

ROBOTIKA

LABORATORIJSKI PRAKTIKUM

SEBASTJAN ŠLAJPAH  
JURE REJC  
ROMAN KAMNIK  
MATJAŽ MIHELJ

**Univerza v Ljubljani, Založba FE**  
Ljubljana, 2018

---

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani

COBISS.SI-ID=297960448

ISBN 978-961-243-357-4 (pdf)

---

URL: <http://www.robotlab.si/~sebastjans/pubs/>

Založnik: Založba FE, Ljubljana

Izdajatelj: Fakulteta za elektrotehniko, Ljubljana

Urednik: prof. dr. Sašo Tomažič

Slika na naslovnici: Irena Podobnik, Istrska polja, olje na platnu, 80 x 120 cm, 2008

1. elektronska izdaja

# Kazalo

1	VARNOST PRI DELU Z INDUSTRIJSKIMI ROBOTI	1
1.1	Nevarnosti pri delu z roboti	2
1.2	Zahteve in zagotavljanje varnosti pri delu z roboti	4
1.2.1	Zagotavljanje varnosti na nivoju strojne opreme	4
1.2.2	Zagotavljanje varnosti pri razvoju programske opreme	7
1.2.3	Zagotavljanje varnosti v laboratorijskem okolju	7
2	PROSTORSKE TRANSFORMACIJE V MATLABU IN GEOMETRIJSKI MODEL ROBOTA EPSON PS3	9
2.1	Cilj vaje	9
2.2	Splošno o homogenih transformacijskih matrikah	11
2.2.1	Lega in premik	11
2.2.2	Osnovne homogene transformacijske matrike	12
2.3	1. del naloge	13
2.3.1	Seznam funkcij za uporabo v Matlabu	14
2.4	2. del naloge	16
2.4.1	Geometrijski model robota z vektorskimi parametri	17
2.5	Koraki za izvedbo vaje	18
2.6	Rezultati za preverjanje kinematičnega modela za robot Epson PS3	20
3	VODENJE DVOSEGMENTNEGA MANIPULATORJA NA OSNOVI ROBOTA MOTOMAN MH5	21
3.1	Cilj vaje	21
3.2	Dvosegmentni manipulator	21
3.3	Dvosegmetni manipulator na osnovi robota Motoman MH5	24
3.4	Vodenje robota	25

3.4.1	PD regulacija položaja	27
3.5	Koraki za izvedbo vaje	29
4	SIMULACIJA VARJENJA S SINHRONIM DELOM ROBOTOV MOTOMAN MH5	41
4.1	Cilj vaje	41
4.2	Struktura sistema	43
4.3	Robotski mehanizem MH5	43
4.4	Ročna učna enota	44
4.4.1	Pomen uporabljenih tipk	45
4.4.2	Zaslon učne enote	48
4.5	Izvedba 1. dela vaje - kalibracija orodja	49
4.6	Izvedba 2. dela vaje - kalibracija robotov	51
4.7	Izvedba 3. dela vaje - ukazi za sinhrono premikanje dveh robotov	53
4.8	Izvedba 4. dela vaje - usklajeno premikanje dveh robotov	56
5	ROBOT ABB IRB 1600 IN TEKOČI TRAK	61
5.1	Cilj vaje	61
5.2	Struktura sistema	62
5.2.1	Krmilnik ABB IRC5	63
5.2.2	Robotski mehanizem ABB IRB 1600	63
5.2.3	Ročna učna enota	64
5.3	Uporabniški vmesnik za premikanje robota	66
5.4	Tekoči trak	71
5.4.1	Povezava z robotskim krmilnikom	71
5.4.2	Definiranje lege tekočega traku in relacij med koordinatnimi sistemi	71
5.5	Kreiranje, pisanje, urejanje in zaganjanje programov	74
5.5.1	Koraki za izvedbo vaje z vzorčnim programom	76
5.6	Koraki za učenje koordiniranega dela robota in tekočega traku	79
6	MANIPULACIJA OBJEKTOV Z ROBOTOM EPSON E2S651 IN ROBOTSKIM VIDOM	81
6.1	Cilj vaje	81
6.2	Lastnosti sistema	82
6.3	Programsko okolje Epson RC+	82
6.3.1	Osnove o pisanju programov	83
6.3.2	Vključitev pogonskih motorjev	85

6.3.3	Upravljanje z digitalnimi vhodi in izhodi	85
6.3.4	Okno za premikanje robota in učenje točk	87
6.4	1. del naloge	89
6.4.1	Ukazi, potrebni za izvedbo naloge	89
6.4.2	Koraki za izvedbo naloge	90
6.5	Matlab in umetni vid	92
6.5.1	Video sistem	92
6.5.2	2. del naloge	93
6.5.3	Koraki za izvajanje naloge	101
7	<b>SODELUJOČI ROBOT ABB YUMI IN ROBOTSKI VID</b>	103
7.1	Cilj vaje	103
7.2	Struktra sistema	103
7.3	Dvoročna manipulacija	105
7.3.1	Ročno vodenje robota	105
7.3.2	Kreiranje delovnega objekta	107
7.3.3	Uporaba prijemala	108
7.3.4	Program RAPID	110
7.4	Robotski vid	111
7.5	Prepoznavna in manipulacija objektov	112
7.5.1	Priprava programa	113
7.5.2	Kalibracija kamere	114
7.5.3	Povezava kamere in robota	116
7.5.4	Definiranje naloge robotskega vida	118
7.5.5	Prijemanje in manipulacija objekta	122
7.5.6	Prepoznavna in manipulacija več objektov	123
7.5.7	Zagon v avtomatskem načinu	125



## Poglavje 1

# VARNOST PRI DELU Z INDUSTRIJSKIMI ROBOTI

Industrijski robot je pozicijsko vodena, programibilna in večopravilna naprava, ki se giblje vzdolž več prostostnih stopenj v prostoru. Namenjen je manipulaciji materiala, obdelovancev in orodij pri izvajanju različnih delovnih nalog in programiranih gibov.

Glede na zagotavljanje varnosti uvajanje industrijskih robotov v proizvodnjo predstavlja dva nasprotna si vidika. Na eni strani uporaba industrijskih robotov v nevarnem in človeku škodljivem okolju povečuje človekovo varnost, saj uporaba robotov za avtomatsko varjenje, kovanje, peskanje, barvanje, itd. omogoča, da je človek umaknjen iz neprijaznega in nevarnega delovnega okolja. Na drugi strani pa lahko roboti med obratovanjem sami ogrožajo varnost delavcev. Pri delu z roboti so možni nesrečni slučajji, lahko tudi tragični, če ni ustrezno poskrbljeno za zagotavljanje varnosti.

Glavna nevarnost pri delu z roboti preti človeku v samem delovnem prostoru robota. Robot je sposoben prostega gibanja v širokem prostoru, sposoben je hitrih nepredvidenih gibov in hitrih sprememb konfiguracije. Navedeno lahko predstavlja neposredno ogrožanje varnosti osebe, ki dela ali stoji v bližini robota. Zato je potrebno pri vsaki robotski celici oceniti, kakšno je tveganje za varnost, in uvesti ukrepe za zmanjšanje možnosti nesreč.

Nepričakovano gibanje robota lahko povzroči okvara sistema ali človeška napaka. Med te prištevamo:

- Nepredvideno obnašanje robota, katerega vzrok je najpogostje napaka v robotskem regulacijskem sistemu.
- Prekinitev pomembnih kabelskih povezav, ki je posledica robotskega gibanja oz. drugih dejavnikov iz okolja.
- Napaka pri prenosu podatkov, ki povzroči večji ali hitrejši gib robota od nastavitvev.

- Napaka ali okvara delovanja orodja na vrhu robota, npr. varilne pištole, prijemala itn.
- Programske napake kot posledica človeške napake ali druge napake v delovanju regulacijskega sistema.
- Premajhna ponovljivost gibov ali izraba robotskega mehanizma, kar lahko botruje v napakah v gibanju robota.
- Nekompatibilnost vpenjal in drugih orodij v robotski celici.

## 1.1 Nevarnosti pri delu z roboti

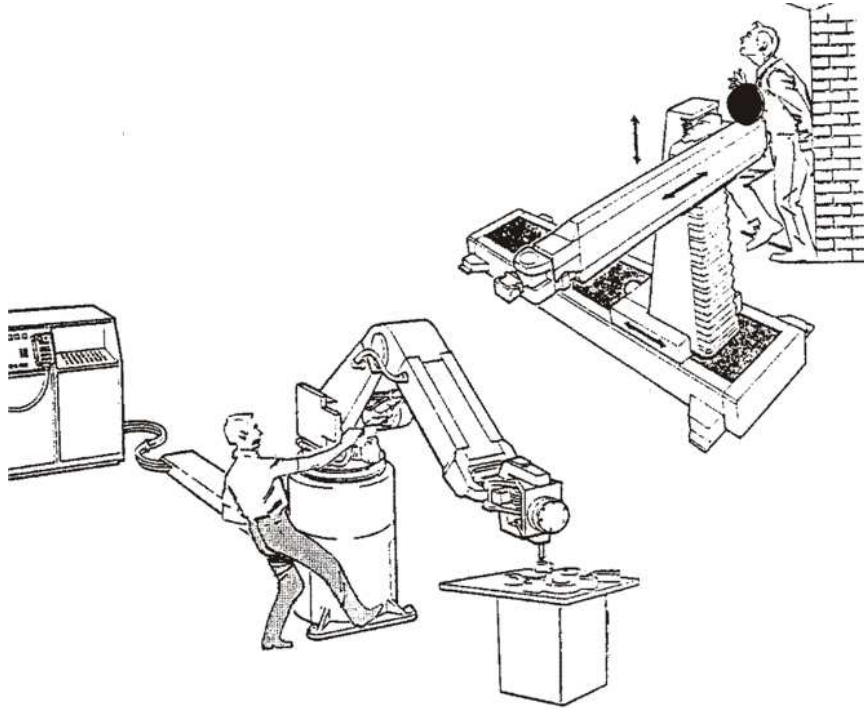
V osnovi obstajajo tri potencialne nevarnosti pri delu z industrijskimi roboti:

- **Nevarnost trka** je možnost, da gibajoči se robot ali orodje, ki ga robot nosi, zadene operaterja. Trk je lahko posledica nepričakovanega giba robota ali izpustitve obdelovanca.
- **Nevarnost stisnjenja** je nevarnost, da robot med gibanjem v bližini objektov, ki so fiksni, kot npr. stroji, oprema ali ograja, stisne operaterja. Nevarnost stisnjenja obstaja tudi pri delu ob vozičkih, tekočih trakovih, paletah in drugih transportnih mehanizmih.
- **Ostale nevarnosti**, ki so specifične posamezni robotski aplikaciji, kot npr. nevarnosti udara električnega toka, vplivov varilnega oblaka, opeklin, strupenih snovi, sevanja, prekomernega zvoka.

Gornje nevarnosti izvirajo iz naslednjih vzrokov:

- **Nevarnosti regulacijskega sistema:** To so nevarnosti napak, ki se pojavijo v robotskem krmilniku, kot so npr. programske napake, napake zaradi interference signalov ter napake v hidravličnih, pnevmatskih ali električnih podsistemih, povezanih z robotom.
- **Mehanske nevarnosti:** V ta razred sodijo nevarnosti, ki so posledica mehanskih lastnosti obdelovancev ali orodij, ki jih prenaša robot. Te so npr. ostri robovi, večje mase ali nezastrite elektrode. Zaradi mehanskih napak lahko robotsko prijemalo nepredvideno izpusti obdelovanec. Vzroki mehanskih napak so prekomerna obremenitev, korozija, utrujanje materiala in pomanjkljivo vzdrževanje.
- **Nevarnosti okolja:** Uporaba robotov lahko v določenih situacijah povzroči tudi tveganja iz okolja. Tvrsten primer so varilne robotske celice, od katerih se širijo varilni plini, varilno iskrenje ter leteči delci. Podobno tveganje predstavljajo tudi prah, vlaga, ionizirajoče in neionizirajoče sevanje, laserski žarki, ultravijolična svetloba ter gorljivi in eksplozivni plini.





Slika 1.1: Nevarnost trka in nevarnost stisnjenja pri delu z industrijskimi roboti

- **Nevarnosti človeških napak:** V večini robotskih celic mora operater delati v bližini robota ali vstopati v njegov delovni prostor. V tem primeru je izpostavljen nevarnosti trka ali stisnjenja, ki lahko nastopi med programiranjem, učenjem gibanja, vzdrževanjem ali delom v bližini robota, npr. vlaganjem ali jemanjem obdelovancev iz celice. Slabo poznavanje opreme je glavni vzrok za človeške napake pri delu z roboti.
- **Nevarnosti perifernih naprav:** V večini robotskih celic robot dela v povezavi s perifernimi enotami, kot so obdelovalni stroji, tekoči trakovi, obdelovalna orodja, stiskalnice. Tovrstna oprema prav tako lahko predstavlja varnostno tveganje, če so nevarni deli v dosegu operaterja in niso zaščiteni z varnostnimi ograjami.

Poročila o nesrečah z industrijskimi roboti razkrivajo, da se večina nesreč dogodi, ko operater vstopi v robotski delovni prostor, ko se je robot že ustavil oz. se giblje z nižano hitrostjo, nenadoma pa se gibanje robota spremenilo.

## 1.2 Zahteve in zagotavljanje varnosti pri delu z roboti

**Splošne zahteve za varno delovanje industrijske strojne opreme** predvidevajo, da morajo biti vsi gibajoči se deli opreme, vsak del prenosnih sistemov in vsak nevaren del varno zakriti. Izjeme obstajajo v primerih, ko so ti deli v takšnem položaju ali so takšne konstrukcije, da so že sami po sebi varni, kot da bi bili zakriti. Pri klasičnih strojih so nevarni deli običajno vgrajeni v njegovi notranjosti. Delovanje strojev je pod popolno kontrolo človeka in so zato vzroki nesreč večinoma pripisani človeškemu faktorju. V nasprotju s stroji pa je pri robotski celici lahko potencialno nevarna širša okolica robota, ki obsega celoten robotov delovni prostor in tudi bližnjo okolico v primeru letočih delcev ali kosov. Zaradi tega je potrebno skrbno proučiti, do kod sega področje nevarnosti in tega ustrezno zaščititi.

### 1.2.1 Zagotavljanje varnosti na nivoju strojne opreme

**Varnostna zaščita** se na nivoju strojne opreme lahko izvaja na treh nivojih:

**Nivo 1** je nivo varovanja obsega celotne robotske celice. Običajno je varovanje izvedeno s fizičnim ograjevanjem s kombinacijo mehanskih ograj in vrat. Kot opcija so lahko uporabljene tudi naprave za zaznavanje prisotnosti.

