatíc na vykonanie transformácií z jedného súradnicového rámca do

jeho susedného súradnicového rámca. DH notácia je cenná pre oblasť robotiky, v ktorej možno robotické manipulátory modelovať ako články tuhých telies. Väčšina priemyselných robotických manipulátorov sú kinematické reťazce s otvorenou slučkou pozostávajúce zo základne, kĺbov, článkov a end-efektora. Schopnosť ovládať end-efektor robota v trojrozmernom priestore vyžaduje znalosť vzťahu medzi kĺbmi robota a polohou a orientáciou endefektora. Vzťah si vyžaduje použitie a pochopenie rotačnej matice a translačného vektora. [2]

1.1.1 DH referenčné rámce

Referenčné rámce (osi x, y, z) ako na **Obrázok *1*** priraďujeme kĺbom podľa pravidiel stanovenými v konvencii. Tie sú:

- z_n je osou rotácie pre rotačné kĺby
- x_n musí byť kolmá na z_n rovnako ako na z_{n-1}
- y_n je určená z osí z_n a x_n, podľa pravidla pravej ruky
- x_n musí pretnúť os z_{n-1} (neplatí pre nultý rámec)

Výber rôznych súradnicových rámcov nie je unikátny, to znamená, že existuje viac spôsobov priradenia referenčných rámcov s rovnakým výsledným odvodením homogénnych transformačných matíc. [3]

1.1.2 Tabuľka DH parametrov

Každý zo štyroch parametrov: theta (θ), alpha (α), r, d, je zodpovedný za poskytnutie nejakej informácie o reťazci:

- **θ** – uhol ktorý zviera os x_{n-1} s osou x_n okolo osi z_{n-1},
- **α** – uhol ktorý zviera os z_{n-1} s osou z_i okolo spoločnej normály,
- **r** – dĺžka spoločnej normály, alebo vzdialenosť medzi osou z_{n-1} a osou x_n v smere tejto osi,
- **d** – posun pozdĺž osi z_{n-1} ku spoločnej normále, alebo vzdialenosť medzi osou z_{n-1} a osou x_n, v smere z_{n-1}

n	θ	α	r	d
1	θ_1	90°	0cm	21,8cm
2	θ_2	0°	19,3cm	0cm
3	θ_3 -90°	0°	14,2cm	0cm

Tabuľka 1: Príklad DH parametrov

V stĺpci tabuľky θ sa k hodnote ktorú si z reťazca odvodíme pridáva ešte uhol θ_n , ktorý nadobúda hodnotu aktuálnej rotácie kĺbu.

Index osí základne reťazca je spravidla 0, pretože DH tabuľka začína s $n = 1$. pre vyplnenie tabuľky dosádzame hodnoty n z prvého stĺpca. Maximálna hodnota n bude vždy o jeden menšia ako celkový počet kĺbov v reťazci.

Z vyplnenej tabuľky vieme odvodiť homogénne transformačné matice ktoré majú nasledovnú formu:

$$H_n^{n-1} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n \cos\alpha_n & \sin\theta_n \sin\alpha_n & r_n \cos\theta_n \\ \sin\theta_n & \cos\theta_n \cos\alpha_n & -\cos\theta_n \sin\alpha_n & r_n \sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Rovnica 1: Homogénna transformačná matica

Kde R je podmatica 3×3 opisujúca rotáciu a T je podmatica 3×1 opisujúca transláciu (posun). Takúto maticu budeme mať pre každý kĺb v závislosti od predošlého. Pre reťazec so štyrmi DOF - štyri kĺby a end efektor (príklad **Obrázok *1***), budeme mať štyri transformačné matice H_1^0, H_2^1, H_3^2, H_4^3. Pre získanie jednej transformačnej matice v tvare H_4^0, ktorá bude opisovať vzťah medzi základňou reťazca a end efektorom, nám stačí matice vynásobiť. [4]

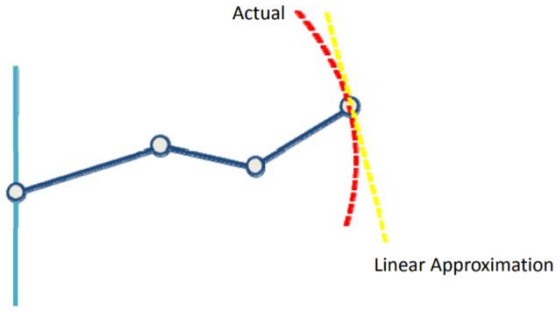
1.2 Jakobiho matica

Jakobiho matica je matica parciálnych derivácií vektorovej funkcie. Ak je táto matica štvorcová, nazývame ju Jakobiho determinant. Tento determinant je rozsiahle využívaný

pri výpočtoch viacrozmerných integrálov. Jakobiho matica má najväčšie využitie pri lineárnych aproximáciách. [5]

$$J = \begin{bmatrix} \frac{\partial px}{\partial \theta_A} & \frac{\partial px}{\partial \theta_B} & \frac{\partial px}{\partial \theta_C} \\ \frac{\partial py}{\partial \theta_A} & \frac{\partial py}{\partial \theta_B} & \frac{\partial py}{\partial \theta_C} \\ \frac{\partial pz}{\partial \theta_A} & \frac{\partial pz}{\partial \theta_B} & \frac{\partial pz}{\partial \theta_C} \end{bmatrix}$$

Rovnica 2: Všeobecný zápis Jakobiho matice



Obrázok 2: Lineárna aproximácia

Každý člen v Jakobiho matici predstavuje, ako zmena špecifického kĺbového uhla ovplyvňuje priestorové umiestnenie end-efektora. E = <px,py,pz>. Prvý člen ukazuje, ako veľmi by sa zmenila poloha end efektora pozdĺž osi X, ak by sa uhol kĺbu A zmenil o diferenciálu hodnotu, napríklad 0,00001. Prvý stĺpec ukazuje, ako veľmi by sa zmenila poloha end-efektora v súradnicovom priestore X-Y-Z, ak by sa uhol kĺbu A zmenil o diferenciálnu hodnotu. Matica v **Rovnica 1** má 3 stĺpce, to znamená, že reťazec ktorý reprezentuje, má 3 kĺby. Jeden stĺpec pre každý kĺb. [6]

Členy do Jakobiho matice vypĺňame pomocou predošle získaných homogénnych transformačných matíc. Podľa nasledovného vzoru:

$$R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (d_n^0 - d_{i-1}^0)$$

Rovnica 3: Všeobecný člen Jakobiho matice

Po doplnení pre 3DOF reťazec dostaneme nasledovnú maticu:

$$J = \begin{bmatrix} R_0^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (d_3^0 - d_0^0) & R_1^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (d_3^0 - d_1^0) & R_2^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (d_3^0 - d_2^0) \end{bmatrix}$$

Rovnica 4: Vzor výslednej Jakobiho matice pre 3DOF reťazec

1.2.1 Pseudoinverzná metóda

Medzi metódy výpočtov inverznej kinematiky patrí napríklad inverzná Jakobiho matica, cyklický súradnicový zostup (CCD) alebo transpozičná Jakobiho matica. Najväčším problémom inverznej metódy je, že nie je definovaná pre neštvorcové matice a cieľom tohto projektu je generalizovať riešenie pre n-stupňov voľnosti. [1]

CCD algoritmus sa implementuje jednoducho. Začína sa od end-efektora. Algoritmus meria rozdiel medzi polohou kĺbu a end-efektora. Potom vypočíta buď rotáciu alebo kvaternión, aby sa tento rozdiel znížil na nulu. Robí to pre každý kĺb, iteruje od end-efektora k nepohyblivému kĺbu v koreni kinematického reťazca.

