# Vaja 3: Robotski manipulator

Robotika in računalniško zaznavanje

2017/2018

Za vajo ustvarite mapo `vaja5`. Sem si prenesite tudi razpakirano vsebino datoteke `vaja5.zip` s spletne strani predmeta. Rešitve nalog boste pisali v *Matlab/Octave* skripte in jih shranili v mapo `vaja5`. Da uspešno opravite vajo, jo morate predstavili asistentu na vajah. Pri nekaterih nalogah so vprašanja, ki zahtevajo skiciranje, ročno računanje in razmislek. Odgovore na ta vprašanja si zabeležite v pisni obliki in jih prinesite na zagovor. Deli nalog, ki so označeni s simbolom ★ niso obvezni. Brez njih lahko za celotno vajo dobite največ 75 točk (zgornja meja je 100 točk kar pomeni oceno 10). Vsaka dodatna naloga ima zraven napisano tudi število točk. V nekaterih vajah je dodatnih nalog več in vam ni potrebno opraviti vseh.

## Uvod

V tej vaji bomo v praksi spoznali nekaj osnovnih pristopov k upravljanju robotskega manipulatorja. Za upravljanje manipulatorja moramo prvo spoznati njegove lastnosti, ki jih na kompakten način opišemo z Denavit-Hartenbergerjevi parametri. Ti parametrji zajamejo nujne parametre geometrijskega modela robotskega manipulatorja. Kako uporabljamo te parametre si bomo pogledali v prvi nalogi.

Geometrijski model manipulatorja lahko sicer v splošnem določimo kot verigo transformacij. Bolj natančno gre za transformacije, ki nas iz izhodiščnega prostora manipulatorja preslikajo v prostor posameznega sklepa (torej lahko neposredno določimo, kako daleč je določena točka od prijemala, lahko pa določimo tudi njeno relativno lego). Na ta način lahko seveda določimo tudi položaj zadnjega sklepa (prijemala), če s končno transformacijo te verige preslikamo kar izhodiščno točko $(0, 0, 0)$.

Transformacijo lahko definiramo s pomočjo homogenih koordinat, oziroma matrike, ki opisuje novi prostor:

$$\mathbf{T}(q) = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

matriko pa lahko interpretiramo tudi kot štiri vektorje $n$, $s$, $a$, in $p$, ki definirajo preslikani prostor:

(a) $p$ - translacija med prostoroma

(b) $a$ - approach (približevanje objektu) ($z$ os)

(c) $n$ - normala ($x$ os)

(d) $s$ - slide ($y$ os)

V zgoraj opisanem primeru je matrika izražena kot funkcija parametra q. Ta je vektor spremenljivk, ki definirajo stanje sklepov.

# Naloga 1: Denavit-Hartenbergerjevi parametri

Problem določanja položaja robotskega manipulatorja lahko poenostavimo, saj gre pri vsem skupaj le za nizanje omejenega števila bazičnih operacij, ki so odvisne zgolj od tipa sklepov ter njihovih položajev. Denavit-Hartenbergjevi parametri nam omogočajo prav to - poenostavljeno računanje transformacije. Vsak sklep opisujejo štirje parametri, trije statični ter eden, ki se spreminja (kateri je to, je odvisno od tipa sklepa):

(a) $\theta$ - rotacija okrog osi $z$ (parameter v $q$, če je sklep rotacijski)

(b) $d$ - premik v samem sklepu po $z$ (parameter v $q$, če je sklep translacijski)

(c) $a$ - razdalja med sklepoma po $x$ osi

(d) $\alpha$ - rotacija okrog osi $x$ osi

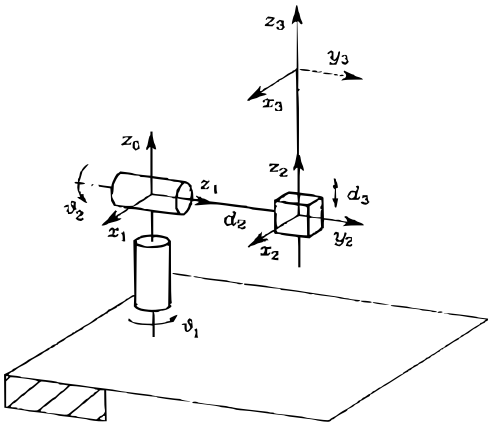
Izračun matrike za $i$-ti sklep izrazimo rekurzivno; z uporabo matrike za $(i-1)$-ti sklep računamo matriko za $i$-ti sklep.

