Řešení inverzní kinematiky redundantního 3D manipulátoru neuronovou sítí

June 27, 2024

1 Určení koncové polohy 3D manipulátoru pomocí rovnic a natočení jednotlivých kloubů

3D manipulátor (robotická ruka) je typ mechanické ruky ovládané pomocí počítače, která má podobné funkce jako lidská ruka. Redundantní znamená, že má více stupňů volnosti než je potřeba pro plnou kontrolu pohybu. Toho je dosaženo použitím více kloubů než je nutné k dosažení všech možných poloh v prostoru. Tyto stupně volnosti jsou užitečné k optimalizaci pohybu.

1.1 Matice rotace

Pro popsání pohybu tělesa v euklidovském prostoru se používají matice rotace. Rotace je pohyb tělesa, které změní natočení objektu kolem určité osy. V 3D prostoru jsou možné rotace okolo 3 hlavních os souřadného systému. Matice rotace matematicky popisují právě pohyb bodu, který se natočí o úhel θ okolo jedné nebo více os souřadného systému s použitím pravidla pravé ruky.

Rotace okolo osy x:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}$$

Rotace okolo osy y:

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

Rotace okolo osy z:

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Výsledná poloha je dána maticovým součinem jednotlivých rotací o úhly α, β a γ okolo os x, y a z. Výsledná matice se tedy spočítá:

$$R(\theta) = R_x(\alpha)R_y(\beta)R_z(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix} \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} cos(\beta)cos(\gamma) & sin(\alpha)sin(\beta)cos(\gamma) - cos(\alpha)sin(\gamma) & cos(\alpha)sin(\beta)cos(\gamma) + sin(\alpha)sin(\gamma) \\ cos(\beta)sin(\gamma) & sin(\alpha)sin(\beta)cos(\gamma) + cos(\alpha)sin(\gamma) & cos(\alpha)sin(\beta)sin(\gamma) - sin(\alpha)cos(\gamma) \\ -sin(\beta) & sin(\alpha)cos(\beta) & cos(\alpha)cos(\beta) \end{bmatrix}$$

1.2 Pohyb tuhého tělesa

Pohyb tuhého tělesa se skládá z kombinace rotace(otočení) a translace(posuv). Následující homogenní matice ukazují posuv o délku a (pohyb v ose x), b (pohyb v ose y) a c(pohyb v ose z).

$$T_x(a) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_x(a) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_x(a) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Následující rovnice udává koncovou polohu bodu, kde vektor **p** je počáteční poloha bodu. $H = T_x T_y T_z R_x R_y R_z p$

1.2.1 Denavit-Hartenbergova(DH) metoda

V robotice se pro určení koncové polohy robotické ruky, ve které se vyskytuje pohyb více tuhých těles, nejčastěji používá Denavit-Hartenbergova metoda, která výpočty značně zjednoduší. Byla vyvinuta v 50. letech 20. století Johnem Denavitem a Stuartem Hartenbergem. Pomocí 4 parametrů popisuje prostorové uspořádání jednotlivých kloubů a článků robotického manipulátoru. Tyto parametry se nazývají Denavit-Hartenbergovy parametry (délka,úhel,posuv,rotace).

1) Délka je vzdálenost podél osy X(popřípadě Y) od počátečního článku k dalšímu.
2) Úhel se měří mezi osami X počátečního článku k dalšímu.
3) Rotace je proměnná kloubu, otáčí se vždy podél osy Z a přičte se úhel mezi osami Z počátečního článku k dalšímu.
4) Posuv je vzdálenost měřená podél osy Z od původního článku k dalšímu.

Pro použití této metody je zásadní správné zavedení souřadného systému v každém kloubu. K tomu existují 4 pravidla:

1) Osa Z musí být osa rotace.
2) Osa X musí být kolmá na osu Z v kloubu ve kterém začínáme i končíme.
3) Osa Y je dána pravidlem pravé ruky.
4) Osa X musí protínat osu Z předešlého kloubu.

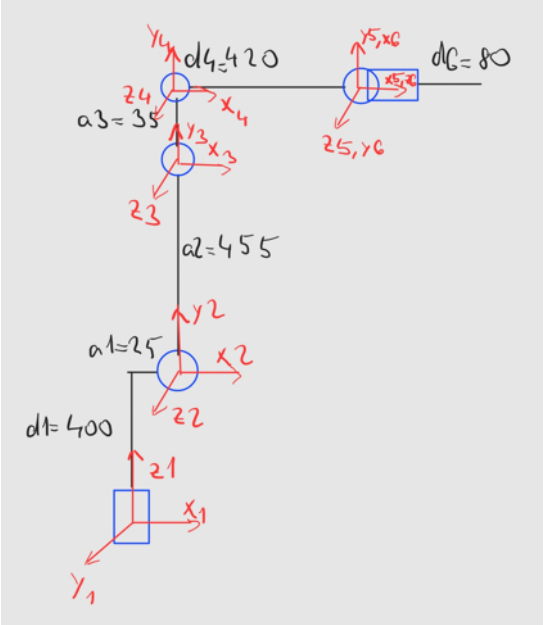
Po náčrtu robotické ruky a správném zavedení souřadných systémů do každého kloubu se vytvoří DH tabulka(č. tabulky). Proměnná a_i je délka, α_i je úhel, d_i je posuv a θ_i je otočení.