S algoritmom CCD sú spojené dva typické problémy. V prvom rade je CCD iteratívna metóda, ktorá posúva kĺby v opačnom poradí. Vonkajšie kĺby sú otáčané skôr, čo spôsobuje, že pohyb pôsobí neprirodzene. Po druhé, keď sa kĺby v blízkosti end-efektora otáčajú viac ako kĺby v blízkosti nepohyblivého kĺbu, kinematický reťazec sa navinie sám do seba. [7]

Výhodou transpozičnej Jakobiho matice je rýchlosť jej výpočtov. Na druhej strane jej chýba potrebná presnosť, preto som zvolil pseudoinverznú metódu.

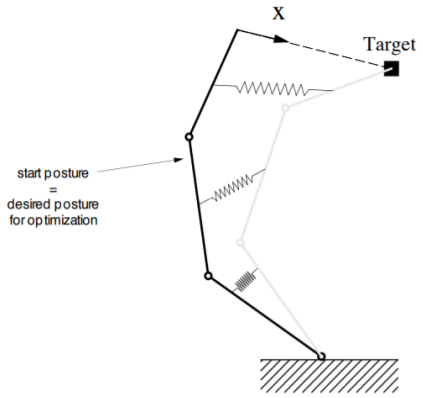
$$\Delta\theta = J^{\dagger}\,\vec{e}$$

Rovnica 5: Pseudoinverzná metóda

Kde $J^{\dagger}$ je pseudoinverzná matica ku Jakobiho matici J, $\vec{e}$ je požadovaná pozícia a $\Delta\theta$ zmena uhlov kĺbov.

Pseudoinverzná matica je najviac známa za zovšeobecnenie inverznej. Bežne sa využíva na výpočet najmenšieho štvorcového riešenia (least squares) sústavy lineárnych rovníc, ktoré nemajú riešenie. Je definovaná aj pre neštvorcové matice, čo znamená že vieme nájsť riešenie pre všeobecný diagram s n-stupňami voľnosti. Ďalej má nasledujúce pekné vlastnosti:

1. Predpokladajme, že $\vec{e}$ je v dosahu reťazca, čiže platí: $\vec{e} = J\,\Delta\theta$ tým pádom $\Delta\theta$ je vektor najmenšej veľkosti spĺňajúci rovnicu vyššie
2. Predpokladajme, že $\vec{e}$ nie je v dosahu reťazca, čiže: $\vec{e} = J\,\Delta\theta$ nemá riešenie; napriek tomu má $\Delta\theta$ vlastnosť minimalizovať veľkosť rozdielu $J\,\Delta\theta - \vec{e}$ čiže k požadovanej pozícií sa bude približovať. [1]



Obrázok 3: Intuícia pseudoinverznej metódy [13]

Po matematických úpravách **Rovnica 5** sa dostaneme ku vzťahu, ktorý bol použitý aj v algoritme v zdrojovom kóde:

$$\Delta\theta = J^{T}(J\,J^{T})^{-1}\,\vec{e}$$

Rovnica 6: Upravená pseudoinverzná metóda

Kde J^{T} je transpozičná Jakobiho matica.

1.3 Programovacie knižnice

1.3.1 Knižnica Flask

Webový aplikačný framework Flask je spolu s Django, najpopulárnejším riešením tvorby webových stránok v prostredí Python. V porovnaní s Django je Flask minimalistickejší a na funkčnosť stránky stačí jediný súbor. Je zameraný na zachovanie

jednoduchého, ale rozšíriteľného jadra. Flask som si vybral hlavne kvôli relatívne nízkemu počtu jednotlivých stránok v aplikácií. [8]

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/index')
def hello_world():
    return render_template('index.html')