$$\mathbf{T}_i = \mathbf{T}_{i-1} \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2)$$

(a) ***Vprašanje:*** Podana je tabela Denavit-Hartenbergerjevih parametrov za antropomorfni manipulator. Iz tabele izpelji transformacijsko matriko za zadnji sklep.

| sklep | $a$ | $\alpha$ | $d$ | $\theta$ |
|---|---|---|---|---|
| *1* | 0 | $\frac{\pi}{2}$ | 0 | $\theta_1$ |
| *2* | 40 | 0 | 0 | $\theta_2$ |
| *3* | 40 | 0 | 0 | $\theta_3$ |

(b) ***Vprašanje:*** Za Stanfordski model, predstavljen na spodnji sliki določi tabelo Denavit-Hartenbergerjevih parametrov. Določi parametre, ki se lahko med delovanjem manipulatorja spreminjajo.




V nadaljevanju bomo implementirali kinematske enačbe v okolju *Matlab/Octave*.

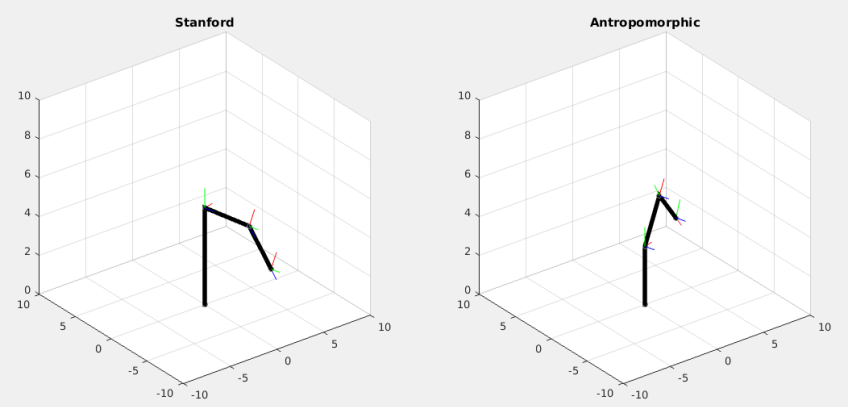
(a) Napišite funkcijo `stanford_manipulator`, ki za dani vektor parametrov vrne matrike transformacij iz izhodišča v vse tri sklepe. Dolžina prvih dveh sklepov naj bo 5 enot, zadnjega pa 2 enoti. Za izhodišče uporabite naslednjo kodo:

```
function [A] = dh_joint(parameters)
% INPUT: DH parameters for joint (a, alpha, d, theta)
% OUTPUT: 4x4 homogeneous transformation matrix

A = zeros(4, 4);

A(1, 1) = cos(parameters(4));
A(2, 1) = sin(parameters(4));

A(1, 2) = -sin(parameters(4)) * cos(parameters(2));
A(2, 2) = cos(parameters(4)) * cos(parameters(2));
A(3, 2) = sin(parameters(2));

A(1, 3) = sin(parameters(4)) * sin(parameters(2));
A(2, 3) = -cos(parameters(4)) * sin(parameters(2));
A(3, 3) = cos(parameters(2));

A(1, 4) = parameters(1) * cos(parameters(4));
A(2, 4) = parameters(1) * sin(parameters(4));
A(3, 4) = parameters(3);

A(4, 4) = 1;
```

Na podoben način določite tudi funkcijo `antropomorphic_manipulator`, ki vrne isti rezultat za antropomorfni manipulator. Dolžina vseh treh sklepov naj bo 3 enote.