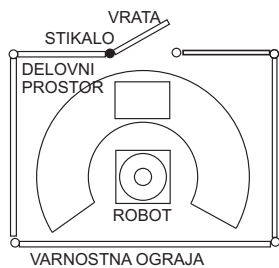
**Nivo 2** vključuje nivo varovanja človeka, ki se nahaja v delovnem prostoru robota. Običajno je varovanje izvedeno s senzorji za zaznavanje prisotnosti človeka. Z razliko s predhodnim nivojem, kjer gre predvsem za ograjevanje, je v tem primeru poudarek na zaznavanju prisotnosti operaterja.

**Nivo 3** je nivo varovanja človeka v neposredni bližini robota. Varovanje na tem nivoju se izvaja z zaznavanjem prisotnosti človeka ali ovir v bližini robota ali pa neposrednega stika z robotom ter posledično s takojšnjo zaustavitvijo delovanja. Za ta namen so uporabljane naprave za merjenje položaja človeka in različni senzorji, kot so npr. senzorji sil in navorov ali kontaktni senzorji dotika.

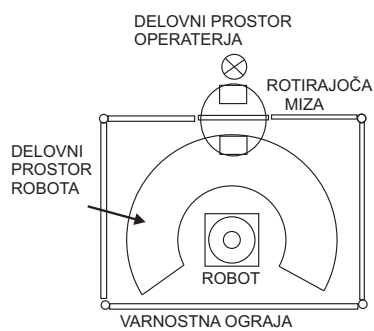
V večini robotskih aplikacij je zahtevan vsaj en nivo varovanja. Glede na oceno tveganja je mogoče izvajati več nivojev varovanja hkrati.

Spodnje slike prikazujejo več primerov **prvega nivoja** varovanja, kjer operater praviloma ne vstopa v samo robotsko celico. Na sliki 1.2 je prikazano **fizično ograjevanje robotske celice z ograjo z vrati**. Operater lahko vstopi v robotsko celico samo v primeru, ko robot ni v obratovanju. Če vstopi med obratovanjem, stikalo na vratih izklopi delovanje. V drugem in tretjem primeru sta delovni prostor operaterja in robota popolnoma ločena. Vstavljanje

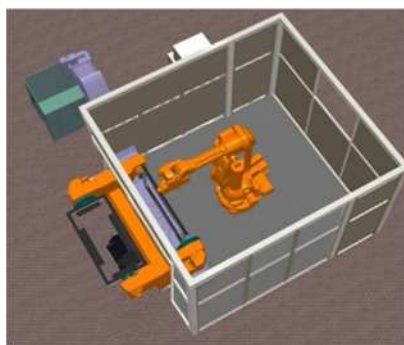
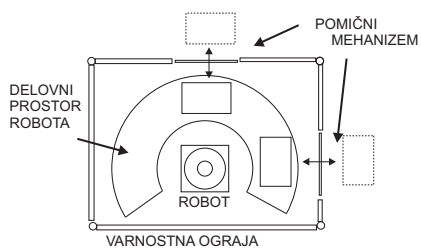
obdelovancev in jemanje obdelavancev iz celice je izvedeno preko rotirajoče mize (glej sliko 1.3) ali pomičnih mehanizmov (glej sliko 1.4).



Slika 1.2: Prvi nivo varovanja s f zično ograjo in vrati



Slika 1.3: Prvi nivo varovanja s f zično ograjo in rotirajočo mizo



Slika 1.4: Prvi nivo varovanja s f zično ograjo in pomičnimi mehanizmi

Na **drugem nivoju** varovanja, pri katerem operater lahko vstopa v robotsko celico, je varovanje izvedeno na osnovi **senzorjev za ugotavljanje prisotnosti operaterja**. To so običajno optični senzori, ki delujejo na principu zaznavanja prekinitve žarka, postavljeni v formacijo optičnih zaves. Alternativa je uporaba senzorskih preprog, ki na osnovi izmerjenega tlaka na podlago zaznavajo položaj operaterja. V osnovi naj bi bili senzori za ugotavljanje prisotnosti uporabljeni le kot sekundarna oblika zagotavljanja varnosti, in to le v primerih, ko je nujno potreben omejen dostop do robota.

Podobno velja za zagotavljanje varnosti na **tretjem nivoju**. **Senzorji za zaznavanje dotika z robotom** so nameščeni na robotske segmente ali na vrh robota. Ta pristop se uporablja v primerih celic z manjšimi roboti, kjer operater med obratovanjem stoji v bližini robota. Signal, ki ponazarja dotik z robotom, povzroči hipno izključitev obratovanja robotske celice.

**Tipka za izklop v sili** je pomembna pri zagotavljanju varnosti, saj operaterju omogoča hitro zaustavitev gibanja robota. Nameščena je na več mestih v robotski celici in je nujno velika ter rdeče obarvana, da je lahko opazna in dosegljiva. Praviloma je nameščena na robotskem krmilniku, na enoti za ročno učenje ter na ograji robotske celice. Vse varnostne naprave, kamor spada tudi tipka za izklop v sili, so zaradi čim hitrejšega izklopa obratovanja s krmilnikom povezane preko žične logike in niso del programske opreme.

### 1.2.2 Zagotavljanje varnosti pri razvoju programske opreme

**Programiranje in učenje robotskega gibanja** v delovnem prostoru robota se izvaja z ročnim vodenjem robota preko točk, ki jih robotski krmilnik pomni in jih nato v avtomatskem načinu izvaja. Za ta namen je uporabljena enota za ročno učenje ali uporabniški vmesnik na osebem računalniku. Možno je tudi učenje s fzičnim vodenjem vrha robota vzdolž trajektorije gibanja, ki si jo robotski krmilnik zapomni in izvaja. V obeh primerih mora biti operater med učenjem znotraj delovnega prostora robota, torej blizu robota. Med učenjem je zato za zagotavljanje varnosti potrebno biti pozoren na:

- Operater, ki robot uči, mora biti za to dobro usposobljen, seznanjen z vsemi nevarnostmi in upoštevati ukrepe za zagotavljanje varnosti.
- Med učenjem gibanja se robot ne sme gibati z visokimi hitrostmi.
- Operater mora imeti lahek in hiter dostop do tipke za izklop v sili.
- Operater mora v vsakem trenutku stati na mestu, kjer je majhna možnost, da ga robot stisne k fksnim objektom v celici ali da ga poškoduje v primeru okvare. Hkrati pa mora poskrbeti, da ima dober pregled nad obratovanjem.
- Priporočljivo je, da je pri učenju prisoten opazovalec, ki se nahaja izven delovnega področja robota, in ima dostop do takojšnjega izklopa v sili.
- Kjer je to potrebno, mora operater nositi zaščitno opremo in zaščitno obleko. Zaščitna čelada je obvezna, če obstaja možnost poškodbe glave.
- Ročna učna naprava mora biti takšna, da omogoča gibanje robota samo v primeru, ko operater drži posebno tipko.

### 1.2.3 Zagotavljanje varnosti v laboratorijskem okolju

V Laboratoriju za robotiko na Fakulteti za elektrotehniko Univerze v Ljubljani je za varno delo z roboti poskrbljeno na naslednji način:

- Delovni prostor robotov je označen na tleh z rumeno/črnim trakom oz. je omejen z ovirami v obliki ograje ali miz.
- Operater, ki vstopa v delovni prostor, mora obvezno nositi zaščitna sredstva ter mora biti obvezno tisti, ki preko učne enote vodi robota.
- Na vidnih mestih v vsaki celici so tipke za izklop v sili, ki jih lahko uporabijo tudi soudeleženci in ne samo oseba, ki v tistem trenutku preko učne enote vodi robot.
- Pri poskusnem zagonu v delovnem prostoru ne sme biti nihče in tudi hitrosti premikanja robota morajo biti počasne.



## Poglavje 2

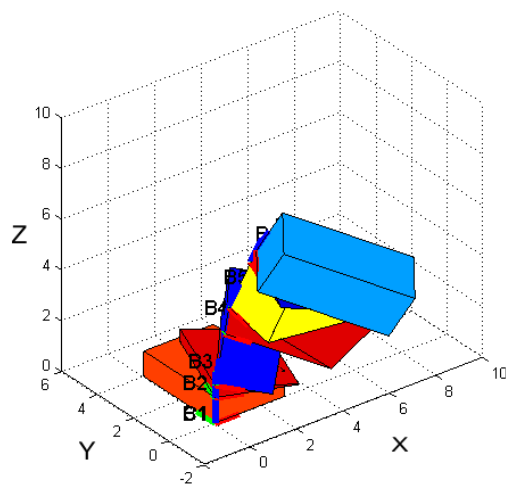
# PROSTORSKE TRANSFORMACIJE V MATLABU IN GEOMETRIJSKI MODEL ROBOTA EPSON PS3

### 2.1 Cilj vaje

Vaja je namenjena spoznavanju programskega paketa Matlab kot orodja za delo s prostorskimi transformacijami oz. homogenimi transformacijskimi matrikami.

#### 1. del

Za izpeljavo tega dela naloge uporabite že pripravljene funkcije za manipulacijo objektov v prostoru. Navodila in opisi funkcij sledijo v nadaljevanju. Naloga zahteva, da v okolje postavite vsaj 7 objektov (kvadri in klini) in zapišete

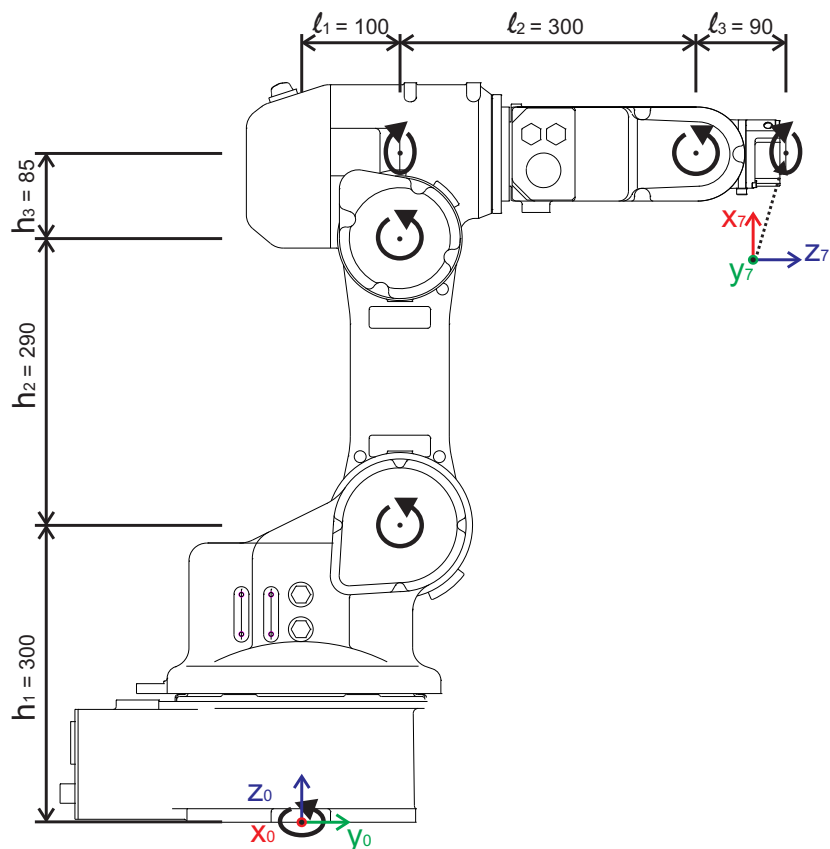


Slika 2.1: Primer spirale

ukaze, ki iz kvadrov in klinov sestavijo spiralo. Končna rešitev je prikazana na sliki 2.1. Uporabite programsko okolje Matlab in že pripravljeno datoteko z imenom *Spirala.m*, kamor vpisujete ustrezne vrstice kode.

## 2. del

Z metodo vektorskih parametrov določite direktni geometrijski model za robotski manipulator **Epson PS3** (Slika 2.2) s šestimi prostostnimi stopnjami. Glede na podano začetno konfiguracijo robota in podan referenčni koordinatni sistem, zapišite tabelo z vektorskimi parametri in jih uporabite za definiranje homogenih transformacijskih matrik med sklepi robota. Končni rezultat je homogena transformacijska matrika  $T$ , ki predstavlja produkt homogenih matrik in predstavlja geometrijski model robota oz. lego vrha robota glede na referenčni k.s. v odvisnosti od sklepnih spremenljivk. Pravilnost rezultata preverite s premikanjem sklepov realnega robota. Za delo uporabite programski paket Matlab z že pripravljeno predlogo in uporabniškim vmesnikom.



Slika 2.2. Robot Epson PS3 z vrisanim baznim k.s. in dejanskim k.s. na vrhu robota



## 2.2 Splošno o homogenih transformacijskih matrikah

### 2.2.1 Lega in premik

**Homogena transformacijska matrika lahko** opisuje **lego** nekega objekta oz. njegovega koordinatnega sistema glede na drug (lahko referenčni) koordinatni sistem. Če homogena transformacijska matrika opisuje lego, potem podmatrika reda  $3 \times 3$  opisuje **orientacijo**, desni stolpec pa **pozicijo** objekta oz. njegovega koordinatnega sistema.

*Najlažje si pojem lega zapomnimo tako, da si predstavljamo ležeč predmet s koordinatnim sistemom. Predmet miruje! Tako ima neko svojo pozicijo in orientacijo gledano na drug objekt, koordinatni sistem oz. prostor.*

$$H_n = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orientacija
Pozicija

Slika 2.3: Homogena transformacijska matrika, ki opisuje lego

**Homogena transformacijska matrika lahko** predstavlja tudi **premik** objekta oz. njegovega koordinatnega sistema glede na drug (lahko referenčni) ali lasten koordinatni sistem v neko novo lego. V tem primeru podmatrika reda  $3 \times 3$  opisuje **rotacijo**, desni stolpec pa **translacijo** objekta oz. koordinatnega sistema.