if __name__ == '__main__':
   app.run()
```

Ukážka kódu 1: Inicializácia Flask aplikácie

`Route` dekoratér spája funkciu `hello_world` s URL adresou `/index`. Vždy keď prebehne presmerovanie na túto URL, privolá sa prislúchajúca funkcia.

Pre načítanie HTML súboru sa využíva funkcia `render_template` ktorá hľadá vami zadaný názov HTML súboru v priečinku templates. Flask je v názve priečinkov veľmi vyberavý, preto musíme pre správne fungovanie dodržať stanovenú súborovú štruktúru.

1.3.2 Knižnica Three.js

Možnosť vytvorenia komplexných kinematických reťazcov je užívateľovi ponúknutá vďaka knižnici Three.js. Umožňuje do scény umiestniť objekty, svetlá, kameru a ovládacie prvky. Túto scénu potom vykreslí na HTML prvok `canvas`. Využíva aplikačné rozhranie WebGL a 3D animácie akceleruje cez GPU. Medzi funkcionality tejto knižnice patria napríklad:

- Scény: pridávanie a odstraňovanie objektov za behu,
- Kamery: perspektívne alebo ortografické ,
- Svetlá: okolité, smerové, bodové,
- Prístup ku všetkým schopnostiam OpenGL Shading Language (GLSL),
- Geometria: rovina, kocka, guľa, torus, 3D text,
- Export a import: nástroje na vytváranie súborov JSON, kompatibilných s Three.js,
- Príklady: viac ako 150 príkladov kódovania plus fonty, modely a textúry.

Three.js sa často zamieňa s WebGL, pretože častejšie, ale nie vždy, ho používa na kreslenie v 3D. Prípady, pri ktorých sa WebGL nevyužíva sú veľmi špecifické. Ide napríklad o načítanie HTML prvkov v 3D, ktoré je tiež použité pre zadanie dĺžky spoju medzi kĺbmi. Používa sa na to CSS3D renderer. WebGL je veľmi nízkoúrovňový systém, ktorý kreslí iba body, čiary a trojuholníky. Three.js si zasa poradí s vecami, ako sú scény, svetlá, tiene, materiály, textúry. [9]

2 Ciele práce

Našim cieľom je vytvoriť webovú stránku, dostupnú čo najväčšiemu počtu ľudí, ktorá by pomohla pri riešení problému inverznej kinematiky všeobecného kinematického diagramu. Za čiastkové ciele považujme tvorbu jednotlivých stránok webovej aplikácie. Sú to:

- Dizajnér, kde si užívateľ pretvorí kinematický diagram,
- Inverzná kinematika, kde naplánuje trajektóriu svojho diagramu,
- Účet, kde môže upravovať svoje uložené diagramy,
- Pomoc pri prípadných otázkach využitia stránky.

Pre dizajnér je potrebné naštudovať si externú knižnicu Three.js a vymyslieť program, ktorý bude úspešne, jednoducho a presne reprezentovať akýkoľvek kinematický reťazec. Dáta z dizajnéra budú poháňať výpočty inverznej kinematiky.

Výzvou sekcie stránky Inverzná kinematika bude interaktívny 3D graf v ktorom sa užívateľovi jeho reťazec zobrazí. V tomto grafe je totiž potrebná aktualizácia požadovanej pozície, neustále meniace sa uhly konfigurácie a trajektória prejdená end-efektorom. Vnútorné fungovanie výpočtov za finálnou trajektóriou bude mať užívateľ v plnej kontrole. V pozadí toho všetkého budú prebiehať náročné výpočty inverznej kinematiky a preto je veľmi dôležité zvoliť presnú a rýchlu metódu výpočtov. Pracovanie IK algoritmu samotného bude na voľbe užívateľa, preto rozhranie, v ktorom bude pracovať musí byť priehľadné a funkčné.

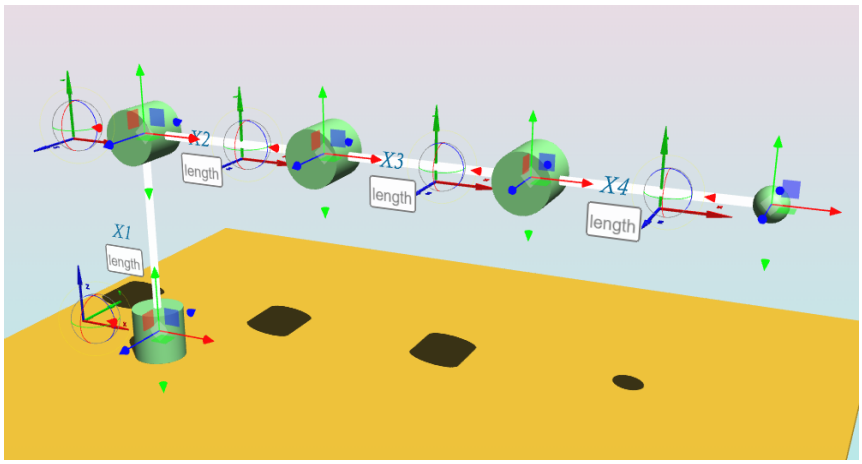
Je potrebné vytvoriť databázu do ktorej sa budú užívatelia prihlasovať a ukladať ich vytvorené reťazce. Musí im byť poskytnutá možnosť reťazec si kedykoľvek upraviť alebo vymazať, preto komunikácia s databázou musí byť bezproblémová.

Bude potrebné prísť na riešenie kombinácie technológií, programovacích jazykov a knižníc aby spolu pracovali v prospech jednoduchosti ovládania pre užívateľa.

3 Materiál a metodika

Jadro web stránky je napísané v programovacom jazyku Python kvôli náročnosti výpočtov IK. Externé knižnice pomáhajú s matematickými operáciami matíc a aj s vytvorením stránky samotnej. Bolo potrebné spojiť viac technológií aby umožnili tvorbu diagramov v 3D scénach, načítanie 3D grafu a prepojím s algoritmom výpočtov IK.

3.1 Dizajnér



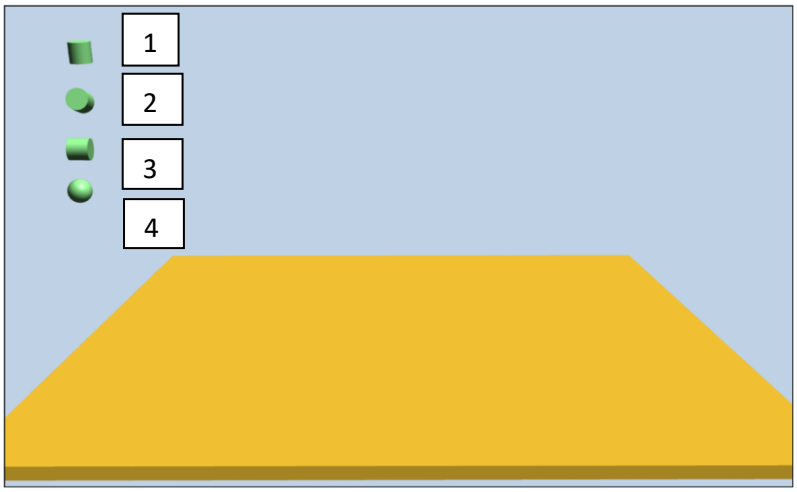
Obrázok 4: Diagram na web stránke

Každý diagram má podstavu na ktorej je umiestnený. Užívateľ má na výber zo štyroch možností jednotlivých kĺbov:
Uvažujme, že pomyselné osi o1, o2, o3 smerujú v tomto poradí doprava, hore a do vnútra strany

- 1.Os rotácie rotačného kĺbu 1 je o2,
- 2. Os rotácie rotačného kĺbu 2 je o3,
- 2. Os rotácie rotačného kĺbu 3 je o1,
- 4. End-efektor reťazca.

Kliknutím na kĺb ho užívateľ pridá do scény. Spolu so samotným kĺbom sa načíta aj referenčný rámec – osi x, y, z opisujúce jeho rotáciu. Kĺb spolu s jeho referenčným rámcom môže užívateľ v scéne voľne presúvať. Takisto sa načíta čiara spájajúca predošlý