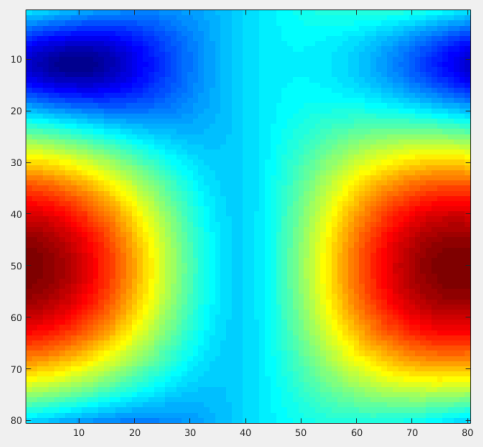
Vizualizirajte položaje posameznih sklepov robotskega manipulatorja za implementirani Stanfordski robotski model. Najprej izračunajte matrike za dano stanje manipulatorja, nato izhodiščno točko $(0, 0, 0)$ pomnožite z matriko vsakega sklepa posebej. Tako dobite položaje posameznih sklepov posebej. V nadaljevanju prikažite koordinatne sisteme za vse tri sklepe s pomočjo funkcije `show_system`, ki ste jo dobili priloženo navodilom (ne pozabite uporabiti ukaza `hold on`, ki fokus risanja zadrži na enem oknu brez vmesnega brisanja vsebine. Za sam izris manipulatorja lahko uporabite funkcije za risanje 3D grafov `plot3` ter `scatter3`, kjer ločeno podate vektorje treh koodrinat sklepov (le-te dobite tako, da izhodiščno točko $(0, 0, 0)$ pomnožite z matriko ustreznega sklepa). Primer uporabe funkcije `plot3`, ki izriše široko črno črto in funkcije `scatter3`, ki izriše točke v podatnih koordinatah:

```
plot3(X, Y, Z, 'color', [0,0,0], 'linewidth', 4);
scatter3(X, Y, Z, 20, [0.5 0.5 0.5]);
```




(b) Z uporabo funkcije `antropomorphic_manipulator` izračunajte razdaljo zadnjega sklepa manipulatorja do točke $(3, 3, 4)$ za vektor parametrov stanja $(0.2, 0.1, 0.3)$. Najprej izračunajte matrike za dano stanje manipulatorja, nato izhodiščno točko $(0, 0, 0)$ pomnožite z matriko zadnjega sklepa, da dobite položaj zadnjega sklepa v prostoru. Izračunajte Evklidsko razdaljo med točkama in jo izpišite v terminal, rezultat bi moral biti 3.257.

(c) ★ (5 točk) Iz prejšnje naloge izpeljite skripto, ki vizualizira razdaljo od točke $(3, 3, 4)$ pri fiksiranem parametru drugega sklepa za vrednosti parametrov prvega in zadnjega sklepa. Za vrednosti posameznega sklepa izberite ustrezen interval ter nato za posamezne pare vrednosti izračunajte razdaljo do točke. Razdalje lahko shranite v matriko in jo prikažete kot sliko, lahko pa si izberete tudi prikaz v obliki površinskega grafa.



# Naloga 2: Inverzna kinematika

Veliko bolj praktično, kot ugotavljanje razdalje do določene točke za izbrani nabor parametrov, pa bi bilo, da bi robotskemu sistemu podali ciljno točko (in rotacijo), robotski

sistem pa bi sam našel ustrezne parametre, v katerih bi manipulator bila v takem stanju. Takemu problemu rečemo povratna oziroma *inverzna* kinematika (okrajšano IK). Le-ta ni uporabna samo v robotiki, uporablja se tudi pri animaciji računalniških likov v računalniških igrah in animiranih filmih.

***Vprašanje:*** Zakaj je inverzna kinematika v splošnem težko rešljiv problem? Namig: koliko je lahko naborov parametrov, ki robotski manipulator postavijo v določeno stanje?

V nadaljevanju si bomo pogledali osnove reševanja problemov inverzne kinematike. Osnovna ideja je preprosta: za podane začetne parametre stanja manipulatorja, prostor preiskovanja ter ciljno točko iščemo take parametre, ki bodo minimizirali razdaljo zadnjega sklepa manipulatorja do ciljne točke. Ker pa se stvari pri implementaciji hitro zapletejo, boste sami implementirali samo nekaj osnovih idej, potem pa boste uporabili malo bolj napreden algoritem za reševanje problemov inverzne kinematike z uporabo stohastične optimizacije.

(a) ★ (5 točk) Implementirajte preprost algoritem, ki optimizira samo položaj prvega sklepa antropomorfnega manipulatorja (druga dva pa sta statična). Uporabite funkcijo `antropomorphic_manipulator`, ki ste jo implementirali v prejšnji nalogi. Naloge se lahko lotite s katero od iterativnih metod optimizacije (na primer Newton-ovo metodo ali njenimi izpeljankami).

***Vprašanje:*** Tak način reševanja je primeren za en parameter, bi se obnesel tudi za optimizacijo večjega števila parametrov?