*Najlažje si pojem premik zapomnimo tako, da si predstavljamo predmet s koordinatnim sistemom. Predmet je v gibanju! To gibanje je lahko samo translatorno, samo rotacijsko ali kombinacija translacije in rotacije.*

$$H_n = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotacija
Translacija

Slika 2.4: Homogena transformacijska matrika, ki opisuje premik

## 2.2.2 Osnovne homogene transformacijske matrike

$$\mathbf{Rot}(x,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha & 0 \\ 0 & \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

Homogena transformacijska matrika 2.1 predstavlja rotacijo okrog  $X$  osi poljubnega koordinatnega sistema za kot  $\alpha$ . Matrika je lahko uporabljena za opis rotacije okrog  $X$  osi lokalnega koordinatnega sistema, glede na os  $X$  absolutnega koordinatnega sistema oz. glede na os  $X$  poljubnega koordinatnega sistema.

$$\mathbf{Rot}(y,\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \beta & 0 & \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

Homogena transformacijska matrika 2.2 predstavlja rotacijo okrog  $Y$  osi poljubnega koordinatnega sistema za kot  $\beta$ . Matrika je lahko uporabljena za opis rotacije okrog  $Y$  osi lokalnega koordinatnega sistema, glede na os  $Y$  absolutnega koordinatnega sistema oz. glede na os  $Y$  poljubnega koordinatnega sistema.

$$\mathbf{Rot}(z,\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 & 0 \\ \sin \gamma & \cos \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

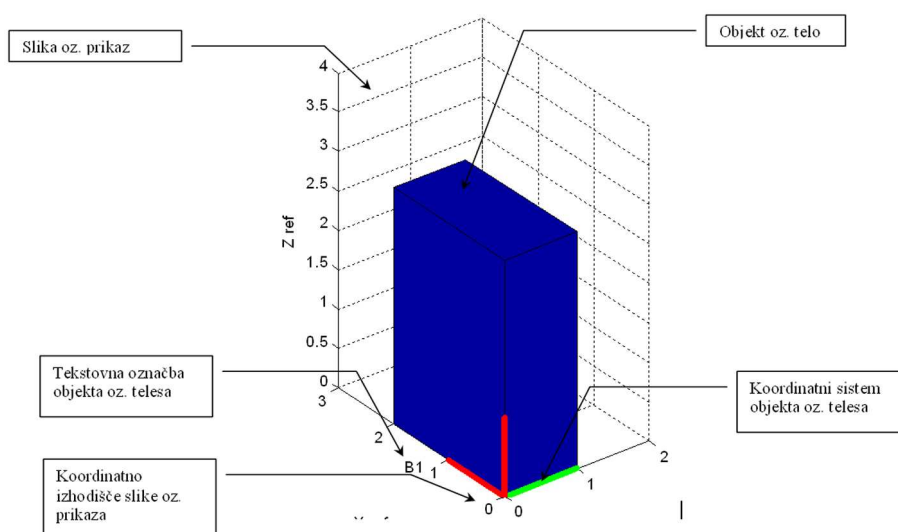
Homogena transformacijska matrika 2.3 predstavlja rotacijo okrog  $Z$  osi poljubnega koordinatnega sistema za kot  $\beta$ . Matrika je lahko uporabljena za opis rotacije okrog  $Z$  osi lokalnega koordinatnega sistema, glede na os  $Z$  absolutnega koordinatnega sistema oz. glede na os  $Z$  poljubnega koordinatnega sistema.

$$\mathbf{Trans}(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

Homogena transformacijska matrika 2.4 predstavlja homogeno translacijsko matriko v smeri  $x$ ,  $y$  in/ali  $z$  osi poljubnega koordinatnega sistema. Matrika je lahko uporabljena za opis translacije v smeri osi lokalnega koordinatnega sistema, glede na osi absolutnega koordinatnega sistema oz. glede na osi poljubnega koordinatnega sistema.

## 2.3 1. del naloge

Paket funkcij v okolju Matlab zna rokovati z različnimi objekti. Pod objekte štejemo točke, koordinatne sisteme in telesa, kot recimo kvadri in klini. Oznacbe teles so shranjene kot globalne spremenljivke z veliko začetnico: P# za točke, C# za koordinatne sisteme in B# za telesa. Oznaka "#" pomeni zaporedno številko telesa. Paket omogoča delo z do 9. objekti vsakega tipa.



Slika 2.5: Grafčni prikaz osnovne postavitve objekta oz. telesa

Vse funkcije (*Matlab datoteke s končnico .m*) za generiranje objektov postavijo objekt oz. njegov koordinatni sistem (**koordinatni sistem je postavljen v oglišče objekta**) v točko (0,0,0), osi koordinatnih sistemov objektov pa so usmerjene enako kot osi prikaza. To situacijo prikazuje slika 2.5, kjer so označene glavne značilke okolja.

Uporabite **programsko okolje Matlab**, je že nastavljeno tako, da ste postavljeni v ustrezni mapi. Odprite že pripravljeno predlogo *Spirala.m*, kamor vpisujete ustrezne ukaze, področje vpisa za študente pa je tudi nazorno označeno. Zapisane ukaze izvedete tako, da datoteko najprej shranite in v ukazni vrstici okolja Matlab potrdite ukaz:

>> **Spirala**

### 2.3.1 Seznam funkcij za uporabo v Matlabu

V nadaljevanju je podan seznam pomembnih ukazov iz paketa za prostorske transformacije (začetnica "robo..."). Opozoriti velja, da so imena objektov zmeraj pisana z **veliko začetnico** (primer: B1), razen, če poimenovanja ne postori uporabnik.

#### Inicializacijski ukazi

##### **roboinit**

Inicializacija okolja: odpre sliko in nastavi graf čne lastnosti okna za prikaz oz. simulacijo.

*Primer: roboinit*

#### Ukazi za ustvarjanje objektov

##### **ks**

Ukaz **ks**, ki nima parametrov, ustvari koordinatni sistem v točki (0,0,0) z osmi usmerjenimi enako kot so osi prikaza. Poleg koordinatnega sistema zapiše tudi tekstovno oznako v obliki C#. Če uporabnik ne poimenuje objekta, dobi objekt ime v obliki C#, vendar največ 9 objektov. V primeru poimenovanja objekta (**glej spodnji primer**) lahko uporabnik ustvari tudi več objektov.

*Primer: ks1 = ks*

##### **kvader** ( $d_x, d_y, d_z$ )

Ukaz **kvader**, ki ima vhodne parametre, ustvari kvader z izhodiščem v točki(0,0,0) z ustreznimi dimenzijami stranic. Dimenzije stranic kvadra so določene tako, da v smeri osi  $x$  dimenzijo določa vhodni parameter funkcije  $d_x$ , v smeri osi  $y$  dimenzijo določa vhodni parameter funkcije  $d_y$  in v smeri osi  $z$  dimenzijo določa vhodni parameter  $d_z$ . Poleg vizualnega prikaza kvadra ima objekt tudi pripadajoč relativni (lokalni) koordinatni sistem ter dodano tekstovno oznako. Če uporabnik ne poimenuje objekta, potem objekt dobi ime v obliki **B#**, vendar največ 9 objektov. V primeru poimenovanja objekta (**glej spodnji primer**) lahko uporabnik ustvari tudi več objektov.

*Primer: kv1 = kvader(2,3,1)*

**klin**( $d_x, d_y, d_z$ )

Ukaz **klin**, ki ima vhodne parametre, ustvari klin z izhodiščem v točki(0,0,0) z ustreznimi dimenzijami stranic. Dimenzije stranic klina so določene tako, da v smeri osi  $x$  dimenzijo določa vhodni parameter funkcije  $d_x$ , v smeri osi  $y$  dimenzijo določa vhodni parameter funkcije  $d_y$  in v smeri osi  $z$  dimenzijo določa vhodni parameter  $d_z$ . Poleg vizualnega prikaza klina ima objekt tudi pripadajoč relativni (lokalni) koordinatni sistem ter dodano tekstovno oznako. Če uporabnik ne poimenuje objekta, potem objekt dobi ime v obliki B#, vendar največ 9 objektov. V primeru poimenovanja objekta (**glej spodnji primer**) lahko uporabnik ustvari tudi več objektov.

*Primer:  $k11 = \text{klin}(4,2,7)$*

**Ukazi za transformacije objektov**

**roborotx**(OBJEKT, KOLIKO, KS, N)

Zarotira OBJEKT okoli osi  $x$  za KOLIKO radianov v N korakih (po želji) glede na koordinatni sistem KS:

KS='R' ali 'r': relativni ali lokalni koordinatni sistem,

KS='A' ali 'a': absolutni ali referenčni koordinatni sistem.

*Primer:  $\text{roborotx}(kv1, -\pi/20, 'A', 100)$*

**roboroty**(OBJEKT, KOLIKO, KS, N)

Zarotira OBJEKT okoli osi  $y$  za KOLIKO radianov v N korakih (po želji) glede na koordinatni sistem KS:

KS='R' ali 'r': relativni ali lokalni koordinatni sistem,

KS='A' ali 'a': absolutni ali referenčni koordinatni sistem.

*Primer:  $\text{roboroty}(kv1, \pi/4, 'R', 10)$*

**roborotz**(OBJEKT, KOLIKO, KS, N)

Zarotira OBJEKT okoli osi  $z$  za KOLIKO radianov v N korakih (po želji) glede na koordinatni sistem KS:

KS='R' ali 'r': relativni ali lokalni koordinatni sistem,

KS='A' ali 'a': absolutni ali referenčni koordinatni sistem.

*Primer:  $\text{roborotz}(B1, \pi/4, 'R', 1)$*

**robotrans(OBJEKT, KOLIKO, KS, N)**

Translira OBJEKT za vektor KOLIKO (vrstični  $3 \times 1$ ) v N korakih (po želji) glede na koordinatni sistem KS:

KS='R' ali 'r': relativni ali lokalni koordinatni sistem,

KS='A' ali 'a': absolutni ali referenčni koordinatni sistem.

Primer: *robotrans(kv1,[1,3,5],'A',1)*

**HM = hmks(OBJEKT)**

Zapiše homogeno transformacijsko matriko, ki predstavlja trenutno lego OBJEKTA v absolutnem oz. referenčnem koordinatnem sistemu.

Primer: *H1 = hmks(kv1)*

**robogentrans(OBJEKT, LEGA, KS)**

Premakne OBJEKT v LEGO, ki jo predstavlja homogena transformacijska matrika (ukaz hmks) glede na koordinatni sistem KS:

KS='R' ali 'r': relativni ali lokalni koordinatni sistem,

KS='A' ali 'a': absolutni ali referenčni koordinatni sistem.

Primer: *H1 = hmks(kv1)*

*robogentrans(kv2,H1,'A')*

**Vse navedene funkcije so lahko uporabljene v 1. delu naloge, pri čemer bi izpostavili uporabo kombinacijo funkcije *hmks()* in *robogentrans()*, ki vam v precejšnji meri skrajša čas gradnje spirale.**

**2.4 2. del naloge**

Vsak robotski krmilnik, za pravilno vodenje motorjev robota, potrebuje **geometrijski model robota**. Pogosti izraz je tudi **kinematični model robota** oz. celo **direktni kinematični model robota**. Za izdelavo kinematičnega modela robota je na voljo več metod s katerimi pa pridemo do istega rezultata. Ena izmed metod se imenuje metoda z **vektorskimi parametri**.

**Geometrijski model robotskega mehanizma opiše lego koordinatnega sistema na vrhu robota glede na bazni koordinatni sistem, in to v odvisnosti od trenutne vrednosti sklepnih spremenljivk (kot v sklepu, če je le-ta rotacijski oz. pozicija sklepa, če je le-ta translacijski) in dolžine segmentov med sklepi robotskega mehanizma.**

### 2.4.1 Geometrijski model robota z vektorskimi parametri

Za zapis direktnega geometrijskega modela uporabimo metodo z vektorskimi parametri, ki ima naslednje korake.

#### Postopek določanja vektorskih parametrov:

1. Mehanizem postavimo v začetno lego, kjer so vrednosti sklepnih koordinat enake nič ( $\vartheta_i = 0$ ,  $d_i = 0$ ,  $i = 1, 2, \dots, n$ ). Pri tem morajo biti osi sklepov vzporedne osem referenčnega koordinatnega sistema  $x_0, y_0, z_0$ .  
*Robot je že narisana v referenčni legi (Slika ??). V tej legi ima tudi realni robot vrednost notranjih spremenljivk enako 0.*
2. V središče sklepa  $i$  namestimo lokalne koordinatne sisteme  $x_i, y_i, z_i$ . Njihove osi so vzporedne osem referenčnega k.s.  $x_0, y_0, z_0$ .  
*Pri vaji so, na priloženi skici robota, izhodišča lokalnih koordinatnih sistemov označena s črno piko (Sliki ?? in ??).*
3. V vsako os mehanizma  $i = 1, 2, \dots, n$  postavimo enotski sklepni vektor  $\mathbf{e}_i$ . Njegova smer kaže v smeri ene osi koordinatnega sistema  $x_i, y_i, z_i$ . V smeri tega vektorja merimo pozitivno translacijsko koordinato  $d_i$ , okrog njega (obratna smer urinega kazalca) pa pozitivno rotacijsko koordinato  $\vartheta_i$ .
4. Med izhodišči lokalnih koordinatnih sistemov povlečemo segmentne vektorje  $\mathbf{b}_{i-1}$ . Segmentni vektor  $\mathbf{b}_n$  leži med izhodiščem koordinatnega sistema  $x_n, y_n, z_n$  in referenčno točko na vrhu mehanizma.

#### Homogene transformacijske matrike določimo s spodnjo predlogo:

$$\mathbf{H}_{i-1,i} = \begin{bmatrix} \mathbf{Rot}_{i-1,i} & d_i \mathbf{e}_i^{i-1} + \mathbf{b}_{i-1,i}^{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

V določenih primerih vpeljemo še dodatni koordinatni sistem v vrh robota oz. v referenčno točko prijemala. Označimo ga z  $x_{n+1}, y_{n+1}, z_{n+1}$ . Med koordinatnim sistemom  $x_n, y_n, z_n$  in  $x_{n+1}, y_{n+1}, z_{n+1}$  ni nobenih prostostnih stopenj, saj sta oba pripeta na isti segment, zato je transformacija konstantna.

*Če hočemo primerjati rezultat lege vrha dejanskega robota Epson PS3 z našim modelom, je vpeljava dodatnega k.s. oz. dodatne transformacije na vrh robota nujna, saj se drugače orientacija vrha robota v našem modelu ne ujema z orientacijo koordinatnega sistema na vrhu realnega robota.*

## 2.5 Koraki za izvedbo vaje

Izdelava vektorskega modela poteka v okolju Matlab. V ukazni vrstici okolja Matlab vpišite:

» **open Za\_studente\_PS3**

Za možnost **preverjanja rezultata** vašega direktnega kinematičnega robota za robot Epson PS3 s tistim, ki teče na samem robotskem krmilniku, je potrebno zagotoviti, da je pognan ustrezen **Epson RC+ vmesnik**. **Običajno to okolje teče na drugem računalniku, zato za njegov zagon povprašajte asistenta.**

Z upoštevanjem navodil za postavljanje vektorskih parametrov na priložen list vrišite ustrezno koordinatne sisteme, zapišite tabelo z vektorskimi parametri in na podlagi le te def nirajte direktni geometrijski model robota.