rámec s terajším spolu so vstupom pre zadanie dĺžky spojnice medzi rámcami. Každý diagram končí end-efektorom.

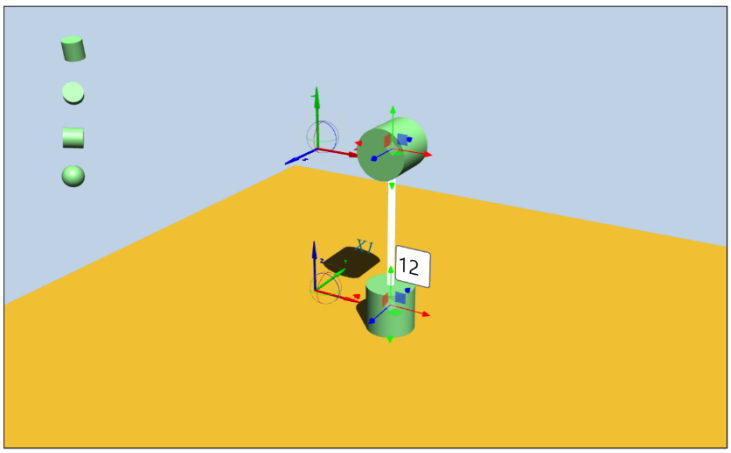


Obrázok 8: Prázdna scéna dizajnu

Šípkami priamo na kĺboch užívateľ premiestňuje kĺb, šípky s názvami osí sú objektom v scéne a užívateľ ich rotuje potiahnutím po kružniciach na nich pripevnených. Osi sa rotujú podľa DH-konvencie (DH konvencia) z ktorých je vygenerovaná DH tabuľka. V sekcií pomoc nájde užívateľ návod ako referenčné rámce rotovať, aby diagram pravdivo reprezentoval jeho robotický systém.

Každý diagram má na stránke svoje unikátne ID, ktoré si môže užívateľ uložiť do profilu, alebo, ak si nechce vytvárať účet, môže kedykoľvek ID zadať do vstupu pod tabuľkou a načítať si tak celý svoj kinematický diagram.

Pod tabuľkou je tlačidlo na presmerovanie „Výpočet IK“. Po kliknutí na toto tlačidlo užívateľa stránka zavedie s jeho vytvoreným diagramom do sekcie stránky – Inverzná kinematika.

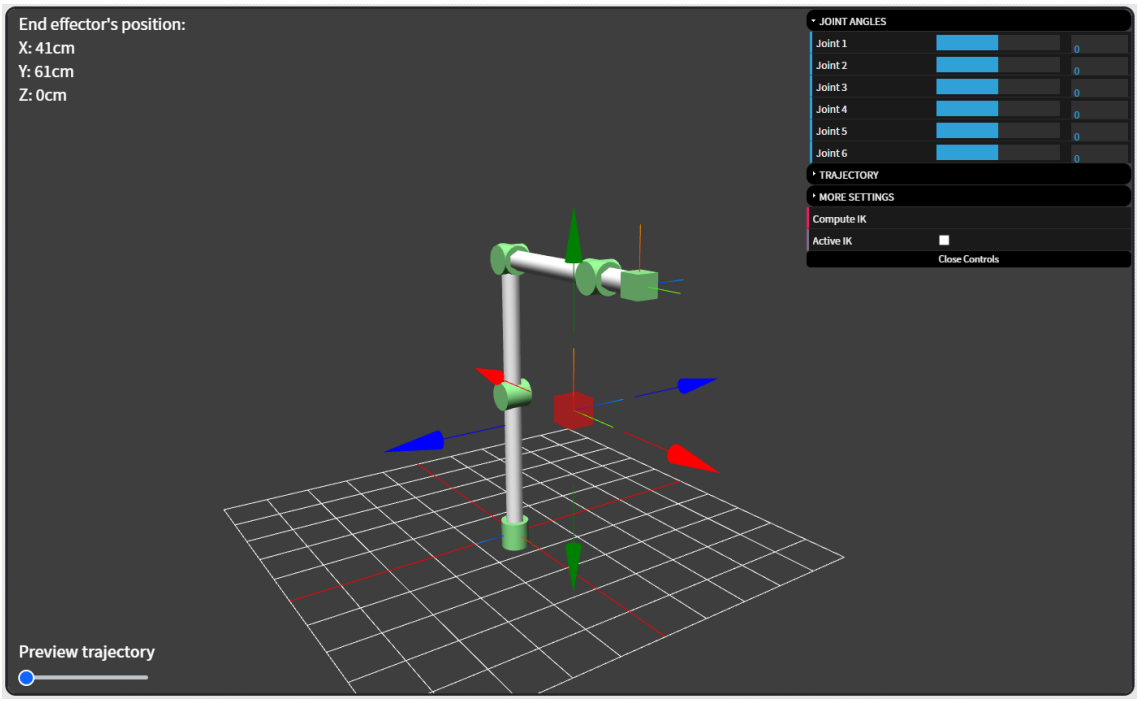


Obrázok 9: Načítanie prvkov do scény

3.2 Inverzná kinematika

V tejto časti je pre užívateľa k dispozícií 3D scéna do ktorej môže zadať požadovaný bod do ktorého sa má end-efektor jeho reťazca dostať. Pre niektoré robotické ruky (6DOF so sférickým zápästím) je možné nastaviť aj orientáciu end-efektora.

Výstupom po zadaní požadovanej pozície end-efektora je lineárna trajektória prejdená end-efektorom zo začiatočného bodu do požadovaného. Užívateľovi sú poskytnuté dáta o jednotlivých kombináciách uhlov v stupňoch, ktoré treba zadať kĺbom, aby end-efektor prešiel po tejto trajektórii. Získané dáta môže využiť vo svojej vlastnej aplikácií.



Červená kocka znázorňuje požadovanú pozíciu a orientáciu end-efektora.

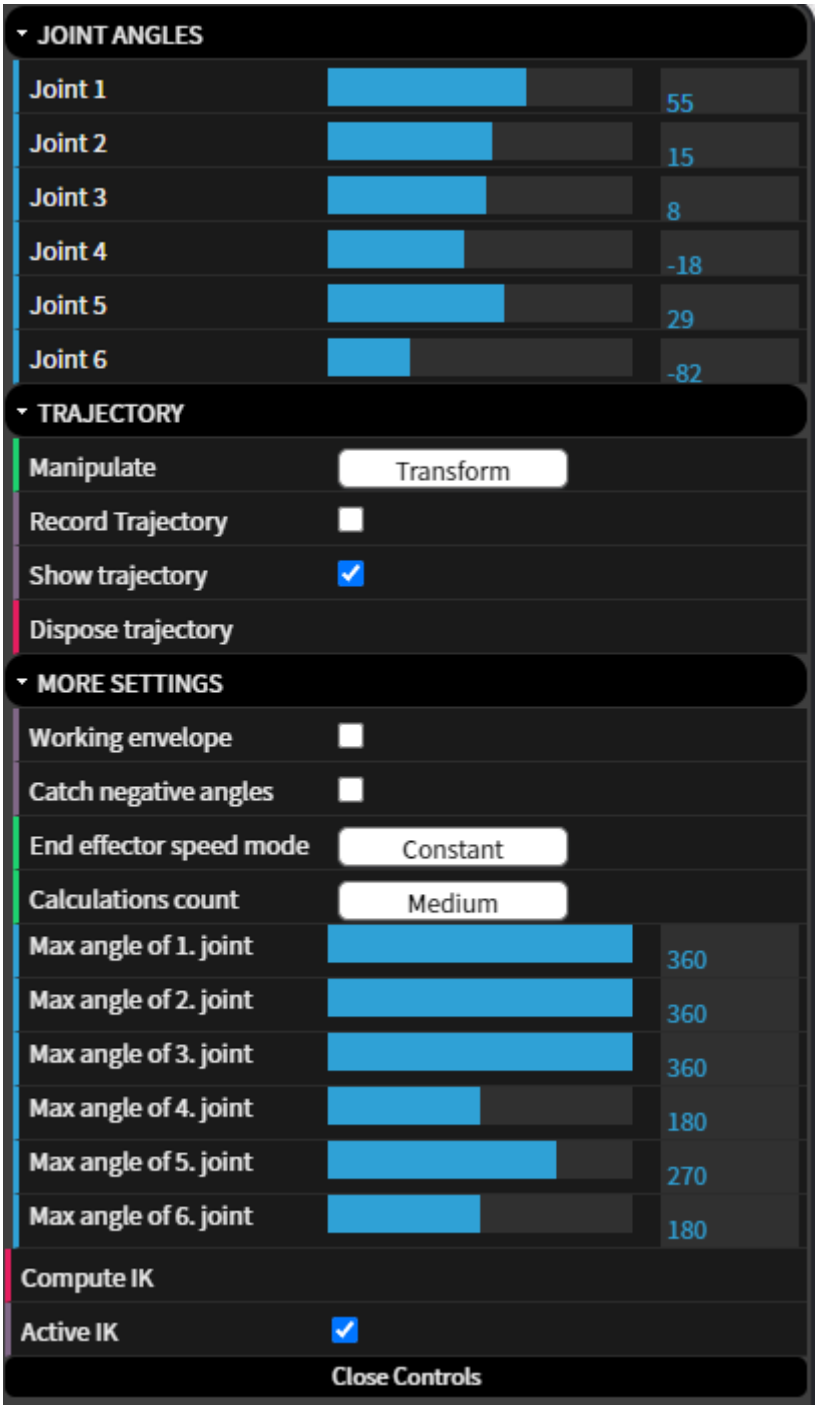
3.2.1 Ovládací panel

Ovládací panel je základnou časťou plánovania trajektórie. Užívateľ ma na výber zmeniť tieto nastavenia:

- zadanie požadovanej pozície (New position),
- nastavenie začiatočných uhlov pre každý z kĺbov,
- maximálny uhol, o ktorý vie konkrétny kĺb rotovať,

- rýchlostný mód end-efektora,
- presnosť výpočtov,
- výnimka záporného uhla,
- forma výsledných uhlov,
- zaokrúhlenie výsledných uhlov.

Zadanie požadovanej pozície je triviálne. Zadáva sa v centimetroch, platným vstupom sú aj desatinné čísla. Do tejto pozície sa bude počítať inverzná kinematika.



Reťazce sa vytvárajú s nulovými začiatočnými uhlami pre každý kĺb. To síce uľahčuje reprezentovanie dát ale nie je to ideálna začiatočná konfigurácia. Preto si užívateľ môže nastaviť jeho vlastnú konfiguráciu uhlov.

Robotická ruka môže mať limitácie v maximálnom možnom uhle okolo ktorého vie kĺb rotovať. Pre väčšinu Servo motorov je to 180°, ale limitácie môžu byť rôzne. Prednastavená hodnota je 360°.

Ovládanie presnosti výpočtov spravidla buď zníži alebo zvýši hustotu bodov, na ktorých je IK počítaná. Je dôležité hneď z niekoľkých dôvodov. Mikrokontrolér, ktorý môže riadiť reťazec užívateľa nemusí stíhať zadávať desiatky uhlov pre posunutie o pár centimetrov, preto treba počet kalkulácií znížiť. Alebo práve naopak, potrebuje užívateľ veľmi presnú trajektóriu, preto sa rozhodne zvýšiť presnosť na maximum. Na výber má užívateľ z troch možností:

- Nízka – priemerne 8 kalkulácií na centimeter trajektórie,
- Stredná – priemerne 14 kalkulácií na centimeter trajektórie,
- Vysoká – priemerne 20 kalkulácií na centimeter trajektórie.

Možnosť výnimky záporného uhla jednoducho buď záporné uhly ignoruje alebo im predchádza.[1]

Ďalej má užívateľ možnosť nastaviť si formu v akej výsledné uhly obdrží. Na výber má buď list pre každý kĺb alebo list listov. List pre každý kĺb je jasný, pre tri kĺby, tri listy, každý rovnakej dĺžky so všetkými uhlami idúcimi po sebe. List listov má takú dĺžku, aký je celkový počet výpočtov v trajektórií. V každom liste sú potom konfigurácie všetkých uhlov.

V poslednom rade zaokrúhľovanie výsledkov sa vysvetľuje samé.

Celý ovládací panel je prispôsobený počtu stupňov voľnosti reťazca. To znamená, že sa načíta taký počet posúvačov koľko má užívateľ kĺbov v reťazci

[1] Bola možnosť záporným uhlom vždy predísť, ale je pomerne jednoduché z uhla -1°, spraviť uhol 359°, ktorý už nie je záporný, preto je táto možnosť voľná.

3.2.2 IK algoritmus

Najskôr z DH parametrov podľa vzoru v **Rovnica 1** vytvoríme homogénne transformačné matice. Naprieč výpočtami sa vyskytujú metódy knižnice `numpy`, ďalej `np`, ktorá je pri počítaní matíc nevyhnutnosťou. Veľmi uľahčuje akékoľvek operácie s nimi. Jednotlivé metódy si podrobnejšie vysvetlíme neskôr.