(b) V nadaljevanju bomo preizusili bolj kompleksen način za iskanje rešitve problema inverzne kinematike. Oglejte si kodo v priloženi datoteki `manipulator_solve.m`. Algoritem v tej datoteki rešuje problem inverzne kinematike z uporabo iterativne metode spusta po koordinatah[1]. Algoritma sicer ni potrebno v celoti razumeti, saj ne sodi v ožjo snov predmeta, morate pa se po njem znajti dovolj, da ga uporabite v lastni kodi. Poleg tega se je potrebno zavedati tudi omejitev algoritma, saj le-ta predstavlja zelo preprost način reševanja prolema inverzne kinematike, ne konvergira vedno, poleg tega pa samo delo upošteva omejitve med posameznimi sklepi robotskega manipulatorja.

Metodo boste najprej preizkusili v preprostem simuliranem okolju, ki vam omogoča preprosto vizualizacijo stanja manipulatorja v prostoru. Oglejte si skripto `demo_simulation`, ki vzpostavi simulacijsko okolje ter inicializira manipulator, nato pa manipulator premika med točkami, ki so razporejene v krožnici okoli manipulatorja. Uporabite metodo `manipulator_solve` za premik manipulatorja iz stanja $(\frac{\pi}{2}, \frac{\pi}{2} + 0.5, 1)$ na točko $(2, 2, 4)$. S pomočjo funkcije `manipulator_solve` izračunajte nove parametre manipulatorja za to točko, nato pa poženite animacijo premika s funkcijo `manipulator_animate`.

(c) Animirajte antropomorfni manipulator, da se bo premikala med tremi točkami v zaporedju, ki nakazuje reševanje problema Hanoiskih stolpičev[2] za štiri diske.

(d) ★ (5 točk) Algoritem iz prejšnje točke razširite, da bo deloval za poljubno število diskov (torej za poljuben in nastavljiv $N \geq 3$).

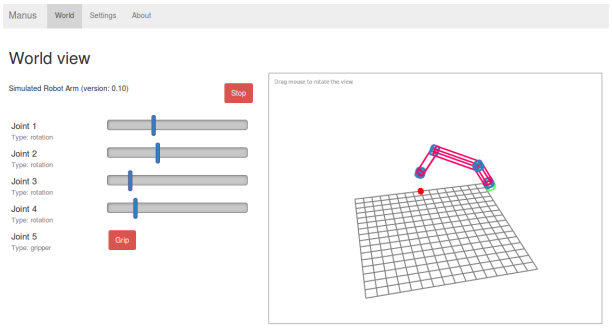
[1] `https://en.wikipedia.org/wiki/Coordinate_descent`

[2] `http://en.wikipedia.org/wiki/Tower_of_Hanoi`

# Naloga 3: Krmiljenje robotskega manipulatorja

V okviru prejšnjega poglavja smo manipulator samo simulirali, v okviru tega poglavja pa si bomo pogledali, kako lahko na enak način krmilimo tudi pravi robotski manipulator. S tem namenom je bil v Laboratoriju za umetne vizualne spoznavne sisteme razvit robotski manipulator za pedagoške potrebe, ki ga lahko iz lastnega računalnika krmilite preko lokalne mreže. V laboratoriju imamo za študente na voljo devet manipulatorjev, ki so med delovnim časom na voljo tudi izven terminov za vaje. Poleg tega imate na voljo tudi sliko navideznega računalnika (za emulator VirtualBox), ki vam omogoča, da krmiljenje manipulatorja preizkušate tudi doma, vendar pa v tem primeru nimate na voljo možnosti manipulacije okolja.

V materialu za vaje je nekaj funkcij, ki služijo kot most med robotskim manipulatorjem, s katerim komunicirajo preko HTTP zahtev. Pričakovano je, da v okviru vaje vzpostavite sistem ter preko vaših skript krmilite manipulator v scenarijih, ki so navedeni v nadaljevanju.