V nadaljevanju v predlogi okolja Matlab zapišite ustrezne homogene transformacijske matrice  $\mathbf{H}_{01}$ ,  $\mathbf{H}_{12}$ , ...  $\mathbf{H}_{xy}$  in  $\mathbf{T}$ . Def nirajte jih z že pripravljenimi funkcijami *H\_RotX\_Trans*, *H\_RotY\_Trans* in *H\_RotZ\_Trans*.

### Primer definiranja matrice $\mathbf{H}_{01}$ :

Koordinatni sistem v sklepu 1 je rotiran po osi  $z$  za  $\phi$  in transliran za  $x$ ,  $y$  in  $z$  glede na prejšnji koordinatni sistem. Funkcijo, s katero zapišemo lego koordinatnega sistema 1 glede na prejšnji koordinatni sistem 0, zapišemo:

$$\mathbf{H}_{01} = \mathbf{H\_RotZ\_Trans}(\phi, x, y, z);$$

Po def niranju vseh  $\mathbf{H}$  matrik z ustreznim zaporedjem množenja izračunajte matriko  $\mathbf{T}$ , ki predstavlja lego vrha robota (zadnji koordinatni sistem) glede na referenčni koordinatni sistem.

V nadaljevanju je prikazana predloga *Za\_studente\_PS3.m* za pisanje vrstic programa za izračun direktnega geometrijskega modela za robot Epson PS3. Vpisuje se samo vrstice, ki so označene s *% STUDENT*. Vhodni parametri funkcije so:  $h_1, h_2, h_3, l_1, l_2, l_3, th_1, th_2, th_3, th_4, th_5$  in  $th_6$ . Vse uporabite v kombinaciji s funkcijami za zapis homogenih transformacijskih matrik.

Ko zapišemo vse vrstice, zaženemo uporabniško okno za preverjanje rezultatov. V ukazno vrstico programskega okolja Matlab vpišemo

» **RezultatiGUI**

in odpre se okno s slike 2.6.

Objekti v uporabniškem vmesniku:

**Sklepne sprem.** → Vpisujemo vrednosti sklepnih spremenljivk.

**Matrika T** → Transformacijska matrika vrha robota glede na ref. k.s.



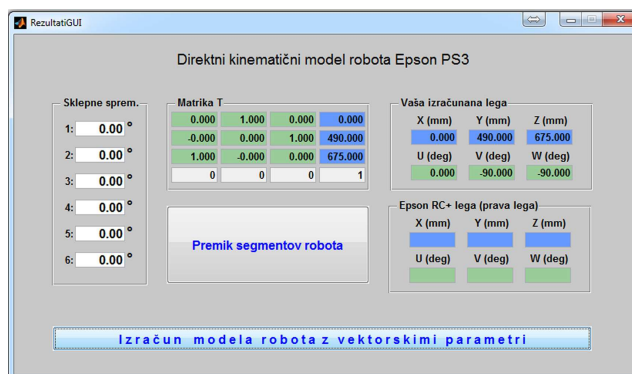
```

% *****
% *** PREDLOGA ZA PISANJE PROGRAMA ***
% *****
% Programske vrstice vpisujete le v podrocja, ki so oznacena s STUDENT!
% *****
% function Za_studente(h1,h2,h3,l1,l2,th1,th2,th3,th4,th5,th6)

% th1 do th6 ... koti v sklepih v radianih
% Definiranje homogenih transformacijskih matrik med sklepi
H01 = % STUDENT
H12 = % STUDENT
H23 = % STUDENT
H34 = % STUDENT
H45 = % STUDENT
H56 = % STUDENT
H67 = % STUDENT

% Multiplikacija homogenih transformacijskih matrik za izracun
% geometrijskega modela robota
T = % STUDENT

% Izpis rezultata
Izpis_rezultatov(T);
% *****
    
```



Slika 2.6: Uporabniški vmesnik za preverjanje rezultatov

**Vaša izračunana lega** → Pozicija in orientacija vrha glede na ref. k.s.

**Epson RC+ lega (prava lega)** → Lega, ki jo vrača originalna programska oprema robota.

**Premik segmentov robota** → Premaknemo robot Epson PS3 v lego, ki jo določajo vpisane kotne spremenljivke!

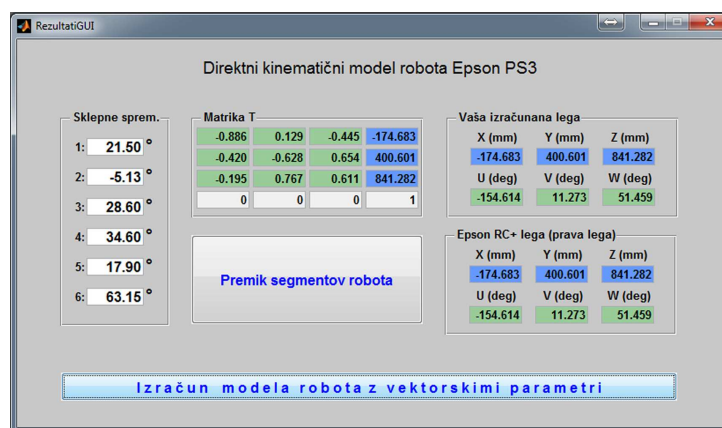
**Izračun modela robota z vektorskimi parametri** → Izračunamo kinematični model s svojimi parametri!

## 2.6 Rezultati za preverjanje kinematičnega modela za robot Epson PS3

Robot Epson PS3 je antropomorfn robot s šestimi rotacijskimi prostostnimi stopnjami, predstavljen na sliki 2.2. Na sliki je robot izrisan v legi, ko so vrednosti notranjih spremenljivk enake 0. Narisane so tudi pozitivne smeri rotacije in translacije na robotu.

Pravilnost direktnega kinematičnega modela preverite tako, da vpišete ustrezne vrednosti notranjih spremenljivk (Slika 2.2) v Matlab uporabniški vmesnik z imenom **RezultatiGUI**. S klikom na gumb *Izračun modela robota z vektorskimi parametri* izračunate polji "Matrika T" in "Vaša izračunana lega". Da lahko primerjate rezultate z dejansko lego robota, je potrebno klikniti še gumb *Premik segmentov robota*, seveda ste morali predhodno z asistentom urediti zagon **Epson RC+** vmesnika. S klikom pošljete robotskemu krmilniku željene kote v sklepih, robot se vanje dejansko premakne in nato v **RezultatiGUI** uporabniški vmesnik vrne lego robota izračunano s kinematičnim modelom robota v krmilniku. Če ste model pravilno sestavili, se morajo vrednosti v poljih *Vaša izračunana lega* in *Epson RC+ lega (prava lega)* ujemati.

V primeru, da je robot zaseden oz. obstaja kakšen drug razlog za njegovo neuporabo, rezultate preverite s pomočjo slike 2.7.



Slika 2.7: Vrednosti notranjih spremenljivk in rezultati za robot Epson PS3

**Pazimo, da v delovnem prostoru robota ni osebe ali predmeta!**

## Poglavje 3

# VODENJE DVOSEGMENTNEGA MANIPULATORJA NA OSNOVI ROBOTA MOTOMAN MH5

### 3.1 Cilj vaje

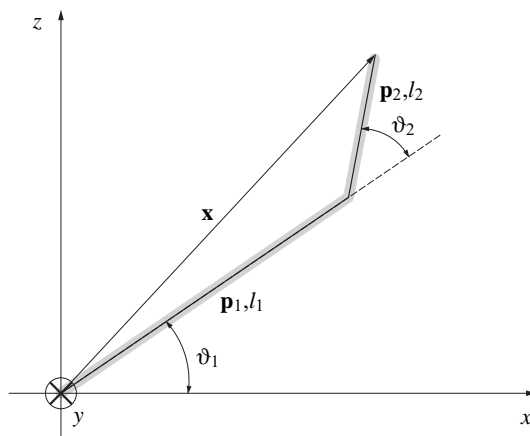
Pri tej vaji se boste spoznali z dvosegmentnim manipulatorjem, ki je osnovni sestav za več tipov robotov. Določili boste direktno in inverzno kinematiko dvosegmentnega manipulatorja na osnovi robota Motoman MH5. Načrtali boste tudi vodenje v notranjih koordinatah, pri čemer bo referenca podana v notranjih in zunanjih koordinatah. Delovanje sistema boste preverili v simulacijskem okolju robota Motoman MH5.

### 3.2 Dvosegmentni manipulator

Pri tej vaji bomo obravnavali ravninski dvosegmentni robotski mehanizem z dvema rotacijskima sklepoma (slika 3.1). Dvosegmentni mehanizem (2DOF mehanizem) je pomemben del tako SCARA kakor tudi antropomorfnega robota, kar nam bo omogočilo spoznati številne značilnosti gibanja robotskih mehanizmov. V kinematiki razlikujemo direktno in inverzno kinematiko. Direktna kinematika predstavlja računanje pozicije končne točke manipulatorja, pri čemer so znane notranje koordinate manipulatorja. Inverzna kinematika pa določa notranje koordinate iz znanega položaja vrha manipulatorja. Direktna kinematika predstavlja enostavnejši problem, saj znanim notranjim koordinatam manipulatorja pripada ena sama rešitev za položaj vrha. Reševanje inverzne kinematike je močno odvisno od same strukture manipulatorja. Za nekatere vrste manipulatorjev problema inverzne kinematike ne moremo rešiti analitično.

Za prvi sklep je os rotacije os  $y$ , ki kaže v papir. Definirajmo vektor  $\mathbf{p}_1$  v smeri prvega segmenta. Tako imamo

$$\mathbf{p}_1 = l_1 \begin{bmatrix} \cos \vartheta_1 \\ \sin \vartheta_1 \end{bmatrix}. \quad (3.1)$$



Slika 3.1: Dvosegmentni ravninski robotski manipulator

Vzdolž drugega segmenta poteka vektor  $\mathbf{p}_2$ . Njegove komponente odčitamo s slike 3.1

$$\mathbf{p}_2 = l_2 \begin{bmatrix} \cos(\vartheta_1 + \vartheta_2) \\ \sin(\vartheta_1 + \vartheta_2) \end{bmatrix}. \quad (3.2)$$

Definirajmo še vektor  $\mathbf{x}$ , ki naj poteka iz koordinatnega izhodišča do vrha robota

$$\mathbf{x} = \mathbf{p}_1 + \mathbf{p}_2. \quad (3.3)$$

Vektor  $\mathbf{x}$  predstavlja položaj vrha manipulatorja

$$\mathbf{x} = \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} l_1 \cos \vartheta_1 + l_2 \cos(\vartheta_1 + \vartheta_2) \\ l_1 \sin \vartheta_1 + l_2 \sin(\vartheta_1 + \vartheta_2) \end{bmatrix}. \quad (3.4)$$

Če definiramo vektor kotov v sklepih

$$\mathbf{q} = [\vartheta_1 \quad \vartheta_2]^T, \quad (3.5)$$

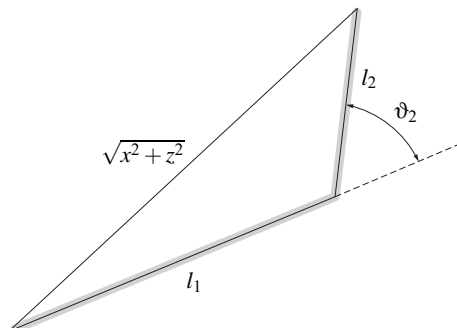
lahko enačbo (3.4) krajše zapišemo

$$\mathbf{x} = \mathbf{k}(\mathbf{q}), \quad (3.6)$$

kjer  $\mathbf{k}(\cdot)$  predstavlja enačbe direktne kinematike.

Pri reševanju inverzne kinematike računamo kote v sklepih iz znanega položaja vrha robota. Slika 3.2 prikazuje samo tiste oznake dvosegmentnega mehanizma, ki so pomembne za izračun kota  $\vartheta_2$ . Uporabimo kosinusno pravilo

$$x^2 + z^2 = l_1^2 + l_2^2 - 2l_1l_2 \cos(180^\circ - \vartheta_2). \quad (3.7)$$



Slika 3.2: Izračun kota  $\vartheta_2$

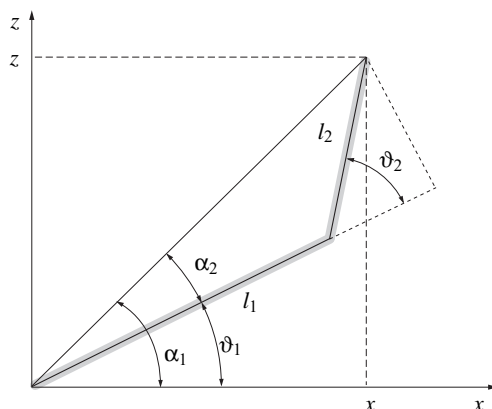
Kot v drugem sklepu dvosegmentnega manipulatorja izračunamo kot obratno trigonometrično funkcijo

$$\vartheta_2 = \arccos \frac{x^2 + z^2 - l_1^2 - l_2^2}{2l_1l_2}. \quad (3.8)$$

Kot v prvem sklepu določimo z uporabo slike 3.3. Izračunamo ga kot razliko

$$\vartheta_1 = \alpha_1 - \alpha_2.$$

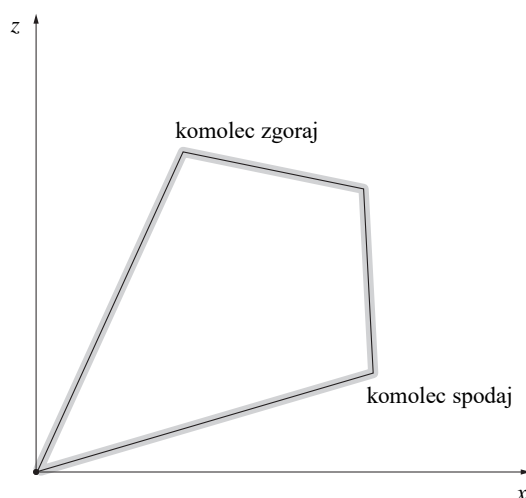
Kot  $\alpha_1$  izračunamo iz pravokotnega trikotnika, ki ga sestavljata horizontalna koordinata vrha robota  $x$  in vertikalna koordinata  $z$ . Kot  $\alpha_2$  pa iz trikotnika, kjer trikotnik s slike 3.2 podaljšamo v pravokotni trikotnik, kot kaže slika 3.3. Tudi sedaj uporabimo obratne trigonometrične funkcije



Slika 3.3: Izračun kota  $\vartheta_1$

$$\vartheta_1 = \arctan\left(\frac{z}{x}\right) - \arctan\left(\frac{l_2 \sin \vartheta_2}{l_1 + l_2 \cos \vartheta_2}\right). \quad (3.9)$$

Glede na izbiro  $\vartheta_2$  imamo dve rešitvi. Pri eni imamo “komolec zgoraj” in pri drugi “komolec spodaj”, kot kaže slika 3.4. Izrojeno rešitev predstavlja



Slika 3.4: Dve rešitvi inverzne kinematike

položaj vrha robota  $x = z = 0$  pri  $l_1 = l_2$ , saj je tedaj  $\arctan\left(\frac{y}{z}\right)$  nedefiniran. Če je kot  $\vartheta_2 = 180^\circ$ , moremo doseči izhodiščno točko ob poljubnem kotu  $\vartheta_1$ . Če je točka  $(x, z)$  izven delovnega prostora manipulatorja, problema inverzne kinematike seveda ne moremo rešiti.