```
for i in range (0, len(UHLY)):

      SIN = math.sin((UHLY[i] * 0.0174532925) + THETAS[i])

      COS = math.cos((UHLY[i] * 0.0174532925) + THETAS[i])

      a = np.array([COS, -SIN * math.cos(ALFAS[i]), SIN *
      math.sin(ALFAS[i]), COS * Rs[i]])

      b = np.array([SIN, COS * math.cos(ALFAS[i]), -COS *
      math.sin(ALFAS[i]), SIN * Rs[i]])

      c = np.array([0, math.sin(ALFAS[i]), math.cos(ALFAS[i]), Ds[i]])

      d = np.array([0, 0, 0, 1])

      HOMOGENEOUSES.append(np.matrix([a, b, c, d]))
```

Ukážka kódu 2: Triedenie do transformačných matíc

DH parametre z reťazca máme zoradené do polí:

- ALFAS – všetky hodnoty α zoradené vzostupne v radiánoch
- THETAS – všetky hodnoty θ zoradené vzostupne v radiánoch
- Ds – všetky hodnoty d zoradené vzostupne v centimetroch
- Rs – všetky hodnoty r zoradené vzostupne v centimetroch

V cykle prechádzame všetkými uhlami. Premenné `SIN` a `COS` majú hodnotu sínusu a kosínusu i-teho uhla v reťazci, ku ktorému je pripočítaná hodnota θ z tabuľky. Do riadkov výslednej matice vkladáme príslušné hodnoty podľa vzoru. Všetky transformačné matice v tvare H_1^0, H_2^1, H_3^2... H_n^{n-1} sa nachádzajú v poli `HOMOGENEOUSES`. Metóda `array` v tomto prípade iba vytvorí polia ktoré tvoria riadky v matici vytvorenej metódou `matrix`.

V ďalšom kroku prechádzame každým prvkom tohto poľa aby sme vytvorili pole so všetkými maticami v tvare $H_1^0, H_2^0, H_3^0 \ldots H_n^0$

Začíname s jednotkovou maticou, ktorá nebude mať vplyv na prvú iteráciu cyklu.

