ELEKTROTEHNIČKI FAKULTET OSIJEK

ZAVOD ZA AUTOMATIKU I PROCESNO RAČUNARSTVO

**ROBOTICS TOOLBOX**

**Upute za upotrebu**

**Robert Cupec**

Osijek, 03.04.2012.

**1. Objekti**

Robot, predmeti kojima on rukuje te objekti u radnoj okolini robota prikazuju se pomoću podatkovnih struktura koje se u nastavku nazivaju *objekti*. Površina objekta sastoji se od ravnih poligonalnih ploha sa tri ili više vrhova. Svakom objektu pridružen je koordinatni sustav. Vrhovi objekta definirani su matricom X koja predstavlja pripadno polje podatkovne strukture kojom je objekt opisan. Svakom vrhu objekta odgovara jedan stupac matrice X tako da ta matrica ima onoliko stupaca koliko objekt ima vrhova. Svaki stupac matrice X ima 4 elementa, koji predstavljaju homogene koordinate određenog vrha objekta s obzirom na koordinatni sustav objekta. Objekti se mogu stvarati i prikazivati na zaslonu računala pomoću funkcija opisanih u nastavku.

**cuboid**

objekt = cuboid(a, b, c)

Funkcija **cuboid** stvara objekt oblika kvadra. Ishodište koordinatnog sustava kvadra je u njegovom centru mase. Duljine stranica kvadra su a, b i c, pri čemu je a duljina stranice paralelne s x-osi koordinatnog sustava kvadra, b duljina stranice paralelne s njegovom y-osi, a c duljina stranice paralelne s njegovom z-osi.

**merge3dobjs**

objekt3 = merge3dobjs(objekt1, objekt2)

Funkcija **merge3dobjs** stvara objekt3 spajanjem objekata objekt1 i objekt2.

**plot3dobj**

plot3dobj(objekt)

Funkcija **plot3dobj** prikazuje objekt na zaslonu računala.

Koordinate vrhova nekog objekta se mogu transformirati iz jednog koordinatnog sustava u drugi množenjem s odgovarajućom matricom homogenih transformacija, što je ilustrirano sljedećim primjerom.

**Primjer 1:**

Sljedeći niz naredbi stvara scenu na kojoj se nalaze dva kvadra i prikazuje je na zaslonu računala.

% stvori objekt obj1

obj1=cuboid(1,1,4);

T10 = [1 0 0 0;

0 1 0 2;

0 0 1 2;

0 0 0 1];

obj1.X = T10\*obj1.X;

% stvori objekt obj2

obj2=cuboid(4,1,2);

T20 = [1 0 0 0;

0 1 0 -2;

0 0 1 1;

0 0 0 1];

obj2.X = T20\*obj2.X;

% stvori scenu spajanjem objekata obj1 i obj2

scena = merge3dobjs(obj1, obj2);

% prikazi scenu na zaslonu racunala

plot3dobj(scena);

Rezultat je prikazan na slici 1.

cuboids

**Sl. 1.**  *Scena koja se sastoji od dva kvadra*.

**2. Robot**

Izgled, kinematički model i fizikalna svojstva robota definiraju se podatkovnom strukturom koja sadrži sljedeća polja:

n - broj zglobova;

L - polje od n elemenata. Svaki element opisuje jedan članak odnosno zglob robota;

g - gravitacijski vektor prikazan s obzirom na bazni koordinatni sustav robota.

Svaki članak robota i njemu pridruženi zglob prikazani su podatkovnom strukturom koja sadrži sljedeća polja:

ksi - parametar tipa zgloba;

DH - vektor kinematičkih parametara [** *d* *a* **]T;

B - matrica kojom je definiran izgled članka. Članak se može sastojati od jednog ili više kvadara. Svaki kvadar opisan je jednim retkom matrice B. Prva tri elementa retka predstavljaju koordinate centra mase kvadra u odnosu na koordinatni sustav članka. Posljednja tri elementa retka predstavljaju duljine stranica kvadra;

m - masa članka;

dc - koordinate centra mase članka s obzirom na koordinatni sustav članka;

Dc - tenzor inercije članka s obzirom na njegov centar mase.