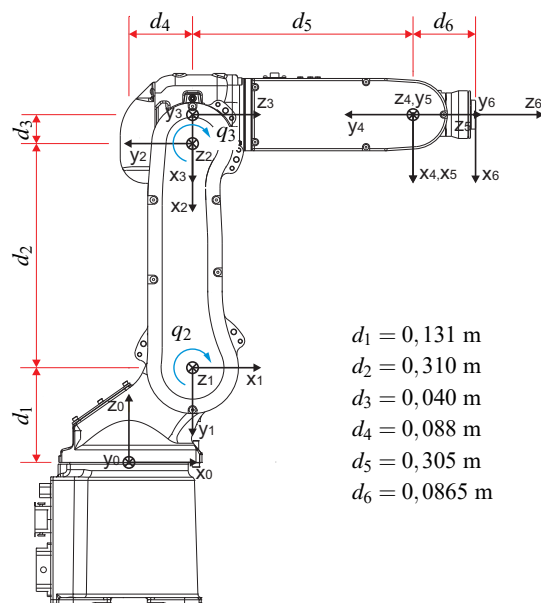
### 3.3 Dvosegmetni manipulator na osnovi robota Motoman MH5

Na sliki 3.5 je prikazan antropomorfn robotski manipulator Motoman MH5 z označenimi koordinatnimi sistemi posameznih sklepov. Označene so tudi kinematične veličine manipulatorja.

Dvosegmetni manipulator lahko pri konfiguraciji robota MH5 umestimo tako, da postavimo prvi sklep manipulatorja v drugi sklep  $q_2$  robota MH5, drugi segment pa povezuje sklep  $q_3$  z vrhom robota (slika 3.6). Pri tem sta dolžini segmentov 2DOF manipulatorja

$$l_1 = d_2 \quad (3.10)$$

$$l_2 = \sqrt{d_3^2 + (d_5 + d_6)^2}. \quad (3.11)$$



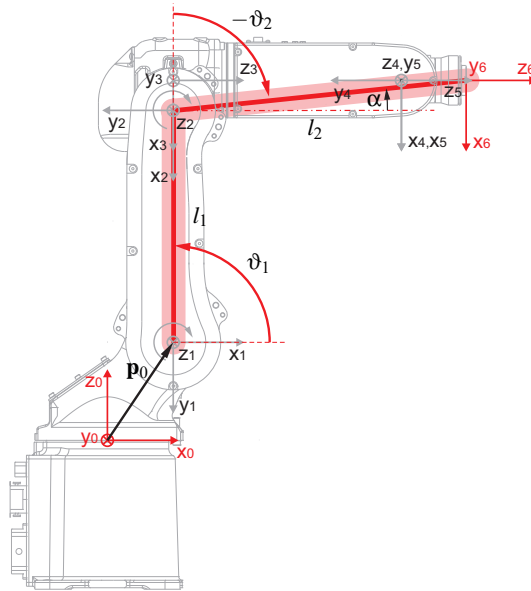
Slika 3.5. Motoman MH5 v začetni legi z označenimi koordinatnimi sistemi, kinematičnimi veličinami in pozitivnima smerema za sklepa  $q_2$  in  $q_3$

Direktna kinematika opisuje lego koordinatnega sistema na vrhu robota  $x_6$ - $y_6$ - $z_6$  glede na bazni koordinatni sistem  $x_0$ - $y_0$ - $z_0$ . Da velja direktna kinematika, predstavljena z enačbo (3.4), je potrebno upoštevati še premik središča 2DOF mehanizma v drugi sklep robota MH5  $p_0$  ter ustrezno relacijo med koti 2DOF manipulatorja in koti v sklepih robota MH5, kot to prikazuje slika 3.7. Pri tem je potrebno upoštevati tudi kot  $\alpha$ , ki opisuje premik drugega segmenta 2DOF manipulatorja zaradi zamaknjenosti vrha robota in tretjega sklepa za razdaljo  $d_3$ .

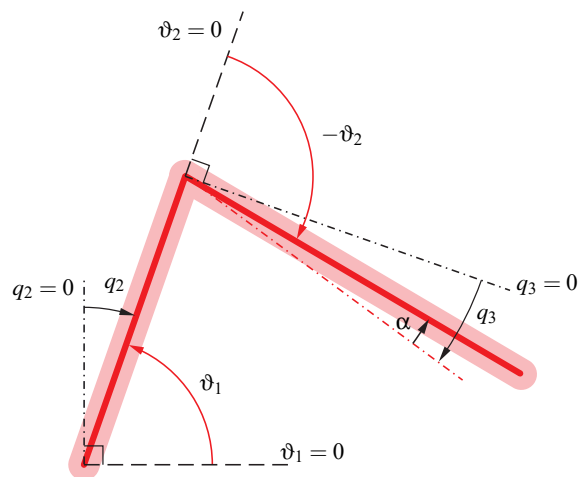
### 3.4 Vodenje robota

Problem vodenja robotov lahko predstavimo kot izračun sil oziroma navorov, ki jih morajo zagotoviti motorji robota za uspešno izvedbo naloge, pri čemer morajo zadovoljiti pogojem delovanja med prehodnim pojavom in v ustaljenem stanju. Naloga lahko pomeni izvajanje gibov v praznem prostoru ali v dotiku z okolico, kjer je potrebna regulacija sile dotika. Pri tej vaji se bomo posvetili načrtovanju vodenja robota, ki ni v dotiku z okolico.

Vodenje robota običajno izvajamo v zunanjih koordinatah, torej v prostoru naloge. To pomeni, da nas neposredno zanima predvsem lega vrha robota, le redkokdaj pa tudi položaji v sklepih. Pri tem je potrebno upoštevati, da neposredno vedno reguliramo le položaje v sklepih, samo posredno pa lego vrha



Slika 3.6: Dvosegmentni manipulator umeščen na robota Motoman MH5

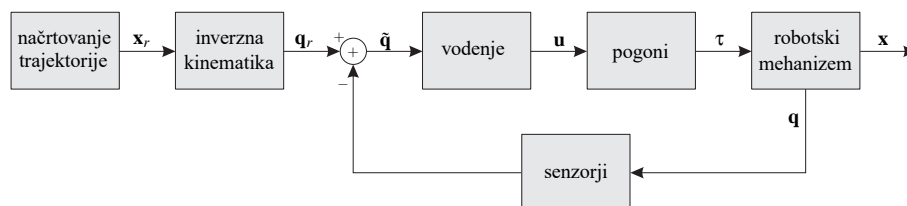


Slika 3.7. Določitev kotov dvosegmentnega manipulatorja  $\vartheta_1$  in  $\vartheta_2$  na osnovi kotov robota Motoman MH5  $q_2$  in  $q_3$

robota, ki je določena s kinematičnim modelom robotskega mehanizma ob danih vrednostih položajev v sklepkih.



Slika 3.8 prikazuje splošno regulacijsko shemo. Vhod v sistem vodenja je zelena lega robota, ki jo označimo s spremenljivko  $\mathbf{x}_r$ . Vektor  $\mathbf{x}_r$  je v splošnem vektor šestih spremenljivk, od katerih tri določajo položaj vrha robota, tri pa orientacijo vrha robota, torej  $\mathbf{x} = [x \ y \ z \ \varphi \ \vartheta \ \psi]^T$ .



Slika 3.8: Posplošena shema vodenja robota.

Z izvedbo algoritma inverzne kinematike izračunamo položaje sklepov  $\mathbf{q}_r$ , ki ustrezajo zeleni legi vrha robota. Spremenljivka  $q_r$  predstavlja zasuk  $\vartheta$ , če je sklep rotacijski oziroma razdaljo  $d$ , če je sklep translacijski. Zelene vrednosti položajev sklepov se v sistemu vodenja primerjajo z dejanskimi položaji robota in na podlagi napake  $\tilde{\mathbf{q}}$  se izračuna izhod iz regulatorja  $\mathbf{u}$ . Izhod  $\mathbf{u}$  pretvorimo v analogno obliko, ojačimo in vodimo na pogone robota. Motorji robota zagotovijo sile oziroma navore potrebne za gibanje robotskega mehanizma. Gibanje robota merimo z različni senzorji, kot so na primer absolutni/relativni inkrementalni dajalniki pozicije.

### 3.4.1 PD regulacija položaja

Najenostavnejše vodenje robotov temelji na regulatorjih, katerih zanka je zaključena preko posamezne prostostne stopnje. Najprej bomo analizirali enostavni proporcionalno-diferencirni (PD) regulacijski sistem. Osnovna zgradba takšnega sistema je prikazana na sliki 3.9. Vodenje temelji na izračunu napake položaja in določitvi takšnih regulirnih veličin, ki bodo omogočile zmanjšanje oziroma odpravo napake. Pozicijsko napako zmanjšujemo v vsakem sklepu posebej, kar pomeni, da moramo zgraditi toliko regulatorjev, kolikor ima robot prostostnih stopenj.

Referenčni položaj  $\mathbf{q}_r$  primerjamo z dejanskim položajem posameznih sklepov robota  $\mathbf{q}$

$$\tilde{\mathbf{q}} = \mathbf{q}_r - \mathbf{q}. \quad (3.12)$$

Napako pozicije  $\tilde{\mathbf{q}}$  ojačimo s proporcionalnim položajnim ojačenjem  $\mathbf{K}_p$ . Ker ima robot več prostostnih stopenj, je napaka  $\tilde{\mathbf{q}}$  izražena v obliki vektorja,  $\mathbf{K}_p$  pa je diagonalna matrika ojačenj regulatorjev posameznih prostostnih stopenj. Tako izračunana regulirna veličina povzroči premik robota v smeri zmanjšanja pozicijske napake. Ker pa je aktuacija pogonov robota proporcionalna napaki,

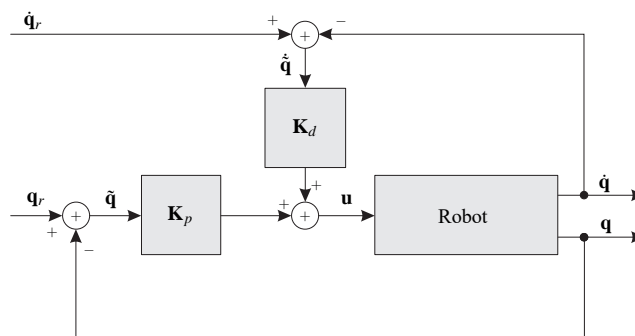
se lahko zgodi, da se robot ne ustavi v želeni legi, ampak jo preniha. V robotiki prenihajev ne smemo dopustiti, saj lahko ob prenihaju pride do trka z objekti v bližini robota. Za zagotavljanje varnega in stabilnega delovanja sistema zato vpeljemo hitrostno povratno zanko z negativnim predznakom. Hitrostna povratna zanka vnaša dušenje v sistem in je določena z napako med željenimi  $\dot{\mathbf{q}}_r$  in trenutnimi hitrostmi gibanja sklepov  $\dot{\mathbf{q}}$ , pomnoženimi s hitrošnim ojačenjem  $\mathbf{K}_d$  (diagonalna matrika hitrošnih ojačenj).

$$\dot{\tilde{\mathbf{q}}} = \dot{\mathbf{q}}_r - \dot{\mathbf{q}}. \quad (3.13)$$

Celoten regulacijski zakon lahko sedaj zapišemo

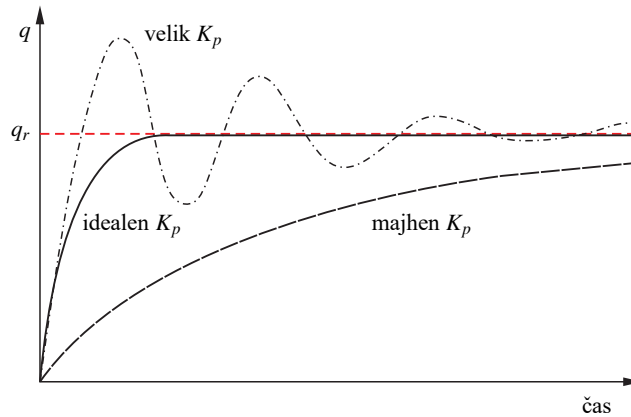
$$\mathbf{u} = \mathbf{K}_p(\mathbf{q}_r - \mathbf{q}) + \mathbf{K}_d(\dot{\mathbf{q}}_r - \dot{\mathbf{q}}), \quad (3.14)$$

pri čemer  $\mathbf{u}$  predstavlja regulirne veličine, to pa so posplošene sile v sklepih robota, ki jih morajo zagotoviti motorji. Iz enačbe (3.14) lahko razberemo, da pri višji hitrosti gibanja sklepa hitrostna zanka zmanjšuje silo v sklepu, s čimer duši sistem in zagotavlja stabilno delovanje.



Slika 3.9: PD regulacija položaja.