```
    H0x = np.array([[1,0,0,0],
                    [0,1,0,0],
                    [0,0,1,0],
                    [0,0,0,1]])
    for i in HOMOGENEOUSES:
        H0x = np.matmul(H0x, i)
        hom_from_0.append(H0x)
        xpos = float(H0x[0:1, 3:4])
        ypos  = float(H0x[1:2, 3:4])
        zpos = float(H0x[2:3, 3:4])
        dis = np.matrix([[xpos],[ypos],[zpos]])
        dis_from_0.append(dis)
```

Ukážka kódu 4: Triedenie do finálnych matíc

Postupne matice násobíme a ukladáme ich do poľa `hom_from_0`. Premenné `xpos`, `ypos`, `zpos` sú časťami 3×1 podmatice opisujúce transláciu T z Tabuľka DH parametrov. Translačné časti matíc ukladáme do poľa `dis_from_0`, ktoré potrebujeme na doplnenie Jakobiho matice. Z homogénnej matice ktorá má rozmery 4×4 tieto hodnoty vyberáme takto: `H0x[0:1, 3:4]` 0:1 znamená, že vyberáme prvok nachádzajúci sa v prvom riadku a 3:4 špecifikuje výber na štvrtý stĺpec.

Metóda `matmul` vynásobí dve polia buď v tvare `array` alebo `matrix`, ktoré pošleme ako parametre.

```
xBodka =  (target_pos_x - akt_pos_x) / precision
yBodka =  (target_pos_y - akt_pos_y) / precision
zBodka =  (target_pos_z - akt_pos_z) / precision
```

Ukážka kódu 6: Zložky smerového vektora výpočtov

Pre začiatok počítania Jakobiho matice, musíme určiť vektor smeru našich výpočtov. Jeho dĺžka bude závisieť od zadanej náročnosti výpočtov, a smer určia dva body: aktuálna

pozícia end-efektora (`akt_pos`) a požadovaná pozícia (`target_pos`). Ako každý 3D vektor má aj náš vektor smeru 3 zložky: xBodka, yBodka, zBodka, každá zložka je vydelená presnosťou (`precision`). Pri neskoršom zvyšovaní uhlov budeme násobiť zložkami tohto vektora, preto dáva zmysel, že čím vyššia presnosť, tým nižšie zvýšenie uhla.

```
jed_matica = np.matrix([[0], [0], [1]])
prvy_element = np.cross(jed_matica, dis_from_0[-1], axis=0)
JACOBIAN = np.matrix([[prvy_element.item(0)],
                      [prvy_element.item(1)],
                      [prvy_element.item(2)]])
```

Ukážka kódu 8: Prvý člen Jakobiho matice

Spomeňme si, že členy Jakobiho matice priraďujeme podľa **Rovnica 3**. Keďže sme na prvom člene, za i dosádzame 1. Rotácia z rámca 0 na rámec 0 nie je žiadna, preto prvou časťou vektorového súčinu bude tretí stĺpec 3x3 jednotkovej matice (rotačné matice majú vždy rozmer 3x3). Translácia d_0^0 z rámca 0 na rámec 0 je nulová, preto na rozdiel nemá vplyv. Posledná matica v poli `dis_from_0` je celková translácia z rámca 0 na rámec n. Prvý člen Jakobiho matice je teda priradený mimo hlavný cyklus.

```
for j,i in enumerate(hom_from_0[:-1]):
    rotx = float(i[0:1, 2:3])
    roty = float(i[1:2, 2:3])
    rotz = float(i[2:3, 2:3])
    rot = np.matrix([[rotx], [roty], [rotz]])
    displacement = np.subtract(dis_from_0[-1], dis_from_0[j])
    col = np.cross(rot, displacement, axis=0)
    JACOBIAN = np.append(JACOBIAN, col, axis=1)
```

Ukážka kódu 10: Tvorba Jakobiho matice

V hlavnom cykle prebehneme každou maticou v poli `hom_from_0` okrem poslednej. Keďže každá rotačná časť matice je násobená tretím stĺpcom jednotkovej $\begin{bmatrix}0\\0\\1\end{bmatrix}$ stačí nám zobrať tretí stĺpec z nej. `i[0:1, 2:3]` vyberie z matice člen v prvom riadku a treťom stĺpci. Podobne vezmeme druhý a tretí člen ktoré nám dajú rotačnú časť. Translačné časti sú jasné, vždy odčítavame d_j^0 od d_n^0 kde j reprezentuje počet prejdených cyklov začínajúc od nuly a n je počet kĺbov v reťazci. Opäť vektorovým súčinom dostaneme člen Jakobiho matice.

```
Jt = JACOBIAN.transpose()
JJt = np.matmul(JACOBIAN, Jt)
JJtinv = np.linalg.inv(JJt)
```

Ukážka kódu 12: Psuedoinverzná Jakobiho matica

Vďaka knižnici `numpy` je veľmi jednoduché vypočítať výraz $\mathrm{J}^{\mathrm{T}}(\mathrm{J}\,\mathrm{J}^{\mathrm{T}})^{-1}$. Ako názvy metód naznačujú, `transpose` vypočíta transpóznu maticu a `linalg.inv` vypočíta k zadanej matici inverznú.

Túto maticu spolu so smerovým vektorom musíme ešte nejako využiť pre zvyšovanie uhlov tak, aby sme sa pozíciou end-efektora priblížili k požadovanej pozícií.
Prechádzame každým uhlom a zvyšujeme ho podľa hodnôt v pseudoinverznej Jakobiho matici a násobkom zložky smerového vektora. Tu aj kontrolujeme či uhol neprekračuje zadaný limit užívateľom ktorý je uložený v poli `MAX_ANGLES`, alebo či nie je záporný. Ak začíname s nulovou rotáciou pre každý kĺb, veľmi ľahko môžeme prejsť do záporných hodnôt. Nenašiel som žiaden spôsob akým sa tomu vyhnúť, zostáva len na to užívateľa

upozorniť, aby zvolil inú začiatočnú kombináciu uhlov pre jeho požadovanú trajektóriu. Možnosťou stále zostáva záporné uhly ignorovať.