	a_i	α_i	d_i	θ_i
1	a_1	α_1	d_1	θ_1
...	...	...	...	...
n	a_n	α_n	d_n	θ_n

Tyto údaje se dále postupně dosadí do matice.

$$H_i = R_{z,\theta}T_{x,a}T_{z,d}R_{x,\alpha} = \begin{bmatrix} cos\theta & -sin\theta cos\alpha & sin\theta sin\alpha & acos\theta \\ sin\theta & cos\theta cos\alpha & -cos\theta sin\alpha & asin\theta \\ 0 & sin\alpha & cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Výsledná hodnota koncové polohy je dána maticovým součinem všech matic H_i



	a_i	d_i	α_i	θ_i
1	25	400	0	θ_1
2	455	0	0	$\theta_2 - 90$
3	35	0	0	$\theta_3 - 90$
4	0	420	0	θ_4
5	0	0	0	θ_5
6	0	80	-90	$\theta_6 - 90$

Inverzní kinematika

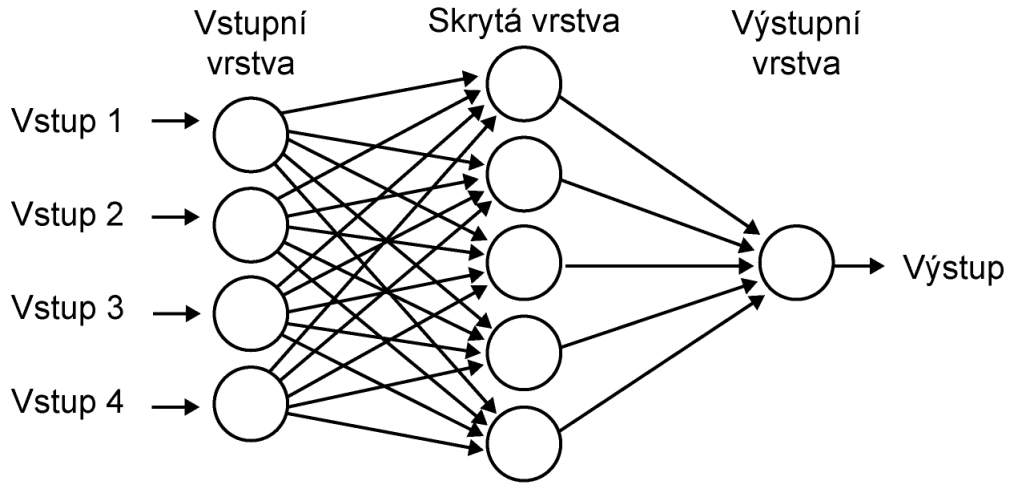
2 Inverzní kinematika

Inverzní kinematika je problém, při kterém se snažíme najít jednotlivé kloubové úhly, kterýma se dostaneme do požadované polohy. Existuje spousta metod, jakýma lze tento problém vyřešit. Ty pak lze rozdělit podle toho, jestli řeší jen potřebné natočení všech kloubů nebo řeší jejich pohyb.

2.1 Řešení pomocí neuronové sítě

Řešení inverzní kinematiky neuronovou sítí má několik výhod oproti analytickému řešení. Neuronové sítě jsou schopné řešit inverzní kinematiku robotů s více než šesti stupni volnosti, což je analyticky obtížné nebo dokonce nemožné. Další výhodou je, že natrénovaná síť je schopná velmi rychle předpovědět výsledky, což může být u analytického řešení velmi výpočetně náročné (zejména pokud se jedná o nelineární rovnice). Také jsou lépe schopny se přizpůsobovat změnám v systému bez předefinování rovnic potřebných k nalezení řešení. Dalším kladem je, že neuronové sítě dokážou do určité míry kompenzovat šum a chyby v měření, na což může být analytické řešení citlivé.

Problém inverzní kinematiky lze vyřešit pomocí různých druhů neuronových sítí. U neredundantních ruk to lze vyřešit hlubokou neuronovou sítí s jednou skrytou vrstvou. Byl experimentálně vyzkoušen různý počet neuronů v této skryté vrstvě s hodnotami [20, 33, 50, 70, 100, 150, 250]. Jako optimalizér byl použit algoritmus Adam a za ztrátovou funkci byla použita funkce MSE. Neuronová síť tedy bude mít schéma podobné jako v následujícím obrázku.