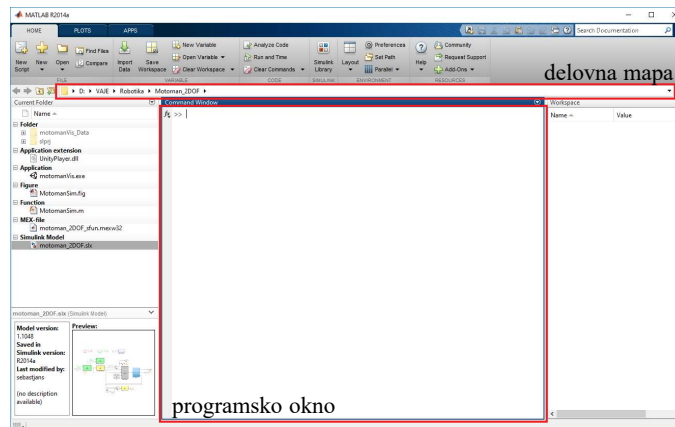
Pri PD regulaciji položaja se sinteza regulacijskega sistema prevede na določanje matrik ojačenj  $\mathbf{K}_p$  in  $\mathbf{K}_d$ . Za hitre odzive morajo biti ojačenja  $\mathbf{K}_p$  velika, s pravilno izbiro ojačenj  $\mathbf{K}_d$  pa zagotovimo kritično dušenje robotskega sistema. Kritično dušenje zagotavlja hitre odzive, vendar brez prenihajev, ki v robotiki niso dovoljeni. Takšen regulator moramo zgraditi za vsak sklep posebej. Delovanje posameznega regulatorja pa je povsem neodvisno od delovanja regulatorjev, ki pripadajo ostalim sklepom robotskega mehanizma. Primer odzivov pri različno nastavljenih ojačenjih  $K_p$  so prikazani na sliki 3.10.



Slika 3.10: Odziv sistema pri različnih ojačenjih  $K_p$

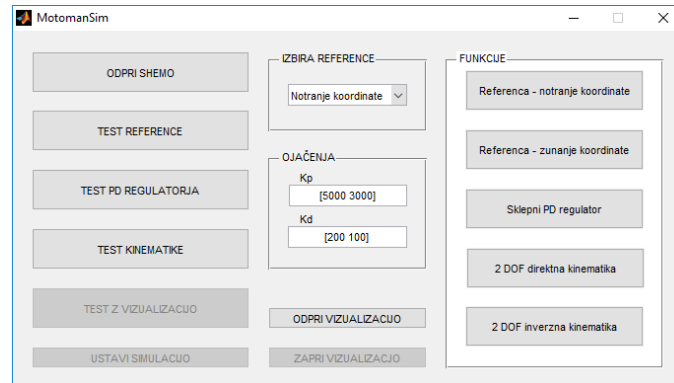
### 3.5 Koraki za izvedbo vaje

- 1 Odprite programsko okolje Matlab s klikom na bližnjico **Robotika 2DOF** na namizju (če še ni odprto).
- 2 V programskem okolju Matlab (slika 3.11) preverite, če je izbrana prava delovna mapa, saj v nasprotnem primeru ne boste mogli dostopati do ustreznih datotek. Biti mora `... \Robotika \Motoman_2DOF`.



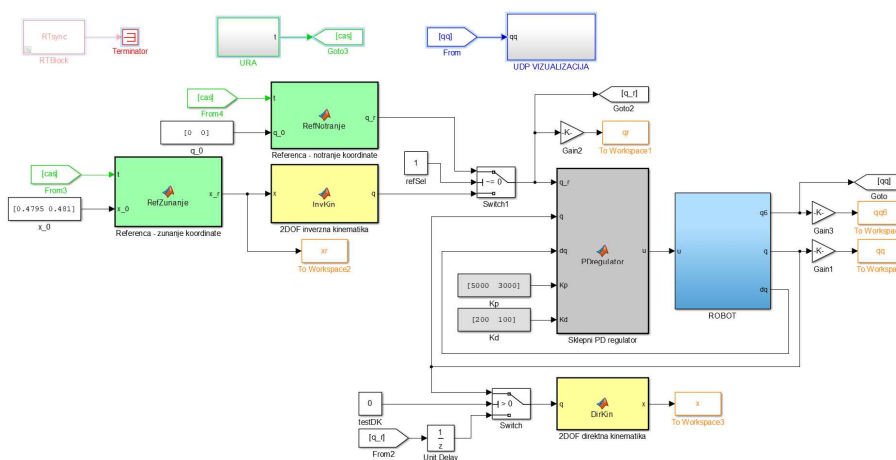
Slika 3.11: Grafčni vmesnik okolja Matlab

- 3 V programsko vrstico vpišete ukaz  
`>>MotomanSim`



Slika 3.12: Graf čni uporabniški vmesnik MotomanSim

- 4 Odpre se vam graf čni uporabniški vmesnik (slika 3.12), kjer kliknete na gumb **ODPRI SHEMO**, da se vam odpre Simulink shema s simulacijsko kodo (slika 3.13). Simulacijski bloki soupadajo s poslošeno shemo vodenja robotov na sliki 3.8.



Slika 3.13: Simulink simulacijska shema

- 5 Preverite, da imate v polju **IZBIRA REFERENCE** izbrano možnost **Notranje koordinate**.
- 6 Z gumbom **Referenca - notranje koordinate** odprete predlogo Matlab funkcije, kjer določite referenco v notranjih koordinatah za premik kota

$q_2$  za  $20^\circ$  okoli osi  $-z$  ter za premik kota  $q_3$  za  $20^\circ$  okoli osi  $z$  glede na začetno lego.

```
function q_r= RefNotranje(t,q_0)
Funkcija RefNotranje ima vhode čas t in začetni položaj robota (sklepa
 $q_2$  in  $q_3$ ) v radianih  $q_0$ . Kot izhod je potrebno določiti ustrezne referenčne
kote  $q_r$ .
```

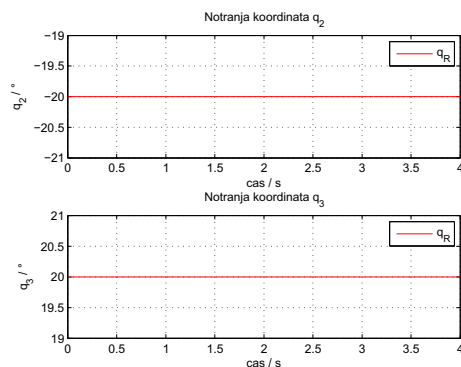
V predlogah Matlab funkcij študentje pišete kodo med značkama

```
%%%%%%%% STUDENT %%%%%%%%%
. . .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

V večini primerov so spremenljivke že definirane (njihova vrednost je postavljena na 0), tako da je potrebno samo zamenjati obstoječo vrednost z ustreznim izrazom.

Pomembni je tudi **branje komentarjev** v funkcijah, saj vsebujejo razlago funkcij, namige za naloge ter primere uporabe sintakse!

- 7 Ko definirate ustrezno referenco, kliknete na gumb **TEST REFERENCE**. Shema se najprej prevede in izvede, nato se odprejo graf z definiranimi referencami. Graf morajo biti podobni grafom na sliki 3.14.



Slika 3.14: Referenca v notranjih koordinatah - prvi del

- 8 Nato je potrebno definirati PD regulator, kot je predstavljen v prejšnjem poglavju. S klikom na gumb **Sklepni PD regulator** se vam odpre funkcija `PDregulator`, kjer zapiše PD regulator za vodenje  $q_2$  in  $q_3$ .

```
function u = PDregulator(q_r, q, dq, Kp, Kd)
```

Funkcija PDregulator ima vhode željen položaj sklepov  $q_r$  (sklepa  $q_2$  in  $q_3$ ), trenutni položaj robota (sklepa  $q_2$  in  $q_3$ ) v radianih  $q$  in trenutno hitrost sklepov  $dq$ . Ojačanji  $K_p$  in  $K_d$  sta tudi vhoda v funkcijo, nastavlja se pa jih lahko preko grafičnega vmesnika. Izhod funkcije  $u$  je regulirna veličina.

Sklepni regulator definira vodenje vsakega sklepa posebej. Proporcionalno-diferencirni regulator vsebuje dva člena. Proporcionalni del ojači napako pozicije medtem ko diferencirni del poskrbi za dušenje sistema na osnovi napake hitrosti.

Napako pozicije  $\tilde{q}$  se določi kot razliko med željeno in dejansko pozicijo sklepa

$$\tilde{q} = q_r - q. \quad (3.15)$$

To napako se potem ojači z ojačanjem  $K_p$

$$u = K_p \cdot \tilde{q}. \quad (3.16)$$

Izhod regulatorja  $u$  si lahko predstavljamo kot hitrost premikanja robota. Poglejmo si primer na sliki 3.15. Če je napaka med dejansko in željeno pozicijo pozitivna, je tudi hitrost premikanja pozitivna. Večja kot je napaka, večja bo hitrost premikanja sklepa. Če je sklep naredil prenehaj preko željene pozicije, potem je napaka negativna, posledično je tudi hitrost negativna. V tem primeru se bo segment premikal nazaj proti željeni poziciji. V primeru, ko pa je sklep v željeni poziciji, je dejanska pozicija enaka željeni. Takrat je napaka enaka 0, zato je tudi hitrost premikanja enaka 0.

Diferencirni del regulatorja poskrbi za ustrezno dušenje sistema. Napako hitrosti  $\dot{\tilde{q}}$  se določi kot razliko med željeno in dejansko hitrostjo

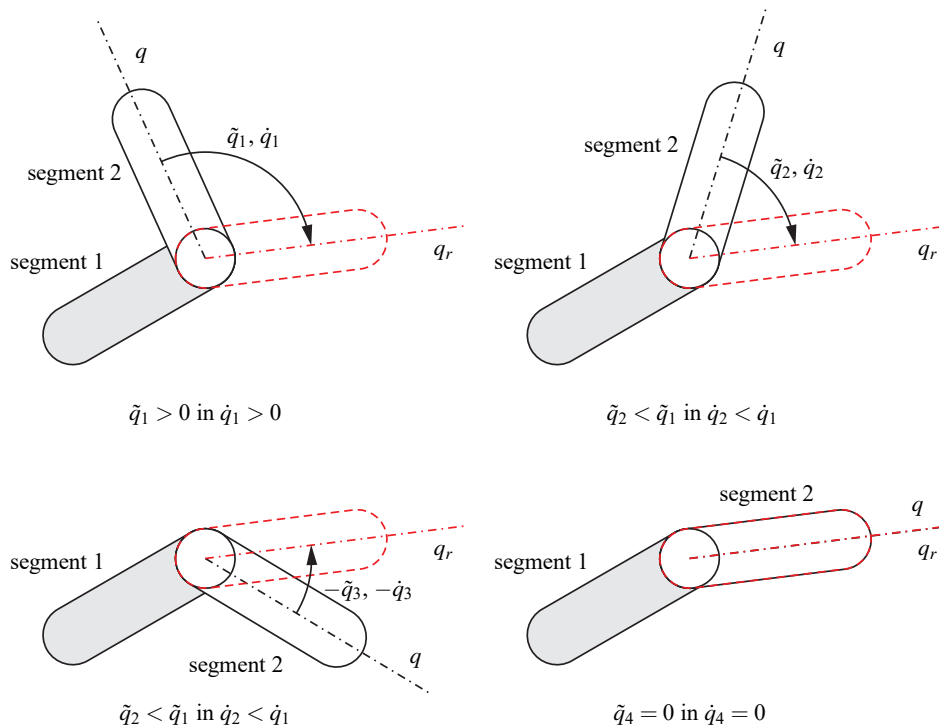
$$\dot{\tilde{q}} = \dot{q}_r - \dot{q}. \quad (3.17)$$

To napako se potem ojači z ojačenjem  $K_d$ , celoten regulator pa se zapiše kot

$$u = K_p \cdot \tilde{q} + K_d \cdot \dot{\tilde{q}}. \quad (3.18)$$

V robotskih sistemih se velikokrat želi, da se robot postavi v željeno lego. V tem primeru je željena hitrost enaka  $\dot{q}_r = 0$ . Regulator se potem zapiše kot

$$u = K_p(q_r - q) - K_d \cdot \dot{q}. \quad (3.19)$$



Slika 3.15: Vpliv člena  $K_p$  na gibanje segmenta

- 9 Preverite delovanje PD regulatorja s klikom na gumb **TEST PD REGULATORJA**. Nastavljene vrednosti ojačenj so

$$K_p = \begin{bmatrix} 5000 & 3000 \end{bmatrix},$$

$$K_d = \begin{bmatrix} 200 & 100 \end{bmatrix}.$$

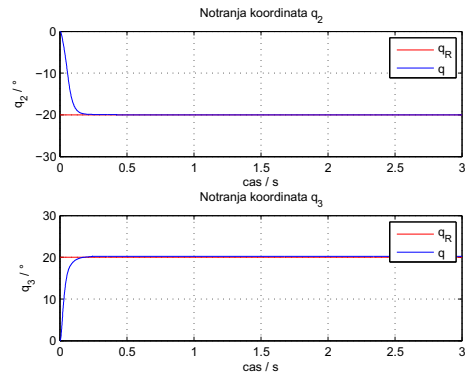
Dobljen graf odziva sistema bi moral biti podoben kot na sliki 3.16.

- 10 Testirajte delovanje PD regulatorja pri različnih vrednosti  $K_p$ :

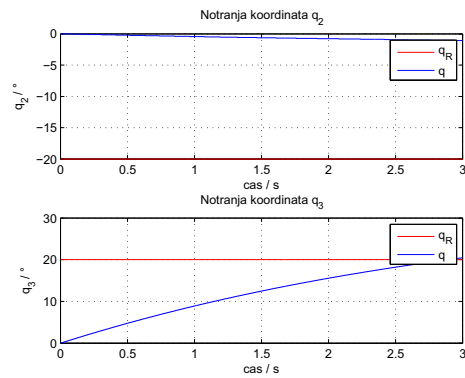
- majhen  $K_p$ :  $K_{p,1} = 0,01 \cdot K_p$ ,
- velik  $K_p$ :  $K_{p,2} = 100 \cdot K_p$ .

Odzivi sistema so predstavljeni z graf na sliki 3.17 za regulator z majhnim  $K_p$  (potrebno je veliko časa, da sistem doseže željeno lego oz. jo sploh ne doseže) in na sliki 3.18 za regulator z velikim  $K_p$  (nestabilen sistem).

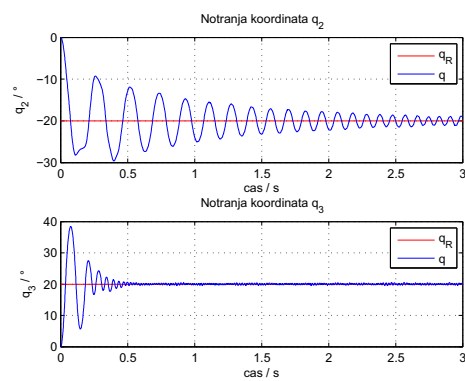
- 11 Določite referenčne vrednosti za kota  $q_2$  in  $q_3$  tako, da se bo kot  $q_2$  spreminjal po sinusu, kot  $q_3$  pa po kosinusu okoli začetne lege. Amplituda



Slika 3.16: Odziv PD regulatorja,  $K_p = [5000, 3000]$



Slika 3.17: Odziv PD regulatorja,  $K_p = [50, 30]$



Slika 3.18: Odziv PD regulatorja,  $K_p = [500000, 300000]$



gibanja naj bo  $A = 30^\circ$ , frekvenca pa  $f = 0,3 \text{ rad/s}$ . S klikom na gumb **Referenca - notranje koordinate** odprete ustrezno predlogo Matlab funkcije (že napisano referenco iz točke 6 lahko zakomentirate: pred ustrezne vrstice postavite znak za procent %).

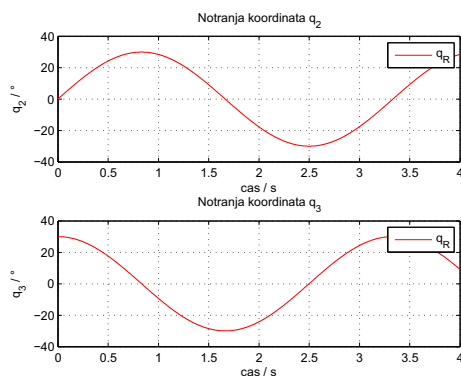
Položaj zapisan s kotno funkcijo ima obliko

$$q = A \sin(\omega t),$$

kjer je  $A$  amplituda,  $\omega$  kotna hitrost in  $t$  čas. Kotno hitrost lahko zapišemo kot funkcijo frekvence  $f$

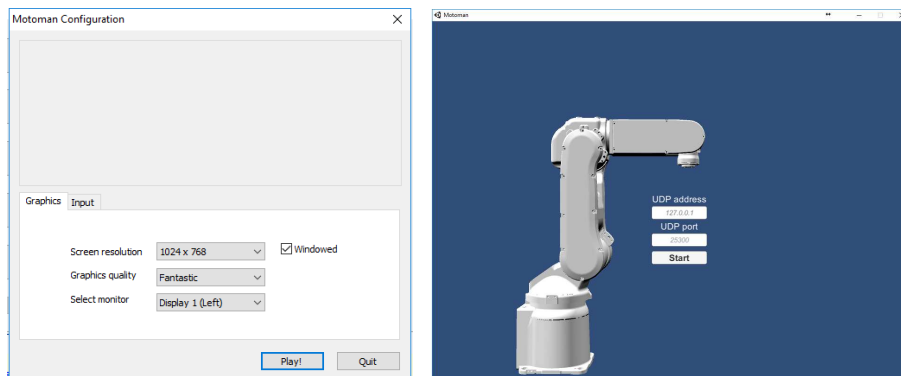
$$\omega = 2\pi f.$$

- 12 S klikom na gumb **TEST REFERENCE** testirajte definirano referenco, ki mora biti podobna prikazani na sliki 3.19

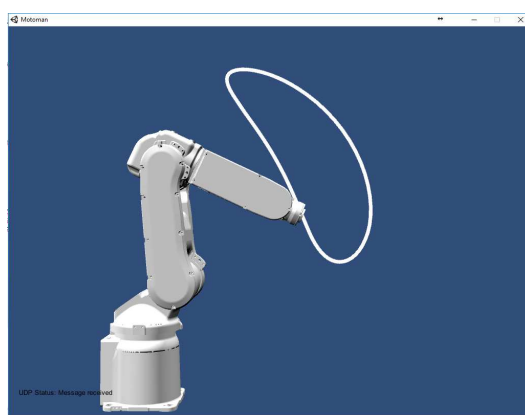


Slika 3.19: Referenca za gibanje po sinusu/kosinusu

- 13 V nadaljevanju boste testirali gibanje robota MH5 z vizualizacijo. Odprete jo s klikom na gumb **ODPRI VIZUALIZACIJO**. Odpre se vam okno *Motoman Configuration* (slika 3.20 levo), kjer izberete ustrezno resolucijo (lahko pustite prednastavljeno vrednost), obkljukate opcijo *Windowed* ter kliknete na gumb **Play!**. Odpre se okno programa Unity (slika 3.20 desno), kjer nato kliknete na gumb **Start**.
- 14 Simulacijo poženete z gumbom **TEST Z VIZUALIZACIJO**. Vizualni model robota se mora začeti gibati, kar nakazuje bela črta, ki opisuje gibanje vrha robota. Trajektorija gibanja vrha je predstavljena na sliki 3.21. Če je potrebno, ustrezno popravite ojačenja, da bo gibanje stabilno (gladka trajektorija).
- 15 Simulacija se bo izvajala dokler je ne ustavite s klikom na gumb **USTAVI SIMULACIJO**. Vizualizacijo zaprete z gumbom **ZAPRI VIZUALIZACIJO**.



Slika 3.20: Okni pri odpiranju vizualizacije



Slika 3.21. Gibanje vrha robota MH5 pri definirani referenci v notranjih koordinatah

- 16 V nadaljevanju boste realizirali kroženje vrha robota okoli začetne lege v  $x$ - $z$  ravnini. Če želimo načrtati gibanje vrha, potrebujemo definirane referenčne vrednosti v zunanjih koordinatah. Najprej nastavite v polju **IZBIRA REFERENCE** možnost **Zunanje koordinate**. S klikom na gumb **Referenca - zunanje koordinate** se vam odpre ustrezna Matlab funkcija. V tej funkciji določite referenco za kroženje vrha robota okoli začetne lege s frekvenco  $f = 0,03$  rad/s, polmerom kroženja  $r = 0,15$  m ter faznim zamikom  $\varphi = \pi/2$ .