```
for i in range(0, len(UHLY)):
      ThetaBodka = JtJJtinv.item(3*i) * xBodka +
      JtJJtinv.item((i*3)+1) * yBodka + JtJJtinv.item((i*3)+2) *
      zBodka
      ZvysTheta = ThetaBodka * 10
      Uhol = (UHLY[i]) + ZvysTheta
      if Uhol > MAX_ANGLES[i] or Uhol < 0:
          #uhol je pre reťazec užívateľa neplatný
```

Ukážka kódu 14: Inkrementácia uhlov

Podmienka, ktorej splnenie ukončí cyklus je vzdialenosť end-efektora k cieľovému bodu. Premenná rozdiel vypočíta dĺžku smerového vektora, teda vzdialenosť od cieľovej pozície. Trajektória sa považuje za úspešne splnenú ak je dĺžka tohto vektora v rozmedzí od 10 milimetrov do 25 milimetrov.
Užívateľ má na výber medzi troma typmi trajektórie:

- Konštantná,
- Zrýchľujúca,
- Spomaľujúca.

Algoritmus ktorý sme si práve opísali platí len pre konštantný typ rýchlosti end-efektora., pretože smerový vektor vypočítame len raz a to pred hlavným cyklom. Vždy násobíme tou istou hodnotou, preto bude zmena uhlov konštantná. Pre spomalenie nám stačí spraviť jeden jednoduchý trik, výpočet smerového vektora z ukážky kódu č.7 premiestnime do hlavného cyklu. Preto vždy, keď sa zmenší vzdialenosť end-efektora a cieľovej pozície, zmenší sa aj dĺžka smerového vektora, tým pádom aj veľkosť zmeny uhlov.

Pre zrýchlený pohyb musíme byť trochu viac kreatívni. Namiesto priameho výpočtu aktuálneho problému si ho musíme prevrátiť. Budeme sa tváriť, že naša aktuálna pozícia je cieľová a naopak. Vyriešime problém rovnako ako pri spomaľovaní a výsledné zložky smerových vektorov uložíme do poľa. Nemusíme ich už počítať, len sa vrátime do pôvodnej pozície a násobíme zápornými hodnotami zložiek vektora v opačnom poradí poľa. Takýmto spôsobom vieme docieliť zrýchlenú trajektóriu. Zvyčajne skončí vo väčšej

vzdialenosti od cieľovej pozície v porovnaní s ostatnými dvoma metódami, pretože na konci výpočtov pre spomaľovanie sú výpočty najpresnejšie, kvôli malej vzdialenosti cieľového bodu a menej presné na začiatku. Preto má koniec trajektórie pre zrýchlenie menšiu presnosť.

3.3 Účet

V sekcií stránky Účet má užívateľ prístup ku všetkým reťazcom, ktoré vytvoril v dizajnéri a uložil ich kliknutím tlačidla „Uložiť reťazec v účte“. Každý reťazec má v účte tieto prvky:

- ID, slúžiace na pretvorenie reťazca kedykoľvek a kdekoľvek,
- Interaktívnu 3D scénu pre dôkladné prezretie,
- Tlačidlo „Upraviť“, ktoré otvorí dizajnér priamo v účte a užívateľ si tak môže zmeniť pozíciu kĺbov a rotáciu referenčných rámcov,
- Tlačidlo „IK reťazca“, ktoré zavedie užívateľa do sekcie stránky plánovač trajektórie, kde sa jeho zvolený reťazec zobrazí v 3D grafe a je pripravený na zadanie cieľa a vlastností ovládacím panelom,
- Tlačidlo „×“ slúžiace na vymazanie reťazca z účtu.

3.3.1 SQLAlchemy

Na vytváranie nových užívateľov, kontrolu hesiel a ukladanie reťazcov do profilu bolo použité rozšírenie pre Flask, ktoré pridáva podporu SQLAlchemy. SQLAlchemy mapuje funkcie SQL do objektov a poskytuje efektívny a vysokovýkonný prístup k databáze. [11]