Na temelju navedene strukture može se pomoću funkcije **robotmesh** stvoriti objekt koji odgovara definiranom robotu. Takav se objekt zatim može prikazati na zaslonu računala funkcijom **plot3dobj** opisanom u prvom poglavlju.

**robotmesh**

robmesh = robotmesh(robot, q)

Funkcija **robotmesh** stvara objekt robmesh koji predstavlja robota definiranog podatkovnom strukturom robot. Varijable zglobova robota zadaju se vektorom q.

**Primjer 2:**

Sljedeći niz naredbi stvara dvoosni robotski manipulator i prikazuje ga na zaslonu računala.

% broj osi

robot.n = 2;

% varijable zglobova

q(1) = -75\*pi/180;

q(2) = 30\*pi/180;

% gravitacijski vektor

robot.g = [0 0 -9.81];

% prvi clanak

robot.L(1).ksi = 1;

robot.L(1).DH = [q(1) 0 0 -pi/2]';

robot.L(1).B = [0 15 0 10 40 10];

% drugi clanak

robot.L(2).ksi = 1;

robot.L(2).DH = [q(2) 10 0 pi/2]';

robot.L(2).B = [0 0 15 10 10 40;

0 0 40 5 5 10;

0 0 46 10 5 2;

-3 0 50 2 5 6;

3 0 50 2 5 6];

% prikaz robota na zaslonu racunala

robmesh = robotmesh(robot, q);

plot3dobj(robmesh)

Rezultat je prikazan na slici 2.

manipulator_12b

**Sl. 2.**  *Dvoosni robotski manipulator*.

**3. Planiranje trajektorije**

Podržane su dvije metode planiranja trajektorije interpolacijom na temelju zadanih točaka: linearna interpolacija s kvadratnim prijelazima i Ho-Cookova metoda. Prva metoda implementirana je funkcijom **linkvadint2**, a druga funkcijom **hocook**.

**linkvadint2**

[W, dW, ddW] = linkvadint2(Wd, tu, T, ksi)

Funkcija **linkvadint2** na temelju prolaznih točaka zadanih matricom Wd generira kontinuiranu trajektoriju linearnom interpolacijom s kvadratnim prijelazima. Svaki stupac matrice Wd predstavlja jednu prolaznu točku. Razmatraju se dvije osnovne vrste trajektorija: trajektorije u prostoru zglobova i trajektorije u prostoru konfiguracije alata. Ukoliko se radi o trajektoriji u prostoru zglobova, prvih *n* elemenata svakog stupca matrice Wd predstavljaju vrijednosti varijabli zglobova za danu točku trajektorije. Ukoliko se želi prikazati i otvaranje odnosno zatvaranje hvataljke, tada pretposljednji element stupca predstavlja otvorenost hvataljke. Ukoliko se radi o trajektoriji u prostoru konfiguracije alata, prvih *n* = 6 elemenata stupca matrice trajektorije predstavljaju komponente vektora konfiguracije alata  za danu točku trajektorije. U oba slučaja, zadnji element stupca predstavlja vrijeme. Vremena ubrzanja zadaju se vektorom tu koji ima onoliko elemenata koliko ima prolaznih točaka odnosno stupaca matrice Wd. Vrijeme uzorkovanja trajektorije zadaje se argumentom T. Posljednji argument ksi predstavlja vektor koji ima onoliko elemenata koliko stupnjeva slobode gibanja ima robot. Svaki element vektora ksi odgovara jednom parametru trajektorije, tj. jednom retku matrice Wd (izuzev vremena, tj. posljednjeg retka). Ukoliko je neki element vektora ksi jednak 1, odgovarajući parametar trajektorije predstavlja kut u radijanima, a ukoliko je 0, odgovarajući parametar trajektorije predstavlja udaljenost odnosno koordinatu u metrima ili milimetrima. Trajektorija koju funkcija **linkvadint2** vraća kao rezultat prikazana je *matricom trajektorije* dimenzija (*n* + 2) × *m*, gdje je *n* broj stupnjeva slobode gibanja, a *m* broj točaka trajektorije. Svaki stupac matrice trajektorije predstavlja jednu točku trajektorije. Elementi stupca sadrže podatke analogne onima sadržanim u matrici Wd. Osim matrice trajektorije W, funkcija **linkvadint2** vraća i matrice dW i ddW koje predstavljaju prvu odnosno drugu derivaciju trajektorije prikazane u istom formatu kao i matrica W.