```
function x_r= RefZunanje(t,x_0)
```

Funkcija RefZunanje ima vhode čas  $t$  in začetni položaj vrha robota v metrih  $x_0$  (vsebuje začetni vrednosti v smeri  $x$  in  $z$ ). Kot izhod je potrebno določiti ustrezne referenčne pozicije  $x_r$ .

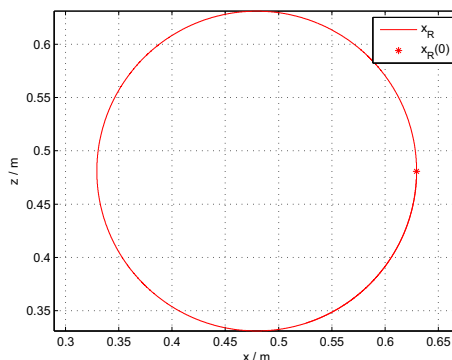
Kroženje je v osnovi definirano kot gibanje v dveh koordinatah po sinusu oziroma kosinusu.

$$x = r \sin(\omega t)$$

$$y = r \cos(\omega t).$$

Upoštevajte, da želite kroženje okoli začetne lege robota z določenim faznim zamikom.

- 17 Pravilnost reference preverite z gumbom **TEST REFERENCE**. Kot referenca se vam mora izrisati krožnica kot je na sliki 3.22.



Slika 3.22: Referenca za gibanje vrha po krožnici

- 18 Povezavo med notranjimi koordinatami (koti v sklepi) in zunanjimi koordinatami (lega vrha robota) opisuje direktna kinematika. Gumb **2 DOF direktna kinematika** vam odpre Matlab funkcijo, kjer zapišete enačbe direktne kinematike 2DOF manipulatorja. Pri tem upoštevajte ustrezno umestitev 2DOF manipulatorja v model robota Motoman MH5. Pri določevanju ustreznih relacij med koti robota Motoman MH5 in dvosegmentnega manipulatorja si pomagajte s sliko 3.7.

```
function x = DirKin(q)
```

Funkcija `DirKin` ima vhod vektor notranjih koordinat  $\mathbf{q}$  (kota v sklepih  $q_2$  in  $q_3$ ), izhod pa vektor zunanjih koordinat  $\mathbf{x}$  (položaj vrha manipulatorja v smeri  $x$  in smeri  $z$ ).

V funkciji je potrebno definirati ustrezni dolžini segmentov 2DOF manipulatorja ( $L_1$  in  $L_2$ ) ter pretvorbo med koti 2DOF manipulatorja (TH1 in TH2) in koti v sklepih robota Motoman MH5  $q_2$  in  $q_3$ :  $\vartheta = \mathbf{f}(\mathbf{q})$ . Potrebno je upoštevati tudi premik baze 2DOF manipulatorja v drugi sklep robota Motoman MH5. V pomoč so vam slike 3.5, 3.6 in 3.7.

Enačbi direktne kinematike dvosegmentnega manipulatorja sta

$$\mathbf{x} = \begin{bmatrix} l_1 \cos \vartheta_1 + l_2 \cos(\vartheta_1 + \vartheta_2) \\ l_1 \sin \vartheta_1 + l_2 \sin(\vartheta_1 + \vartheta_2) \end{bmatrix}.$$

- 19 Glede na shemo 3.8 za vodenje z uporabo sklepnega regulatorja potrebujemo še pretvorbo iz zunanjih koordinat (položaj vrha robota) v notranje koordinate (kote v sklepih). Te relacije nam opisuje inverzna kinematika. Gumb **2 DOF inverzna kinematika** vam odpre Matlab funkcijo, kjer zapišete enačbe inverzne kinematike 2DOF manipulatorja. Pri tem upoštevajte konfiguracijo "komolec zgoraj".