```
new_user = User(email=email, first_name=first_name,
password=generate_password_hash(password1, method='sha256'))
db.session.add(new_user)
db.session.commit()
login_user(new_user, remember=True)
```

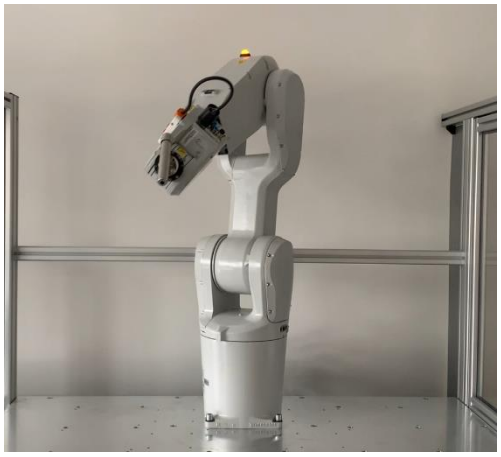
Ukážka kódu 16: Vytvorenie nového profilu

3.4 Pomoc

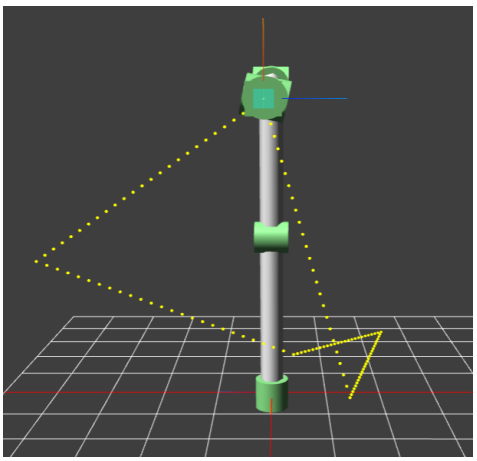
Náročnosť reprezentácie dát si vyžaduje istú mieru pozornosti od užívateľa. Ak chce využiť stránku čo najviac, musí sa napríklad naučiť ako udeľovať referenčné rámce kĺbom, čo znamenajú DH parametre aby si vedel overiť, či zadal všetko správne. Užívateľ nie je v hľadaní odpovedí na mnoho otázok sám, môže sa obrátiť na sekciu stránky Pomoc, aby našiel odpovede na otázky, s ktorými sa spája najviac problémov. Užívateľ sa tu stretne s najčastejšie pýtanými otázkami k tejto téme, bude mu vysvetlené využitie každej časti a nejasnosti spájajúce sa s ňou. Sekcia stránky Dizajnér je obzvlášť problematická, pretože sa užívateľ musí zoznámiť s ovládaním scény, pridávaním a rotovaním objektov. Určite to pomôže celkovému dojmu zo stránky a užívateľov neodradí systém akým je stránka vytvorená.

3.5 Overenie presnosti výpočtov

Pre overenie presnosti výpočtov bol vykonaný pokus na priemyselnej robotickej ruke IRB 1200 so šiestimi stupňami voľnosti od spoločnosti ABB. Na webstránke bol pretvorený kinematický diagram dostupný v dokumentácií pre robotickú ruku a následne s ním bola naplánovaná trajektória. Uhly boli zadané pomocou programovacieho jazyka RAPID, ktorý je používaný v softvéri pre ABB robotické ruky, Robot Studio. Cieľom bolo zrealizovať lineárnu trajektóriu a konštantnú orientáciu nástroja. Porovnanie plánovača trajektórie webstránky s výsledkami v reálnom svete je dostupné online na adrese https://www.youtube.com/watch?v=MN2ECP03tcM&ab_channel=Adri%C3%A1nHochla



Obrázok 8*:* Robotická ruka IRB 1200



Obrázok 9: Naplánovaná trajektória

Výsledok pokusu je jasný. Robotická ruka úspešne vykonala zadanú trajektróriu s vysokoui presnosťou. Dôkazom je konštantná orientácia nástroja pripevneného na konci robotickej ruky. Z toho vyplýva, že riešenie poskytnuté webstránkou je na priemyselnej úrovni a výpočty sú veľmi presné.

4 Výsledky a diskusia

Najdôležitejším faktom je, že bolo úspešne implementované riešenie inverznej kinematiky pre kohokoľvek vo forme webovej stránky. Zistilo sa, že výsledky výpočtov sú aplikovateľné pre všeobecný kinematický diagram čo veľmi rozširuje dosah a využitie aplikácie. Takáto pomôcka, vďaka svojej flexibilite nájde svoje využitie pre kohokoľvek kto sa zaoberá danou problematikou ale aj pre tých, ktorí chcú čo najrýchlejšie a najpohodlnejšie vyriešiť ich IK problém. Obrovskou výhodou realizácie tohto projektu na webe je fakt, že užívateľ si nemusí sťahovať žiadne dodatočné súbory, ani sa učiť špecifické príkazy pre ovládanie svojho reťazca, ako je to v iných implementáciách. Jedinečnosť takejto aplikácie jej dáva hodnotu. Kombinácia rozličných technológií umožňuje užívateľovi veľkú škálu možností, pre tvorbu a úpravu reťazcov a plánovanie komplexnej trajektórie.

Pseudoinverzná metóda výpočtov obstála na jednotku, pretože je veľmi spoľahlivá a ukázalo sa, že aj nadpriemerne rýchla. Jej najväčšou výhodou je práve všeobecnosť pre n-stupňov voľnosti, ktorá bola esenciálna pre dosiahnutie vytýčeného cieľu. Jediná nepresnosť v správaní výpočtov nastáva, ak požadovaná pozícia nie je v dosahu kinematického reťazca, vtedy sú výsledky chaotické a nepredvídateľné. Napriek tomu sa v pracovnej obálke reťazca bude bodu približovať.

Dôkazom o úspešnom zrealizovaní projektu nie je len vlastný pokus, kde bol na industriálnej robotickej ruke vyriešený problém inverznej kinematiky, ale už aj množstvo naplánovaných trajektórií na stránke pre špecifické kinematické diagramy.

Pred vzniknutím tohto projektu neexistoval žiaden prostriedok na webe, ktorý by umožňoval takúto funkcionalitu.

Výhodou je, že aplikácia funguje bez problémov aj na mobilných zariadeniach. Náročnosť problému spôsobí, že väčšina užívateľov bude na väčších obrazovkách no WebGL, nevyhnutný pre 3D scény, je kompatibilný aj s mobilnými zariadeniami. Preto nie je vylúčené, že užívateľ bude so stránkou pracovať aj na mobilom zariadení.

5 Závery práce

Vytýčený hlavný cieľ, spolu s jeho čiastkovými cieľmi som splnil, čoho dôkazom je plne funkčná web stránka na adrese www.ikforeveryone.com.

Verím, že táto kontribúcia nie je zanedbateľná a nájde si množstvo uplatnení. S určitosťou pomôže veľkému počtu ľudí s ich špecifickými riešeniami inverznej kinematiky. Aplikácií robotických rúk sú stovky, preto príležitostí pre návštevu tejto web stránky je tiež množstvo. K tomu je navrhnutá tak, aby bola porozumiteľná a intuitívna. Jednoduchosť spolu s dostupnosťou sú najväčšími výhodami oproti iným implementáciám riešenia tohto problému. Stránka je automaticky preložená do všetkých jazykov, aby sa predišlo problémom s porozumením odbornej terminológie.

Je veľmi dôležité stále hľadať lepšie a praktickejšie riešenia zložitých problémov a robiť ich dostupnejšími. Robotika a kinematika sú rýchlo meniacimi sa odvetviami a preto je dôležité riešiť problémy rýchlo a efektívne. Je potrebné sa sústrediť na jednoduchosť použitia pre užívateľa aby ich téma neodstrašila, ale práve naopak, zaujala. Práve web je miesto, kde treba takéto riešenia ukázať.

Samozrejme plánujem stránku ešte vylepšovať. To, čo podľa môjho názoru stránke najviac chýba, je možnosť pridávania prizmatických kĺbov. Aj tie totižto tvoria značnú časť robotických reťazcov. Budem sa snažiť túto funkcionalitu pridať čo najskôr, ale pravdou zostáva, že v terajšej verzií táto možnosť poskytnutá nie je.

6 Zhrnutie

Cieľom tejto práce bolo vytvoriť webovú stránku, zameranú na riešenie inverznej kinematiky pre všeobecný kinematický reťazec. Docielili sme to tvorbou prehľadných 3D scén, ktoré umožňujú pretvoriť špecifický reťazec a následne naplánovať trajektóriu s množstvom nastavení. Užívateľ obdrží všetky kombinácie uhlov, ktoré dostanú end-efektor jeho reťazca po jeho naplánovanej trajektórií. Pre výpočty bola zvolená metóda pseudoinverznej Jakobiho matice. Zistilo sa, že metóda fungovala dokonca v rozličných formách, čo umožnilo ovládať aj typ trajektórie samotnej. Najväčšou výhodou tejto aplikácie je fakt, že, sa k nej môže dostať naozaj každý, odkiaľ pochádza aj názov pre túto prácu “Inverzná kinematika pre každého”

7 Zoznam použitej literatúry

1. **Buss, Samuel R.** [Online] 7. Október 2009. http://www.cs.cmu.edu/~15464-s13/lectures/lecture6/iksurvey.pdf.

2. **P, Desai Jaydev.** D-H Convention. [Online] 2004. https://www.semanticscholar.org/paper/D-H-Convention-Desai/9602f5223df58687b7edcd591a84fdcb4141e9bc.

3. **FORWARD KINEMATICS:THE DENAVIT-HARTENBERG CONVENTION. [Online] https://users.cs.duke.edu/~brd/Teaching/Bio/asmb/current/Papers/chap3-forward-kinematics.pdf.**

4. Sodemann, Angela. youtube.com. [Online] 10. Október 2015. https://www.youtube.com/watch?v=VL1wJ-ui9BI&ab_channel=AngelaSodemann.

5. wikipedia.com. [Online] 9. Jún 2019. https://cs.wikipedia.org/wiki/Jacobiho_matice.

6. Cristina, Stefania. Machine Learning Mastery. [Online] 2. August 2021. https://machinelearningmastery.com/a-gentle-introduction-to-the-jacobian/.

7. Wei Song, Guang Hu. A Fast Inverse Kinematics Algorithm for Joint Animation. [Online] 2011. https://pdf.sciencedirectassets.com/278653/1-s2.0-S1877705811X00206/1-s2.0-S187770581105507X/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEC8aCXVzLWVhc3QtMSJIMEYCIQC3cbfdSt0Rsckq8igfxXC9g4ZakJJgHjrpCtatahNvNwIhAJbfd4y%2BVrxRaGm6zRKaW97T%2FSa1f6nqBIySz7di.

8. Flask documentation. [Online] https://flask.palletsprojects.com/en/2.0.x/.

9. wikipedia.com. *three.js.* [Online] 28. Január 2022. https://en.wikipedia.org/wiki/Three.js.

10. dash.plotly.com. *Basic-callbacks.* [Online] https://dash.plotly.com/basic-callbacks.

11. sqlalchemy.org. *SQLAlchemy.* [Online] 21. Január 2022. https://www.sqlalchemy.org/.

12. [Online] https://www.researchgate.net/figure/BioRob-4-DOF-robot-arm-kinematic-structure-and-table-with-DH-parameters_fig4_220850180.

13. [Online] https://homes.cs.washington.edu/~todorov/courses/cseP590/06_JacobianMethods.pdf.

Prílohy

Zdrojový kód ku aplikácií – *zdrojovy_kod.zip*