2.1.1 Adam

Adam (Adaptive Moment Estimation) je jedním z nejpoužívanějších algoritmů pro optimalizaci neuronových sítí. Byl navržen Diederikem Kingmou a Jimmym Ba v roce 2014. Jedná se o kombinaci 2 rozšíření gradientního spádu a to kontrétně AdaGrad(Adaptive Gradient Algorithm) a RMSProp(Root Mean Square Propagation). Síť byla trénována po dobu 1000 epoch.

2.1.2 Vlastnosti

- **Adaptivní Učení:** Adam upravuje učící se rychlost pro každý parametr na základě prvního momentu (průměru) a druhého momentu (nekvantovaného rozptylu) gradientů. To zajišťuje rychlou konvergenci a zároveň stabilitu.
- **Inkrementální aktualizace:** Místo globálního nastavení konstantní učící se rychlosti, Adam adaptivně mění rychlost učení během tréninku, což umožňuje efektivnější učení.
- **Méně parametrů k ladění:** Použití Adam optimalizátoru vyžaduje méně ladění hyperparametrů ve srovnání s tradičními gradientními metodami.

2.1.3 Matematický Popis

Adam využívá dvě veličiny:

- m_t (první moment, odhad střední hodnoty gradientu)
- v_t (druhý moment, odhad nekvantovaného rozptylu gradientu)

Aktualizace váhy θ_t se provádí následovně:

1. **Inicializace:**

$$\begin{aligned} m_0 &= 0 \\ v_0 &= 0 \\ t &= 0 \end{aligned}$$

2. **Pro každý krok:**

$$t = t + 1$$

Výpočet gradientu g_t v čase t.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \quad \text{(Bias korekce pro } m_t\text{)} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \quad \text{(Bias korekce pro } v_t\text{)} \\ \theta_t &= \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \end{aligned}$$

Kde:

- α je učící se rychlost.
- β_1 a β_2 jsou hyperparametry pro průměrování momentů.
- ϵ je malá hodnota pro numerickou stabilitu.

[Chat GPT, nutná kontrola]

2.1.4 Mse

MSE(Mean Squared Error) je loss funkce definována jako průměrná hodnota druhé mocniny chyb, kde chyba je rozdíl mezi predikovanou hodnotou a skutečnou hodnotou.

Matematicky je MSE definována jako:

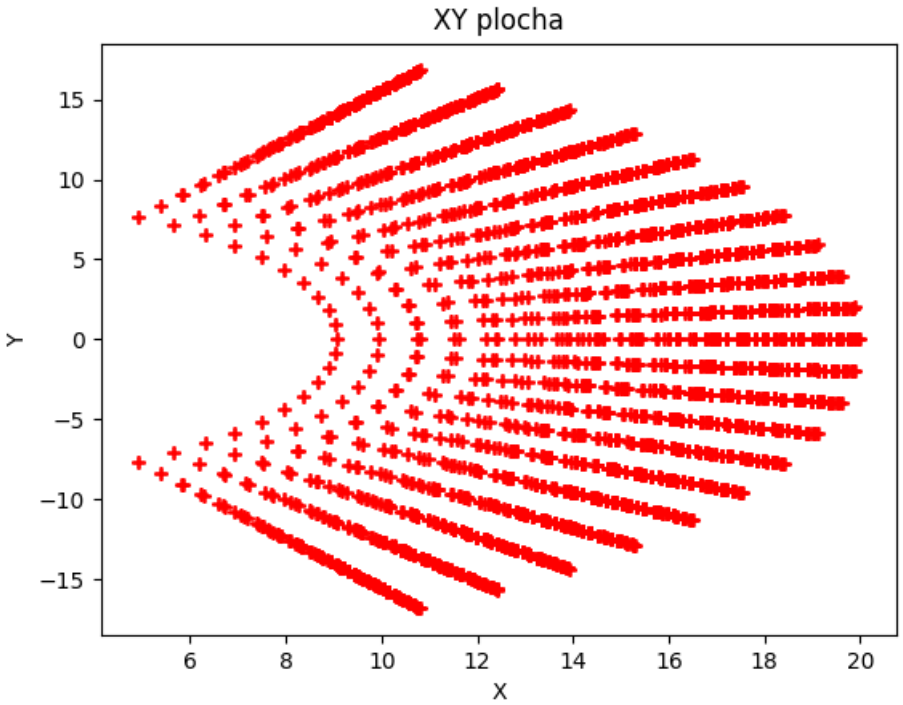
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Kde:

- n je počet pozorování.
- y_i jsou skutečné hodnoty.
- $\hat{y}_i$ jsou predikované hodnoty modelu.