**Primjer 3:**

Sljedeći niz naredbi generira trajektoriju linearnom interpolacijom s kvadratnim prijelazima i prikazuje ju na zaslonu računala u obliku vremenskog dijagrama te u 3D prikazu.

% zadavanje prolaznih tocaka

Wd = [0.0 0.0 0.5 0.5;

0.3 0.3 0.5 2.0;

0.5 0.0 0.0 5.0;

0.3 -0.3 0.5 8.0;

0.0 0.0 0.5 9.5]';

% linearna interpolacija s kvadratnim prijelazima

[W, dW, ddW]=linkvadint2(Wd,[0.5 0.5 0.5 0.5 0.5]',0.01,[0 0 0]');

% prikaz trajektorije na zaslonu racunala

figure(1)

plot(W(4,:),W(1:3,:),'b',Wd(4,:),Wd(1:3,:),'ro')

figure(2)

plot3(W(1,:),W(2,:),W(3,:),'b',Wd(1,:),Wd(2,:),Wd(3,:),'ro')

Rezultat je prikazan na slici 3.

trajectory_plot

trajectory_plot_02

**Sl. 3.**  *Trajektorija prikazana vremenskim dijagramom i 3D prikazom*.

**hocook**

[Qc, dQc, ddQc] = hocook(Q, dqgr, ddqgr, Ts)

Funkcija **hocook** na temelju prolaznih točaka zadanih matricom Q generira kontinuiranu trajektoriju Ho-Cookovom metodom. Trajektorija dobivena ovom metodom ima neprekinutu prvu i drugu derivaciju, a brzine i ubrzanja su unutar granica zadanih vektorima dqgr i ddqgr. Vektori dqgr i ddqgr imaju onoliko elemenata koliko robot ima stupnjeva slobode gibanja. Svaki element vektora dqgr predstavlja najveću dozvoljenu brzinu promjene odgovarajuće varijable zgloba. Analogno tome, svaki element vektora ddqgr predstavlja najveće dozvoljeno ubrzanje odgovarajućeg zgloba. Parametar Ts predstavlja vrijeme uzorkovanja trajektorije. Kao rezultat funkcija **hocook** vraća matricu trajektorije istog formata kao i funkcija **linkvadint2**.

**4. Dinamički model robotskog manipulatora**

Dinamički model robotskog manipulatora može se prikazati matričnom diferencijalnom jednadžbom

,

gdje je

**q** - vektor varijabli zglobova,

**** - vektor pogonskih sila/momenata,

**D**(**q**) - tenzor inercije manipulatora,

 - vektor centrifugalnih i Coriolisovih sila,

**h**(**q**) - vektor gravitacijskog djelovanja.

Matrica **D** te vektori **N** i **h** mogu se za zadane vrijednosti **q** i  izračunati pomoću funkcije **dynmodel**.

**dynmodel**

robotNew = dynmodel(robot, q, dq, Ftool)

Funkcija **dynmodel** na temelju parametara robota zadanih podatkovnom strukturom robot opisanom u poglavlju 2 te vektora varijabli zglobova q vektora derivacija varijabli zglobova dq i vektora vanjskih sila koje djeluju na alat robota Ftool izračunava tenzor inercije manipulatora **D**, vektor centrifugalnih i Coriolisovih sila **N** te vektor gravitacijskog djelovanja **h** i sprema ih u polja D, N i h podatkovne strukture robotNew. Ukoliko nema kontakta između robota i njegove okoline, Ftool je nul-vektor od *n* elemenata, gdje je *n* broj osi robota.

**5. Simulacija robota upravljanog digitalnim regulatorom**

Na slici 4 shematski je prikazan sustav upravljanja robotskim manipulatorom. Zadatak regulatora je da varijable zglobova **q** što manje odstupaju od referentne trajektorije **q***r*. Na temelju vrijednosti **q**, , **q***r*,  i , regulator izračunava odgovarajuće vrijednosti momenata **** koji predstavljaju upravljačke signale. Povratna veza po poopćenoj sili **F** omogućuje upravljanje silom dodira.