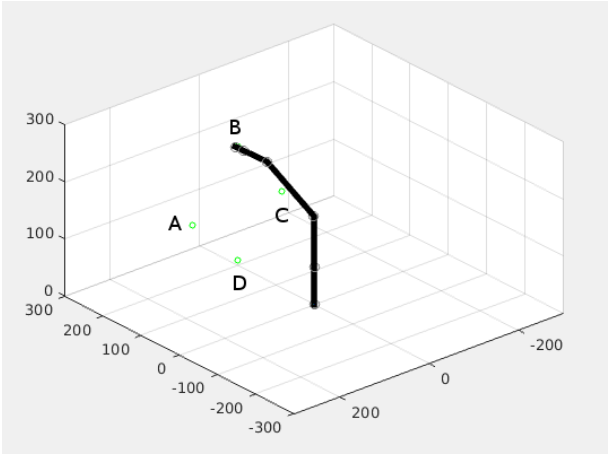



(a) V okviru te točke boste preverili komunikacijo deluje. Manipulator priklopite tako, da jo preko USB kabla najprej povežete z računalnikom, nato pa še priklopite zunanje napajanje. V primeru, da bo manipulator prišel v položaj, ko bo kakšen motor zataknil v oviro, čimprej izklopite napajanje motorjev z uporabo strojnega stikala na krmilni plošči, da se prepreči okvara motorjev.

*Opozorilo*: Ker boste robotske manipulatorje lahko uporabljali le v fakultetnih prostorih, na samih vajah pa verjetno ne bo dovolj časa, niti ni na voljo dovolj manipulatorjev za vse študente, se z asistentom dogovorite za termine, ko bodo roboti na voljo za uporabo v laboratoriju. Doma lahko že vse vaje vsaj okvirno preverite v simulacijskem načinu, z uporabo VirtualBox navideznega računalnika.

(b) V okolju *Matlab/Octave* si oglejte priloženo skripto `demo_remote`, ki demonstrira uporabo funkcij za krmiljenje manipulatorja. Skripta uporablja iste funkcije, ki so bile uporabljene v prejšnji vaji, edina razlika je, da se sedaj parametri manipulatorja pridobijo preko spletnega strežnika, prav tako se preko njega manipulator tudi krmili.

Napišite skripto, ki robotski manipulator krožno premika med štirimi točkami kvarata nad delovno površino. Koordinate točk si izberite sami, vendar bodite pozorni, da je ena virtualna enota enaka enemu milimetru v naravi ter, da točke ne bodo take, da bi manipulator prišel v ilegalno stanje ter se s tem kvarila motorjev. Manipulator naj se torej premakne v točko $A$, nato v točko $B$, točko $C$, točko $D$. Nato sledi spet točka $A$ in tako dalje.




(c) Implementirajte manever manipulatorja, da bo le-ta nad delovno površino "narisal" črko A, predstavljajte si, da jo rišete na papir, ki leži na delovni površini. Naloge se lotite z neposrednim krmiljenem vsakega motorja posebej, izračunajte torej kombinacijo sklepov, za vsak položaj v gibu. Pazite na to, da se roka nikoli ne zadane v podlago. Pri pisanju črke bodite pozorni na dvig manipulatorja, ko se morate premakniti iz enega mesta na drugo brez risanja črte (torej pri risanju prečne črte pri črki A).

(d) Enako nalogo kot v prejšnji točki, implementirajte še z uporabo inverzne kinematike (bodite pozorni, da so točke razporejene v pravilnem vrstnem redu, ter, da jih izberete dovolj na gosto, saj ne morete krmiliti vsakega sklepa posebej).

(e) ★ (10 točk) Eno izmed implementacij risanja črke A lahko v navezavi s pravim robotskim manipulatorjem izpopolnite tako, da v prijemalo postavite pravi svinčnik ali kemik, na delovno površino pa list papirja (delovno površino res zaščitite pred poškodbami!). Roka naj na list nariše črko, število dodatnih točk pa bo odvisno od kvalitete izrisa. Na zagovoru vam ni potrebno ponoviti celotnega procesa, lahko prinesite izvorno kodo, list z izrisano črko ter posnetek risanja.

(f) ★ (10 točk) Z uporabo inverzne kinematike napišite program, ki bo na vnaprej določenem mestu na delovni površini pobral kocko ter jo prenesel na drugo vnaprej določeno mesto. Nato se bo vrnil po novo kocko in jo postavil na prvo kocko. Ta proces se naj ponavlja vmes pa lahko čakate nekaj sekund za postavitev nove kocke ali pa na pritisk tipke (funkcija `pause`). Kot je očitno, morate to nalogo v večjem delu izdelati v laboratoriju s pravo robotsko roko. Za polno število točk morate eno na drugo zložiti vsaj pet kock. Končni rezultat lahko posnamete in na zagovor prinesete posnetek in izvorno kodo.