2.1.5 Trénink

Nejprve je potřeba vytvořit si data, která se budou trénovat. Při trénování dat bude základem Denavit Hartenbergova notace(DH). Byla vytvořena tři pole obsahující různé úhly, jejichž hodnoty byly dosazeny do DH, čímž jsme získali různé pozice koncového členu. Ty byly zobrazeny v grafech pro jednotlivé plochy(XY,XZ,YZ).



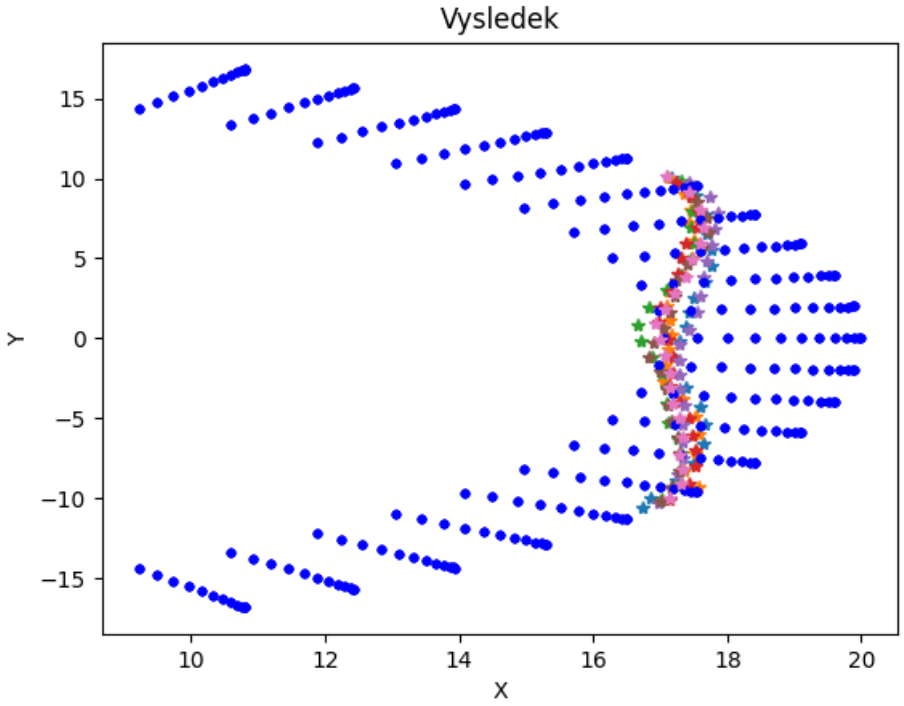
Vzhledem k tomu, že neredundantní manipulátory jsou stavěny určitým způsobem - 6 stupňů volnosti, kde první kloub udává otočení okolo osy Z, poslední tři klouby nám udávají konečnou polohu a natočení, tak nás bude zajímat hlavně pohyb dvou kloubů v jedné ploše (značená XY).

Po získání jednotlivých souřadnic(X,Y,Z) ze zadaných úhlů jednotlivých kloubů se trénuje hluboká síť s přehozenými daty- zadají se souřadnice X,Y,Z a UI se snaží najít správné natočení kloubů aby se do těchto souřadnic dostala.

2.1.6 Výsledek

Po natrénování lze zobrazit výsledek zadáním jednotlivých požadovaných poloh a porovnáním s predikcí natočení jednotlivých kloubů pro dané polohy pomocí UI a následovným dosazením těchto úhlů do DH.

Bylo zadáno 25 bodů o souřadnicích (17,y,0), y=-10..+10. V následujícím grafu jsou ukázány polohy dosažené pomocí UI.



Průměrné chyby pro různý počet neuronů ve skryté vrstvě.

```
Průměrné chyby pro jednotlivé konfigurace:
Konfigurace 1 s 20 uzly: 0.529256875419533
Konfigurace 2 s 33 uzly: 0.3217762436710383
Konfigurace 3 s 50 uzly: 0.38402592673499664
Konfigurace 4 s 70 uzly: 0.20959636189090758
Konfigurace 5 s 100 uzly: 0.43223750176084186
Konfigurace 6 s 150 uzly: 0.41205888829968873
Konfigurace 7 s 250 uzly: 0.2638065902822686
```

Z těchto dat lze vyčíst, že neuronová síť by chtěla ještě trochu doladit parametry. Toho by šlo dosáhnout pomocí zvýšení počtu epoch, zvětšení počtu trénovaných dat, přidáním neuronových vrstev nebo počtu neuronů. Také by šlo zkusit jiný optimalizér.