**Sl. 4.**  *Blokovska shema sustava upravljanja robotskim manipulatorom*.

Sumulacija ovakvog sustava upravljanja može se izvesti pomoću programskog paketa SIMULINK i simulacijskog modela **robman\_sim.mdl**. Ovaj simulacijski model poziva funkciju **roboctrlalg** kojom je implementiran digitalni regulator. Prije pokretanja simulacije treba definirati podatkovnu strukturu robot kojom je opisan robotski manipulator na način objašnjen u poglavlju 2 te podatkovnu strukturu ctrlparam kojom se konfigurira regulator. Struktura ctrlparam ima sljedeća polja

nStates - broj memorijskih varijabli koje prenose informaciju između dva koraka izvođenja algoritma;

x0 - početne vrijednosti memorijskih varijabli;

T - period uzorkovanja;

umax - najveća vrijednost upravljačkog signala;

umin - najmanja vrijednost upravljačkog signala;

bCartesian - ima vrijednost 1 ukoliko se radi o upravljanju u prostoru konfiguracije alata, odnosno 0 ukoliko se radi o upravljanju u prostoru zglobova.

Pored ovih polja korisnik može u strukturu ctrlparam dodati i vlastita polja pomoću kojih može zadati različite parametre regulatora.

**roboctrlalg**

[uc, xNew] = roboctrlalg(robot, ctrlparam, t, q, dq, F,   
qr, dqr, ddqr, x)

Funkcija **roboctrlalg** poziva se iz simulacijskog modela **robman\_sim.mdl** i predstavlja upravljački algoritam za robotski manipulator. Dodavanjem vlastitog koda u funkciju **roboctrlalg** korisnik može implementirati različite digitalne regulatore za upravljanje robotskim manipulatorom. Parametri robota kojim regulator upravlja prosljeđuju se argumentom robot. Konfiguriranje regulatora izvodi se postavljanjem polja podatkovne strukture ctrlparam na željene vrijednosti prije pokretanja simulacije. Argument t predstavlja vrijeme od početka izvođenja simulacije. Varijable zglobova manipulatora sadržane su u vektoru q, a njihove derivacije u vektoru dq. Argument F predstavlja silu kojom okolina djeluje na alat manipulatora. Trenutna referentna vrijednost varijabli zglobova prosljeđuje se regulatoru argumentom qr, dok se argumentima dqr i ddqr. Memorijske varijable prosljeđuju se argumentom x. Funkcija vraća upravljački signal uc koji predstavlja vektor od n elemenata, gdje je n broj osi robota, te nove vrijednosti memorijskih varijabli xNew koje se mogu mijenjati unutar funkcije.

**6. Animacija**

Kretanje robota može se prikazati animacijom pomoću funkcije **dyn3dscene**.

**dyn3dscene**

dyn3dscene(robot, Q, env, plotbox, alpha, beta, T, Tpause)

Funkcija **dyn3dscene** omogućuje prikaz animacije kretanja robota na temelju trajektorije dobivene npr. simulacijom opisanom u poglavlju 5. Robot je definiran argumentom robot koji predstavlja podatkovnu strukturu opisanu u poglavlju 2. Trajektorija koja definira kretanje robota zadaje se argumentom Q koji predstavlja matricu trajektorije opisanu u poglavlju 3. Okolina robota definirana je argumentom env koji predstavlja objekt opisan u poglavlju 1. Okolina može biti samo jedan predmet ili se može sastojati od više objekata grupiranih u jedan objekt kako je opisano u poglavlju 1. Robot i njegova okolina prikazani su u baznom koordinatnom sustavu robota. Skala prikaza definirana je argumentom plotbox koji predstavlja vektor od 6 elemenata. Prva dva elementa predstavljaju minimalnu i maksimalnu x-koordinatu prikaza, druga dva minimalnu i maksimalnu y-koordinatu, a treća dva minimalnu i maksimalnu z-koordinatu. Kut promatranja scene definiran je argumentima alpha i beta, gdje alpha predstavlja azimut, a beta elevaciju virtualne kamere. Argument T predstavlja vrijeme uzorkovanja trajektorije. Brzina izvođenja animacije može se podesiti odgovarajućim izborom argumenta Tpause. Naime, vrijeme između prikaza dvije uzastopne slike animacije na zaslonu računala jednako je vremenu potrebnom za prikaz slike uvećanom za Tpause.