```
function q = InvKin(x)
```

Funkcija `InvKin` ima vhod vektor zunanjih koordinat  $\mathbf{x}$  (položaj vrha manipulatorja v smeri  $x$  in smeri  $z$ ), izhod pa vektor notranjih koordinat  $\mathbf{q}$  (kota v sklepih  $q_2$  in  $q_3$ ).

V funkciji je potrebno definirati ustrezni dolžini segmentov 2DOF manipulatorja ( $L_1$  in  $L_2$ ). Določite pretvorbo med koti v sklepih robota Motoman MH5  $q_2$  in  $q_3$  in koti 2DOF manipulatorja (TH1 in TH2):  $\mathbf{q} = \mathbf{f}(\vartheta)$ , obratno kot pri direktni kinematiki. Upoštevajte tudi premik iz baze 2DOF manipulatorja, ki je v drugem sklepu robota MH5, v bazo robota Motoman MH5. V pomoč so vam slike 3.5, 3.6 in 3.7.

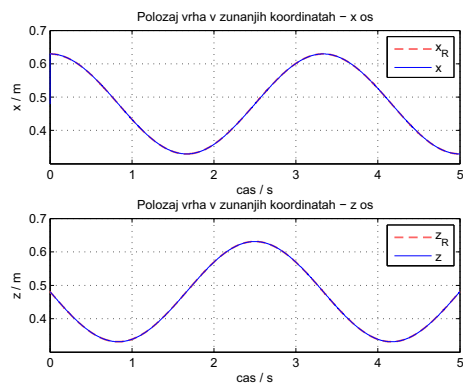
Enačbi inverzne kinematike dvosegmentnega manipulatorja v konfiguraciji "komolec zgoraj" sta

$$\vartheta_2 = -\arccos \frac{x^2 + z^2 - l_1^2 - l_2^2}{2l_1 l_2}$$

$$\vartheta_1 = -\arctan \left( \frac{z}{x} \right) - \arctan \left( \frac{l_2 \sin \vartheta_2}{l_1 + l_2 \cos \vartheta_2} \right).$$

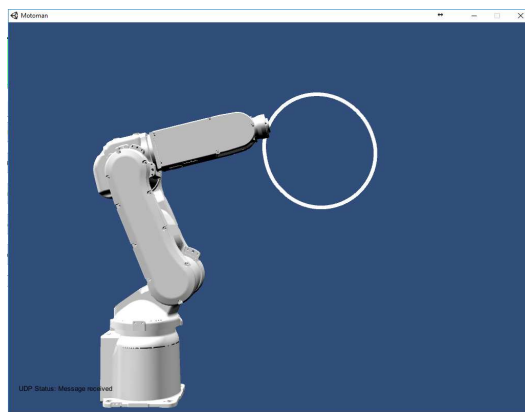
- 20 Delovanje inverzne in direktne kinematike preverite z gumbom **TEST KINEMATIKE**. S tem testom referenco kroženja v zunanjih koordinatah pre-

tvorite z inverzno kinematiko v notranje koordinate  $\mathbf{q}$ , te pa z direktno kinematiko nazaj v zunanje koordinate  $\mathbf{x}$ . Rezultat na grafu pokaže, da položaj vrha robota soupada z določeno referenco, kot prikazuje slika 3.23.



Slika 3.23. Primerjava reference in položaja vrha po preračunu referenca-inverzna kinematika-direktna kinematika

- 21 V nadaljevanju boste testirali gibanje robota MH5 z vizualizacijo. Sledite korakom v točki 13 in točki 14. Vrh robota mora opisati krog, kot je prikazano na sliki 3.24.



Slika 3.24. Gibanje vrha robota MH5 pri definiranem kroženju v zunanjih koordinatah

- 22 Spreminjajte vrednosti ojačenj  $K_p$ , da dobite i) sistem, ki ne more slediti referenci, ii) nestabilen sistem.
- 23 Ustavite simulacijo (gumb **USTAVI SIMULACIJO**) ter zaprete vizualizacijo (gumb **ZAPRI VIZUALIZACIJO**).



## Poglavje 4

# SIMULACIJA VARJENJA S SINHRONIM DELOM ROBOTOV MOTOMAN MH5

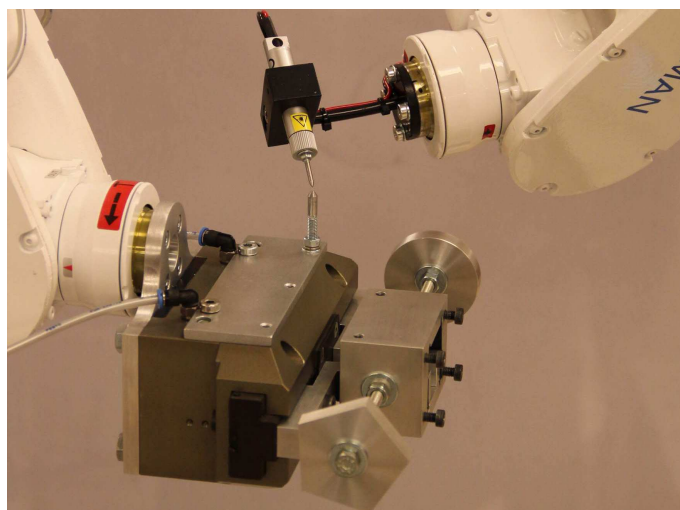
### 4.1 Cilj vaje

#### 1. del

Na robotu št. 2 kalibrirajte konico laserskega kazalnika kot novo orodje oz. nov vrh robota. Kot statični objekt uporabite konico, ki je nameščena na robotu št. 1 (Slika 4.1). Kalibracijo novega orodja tudi preverite.

#### 2. del

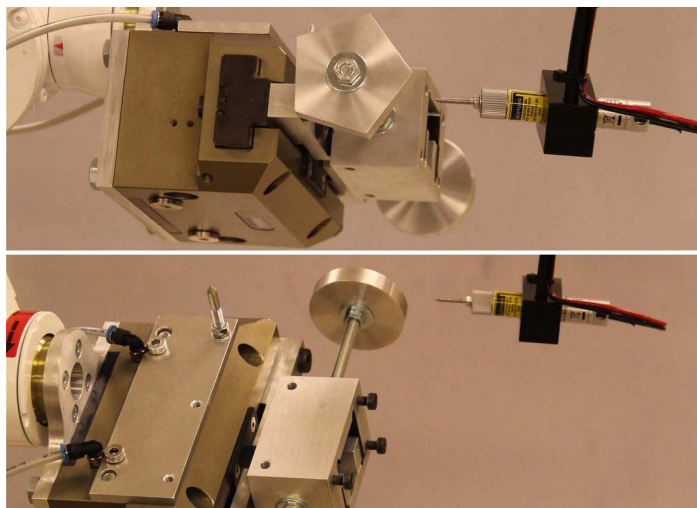
Med seboj kalibrirajte oba robota, kar je predpogoj za sinhrono delo robotov. Kalibracijo sinhronnega premikanja robotov tudi preverite.



Slika 4.1. Kalibracija konice kazalnika in preverjanje medsebojne kalibracije robotov

### **3. del**

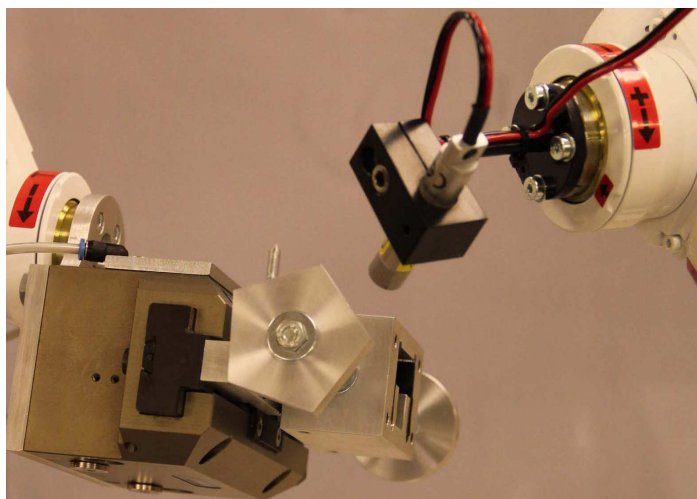
Z lastnim programom za premikanje robota preizkusite razliko med ukazoma za premikanje MOVL in SMOVL za robot št. 2 glede na robot št. 1. Slednji mora med premikanjem spremeniti orientacijo vrha (Slika 4.2).



Slika 4.2: Premiki robotov pri testiranju ukazov MOVL in SMOVL za robot št. 2

### **4. del**

Zapišite nov program za sinhrono premikanje obeh robotov med namišljenim varjenjem po robu petkotnika in kroga. Pri nalogi mora robot št. 1 prilagajati lego objekta robotu št. 2 (Slika 4.3).

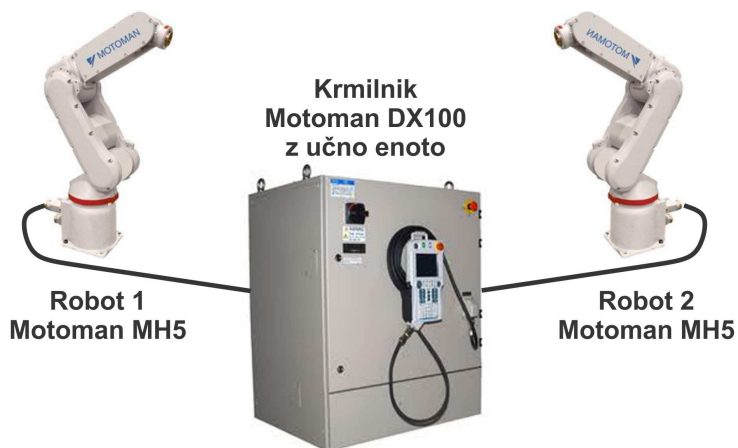


Slika 4.3: Izsek iz sinhronega sledenja robu objektov na robotu št. 1 z robotom št. 2



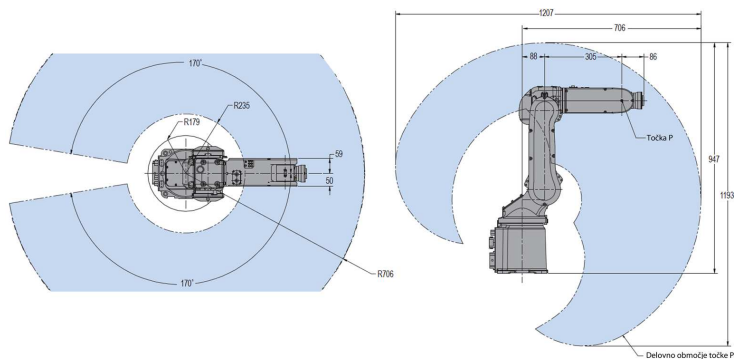
## 4.2 Struktura sistema

Osnova robotskega sistema (Slika 4.4) je krmilnik Motoman DX100. Nanj sta priključena dva robotska mehanizma Motoman MH5 in prenosna učna enota.



Slika 4.4: Struktura robotskega sistema

## 4.3 Robotski mehanizem MH5

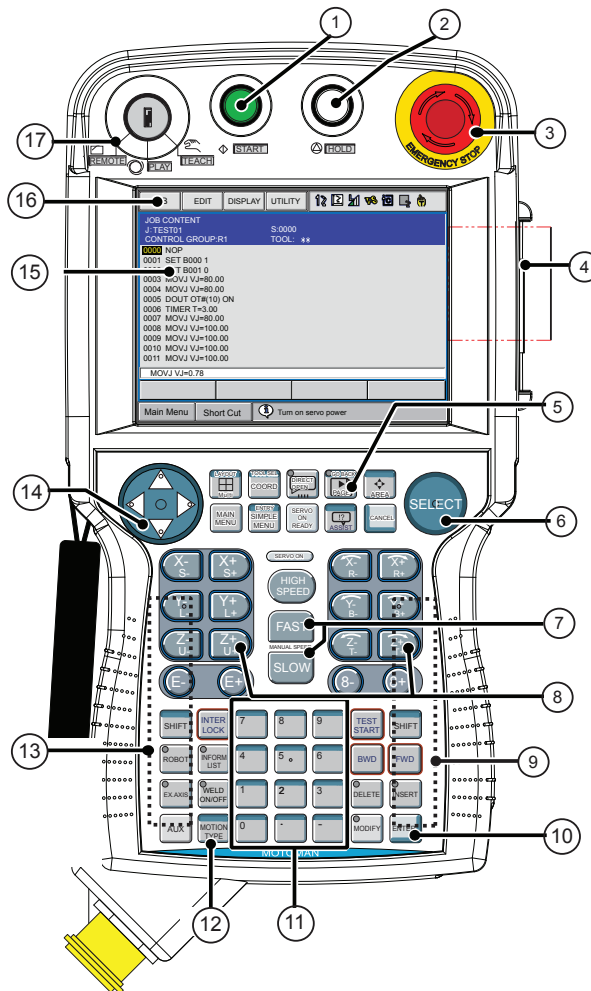


Slika 4.5: Delovni prostor robotskega mehanizma MH5

Število prostostnih stopenj:	<b>6</b>
Največja obremenitev:	<b>5 kg</b>
Ponovljivost:	<b>±0.02 mm</b>
Masa mehanizma:	<b>27 kg</b>

Slika 4.6: Lastnosti robota Motoman MH5

#### 4.4 Ročna učna enota



1	<b>START</b> gumb	9	Gumb za start napajanja motorjev
2	<b>HOLD</b> gumb	10	Tipka <b>ENTER</b>
3	<b>Gumb zasilne zaustavitve</b>	11	Numerične in funkcijske tipke
4	Reža za CF kartico	12	Tipka <b>MOTION TYPE</b>
5	Tipka <b>PAGE</b>	13	Gumb za start napajanja motorjev
6	Tipka <b>SELECT</b>	14	Smerna tipka
7	Tipke za nastavljanje hitrosti premikanja robota	15	Na dotik občutljiv zaslon
8	Tipke za premikanje robota	16	Glavni zaslonski menu
		17	Ključ za izbiro načina delovanja sistema

Slika 4.7: Ročna učna enota in pomen posameznih tipk

#### 4.4.1 Pomen uporabljenih tipk

##### Tipke za vključitev napajanja in ustavljanje robota



Omogoči vključitev napajanja motorjev. Tipko pritisnemo vedno, ko je napajanje motorjev izključeno ali po pritisku tipke za zasilno zaustavitev robota. Po pritisku utripa lučka **SERVO ON**.

##### **Gumb za start napajanja motorjev**

S pritiskom na tipko na zadnji strani učne enote vključimo napajanje motorjev. To je možno šele, ko lučka **SERVO ON** utripa. Če tipko pritisnemo povsem do konca, se napajanje motorjev zopet prekine.

##### Tipke za premikanje robota in izbiro koordinatnega sistema



Če izberemo premikanje robota po sklepih, s tipkami (S-,S+), (L-,L+), (U-,U+), (R-,R+), (B-,B+) in (T-,T+), robot premikamo po posameznem sklepu neodvisno eden od drugega.

Če izberemo premikanje robota v izbranem kartezičnem koordinatnem sistemu, ga s tipkami (X-,X+), (Y-,Y+) in (Z-,Z+) transliramo v izbrani smeri koordinatne osi. S tipkami ( $\overleftarrow{X-}, \overrightarrow{X+}$ ), ( $\overleftarrow{Y-}, \overrightarrow{Y+}$ ) in ( $\overleftarrow{Z-}, \overrightarrow{Z+}$ ) vrh robota rotiramo okrog izbrane koordinatne osi.



Izberemo koordinatni sistem (Slika 4.8) za premikanje robota. Vsakič, ko pritisnemo na tipko, izberemo drug koordinatni sistem. Trenutno izbrani koordinatni sistem je prikazan s sliko povsem zgoraj na zaslonu učne enote. Pomen teh slik je zapisan spodaj.



Robot premikamo po sklepih neodvisno.



Robot premikamo v kartezičnem baznem k.s.



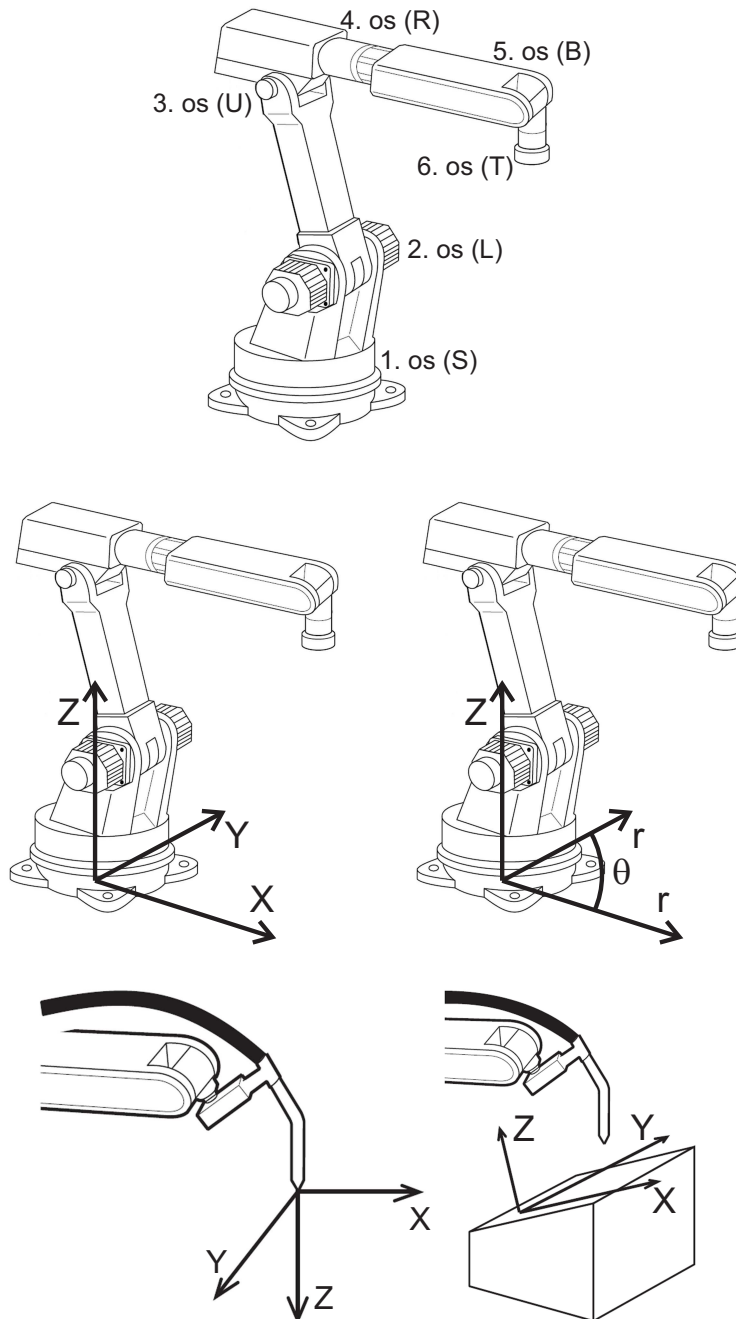
Robot premikamo v cilindričnem koordinatnem sistemu.



Robot premikamo v koordinatnem sistemu orodja.



Robot premikamo v zunanjem k.s. objekta.



Slika 4.8: Koordinatni sistemi za premikanje robota

### Tipke za izbiro hitrosti premikanja robota pri ročnem vodenju



Uporabnik viša hitrost ročnega vodenja robota.



Uporabnik nižja hitrost ročnega vodenja robota.



Pritisk te tipke z vsaj eno tipko za premikanje robota povzroči premikanje robota z največjo hitrostjo.

Trenutno izbrana hitrost ročnega premikanja robota je prikazana s sliko povsem zgoraj na zaslonu učne enote. Pomen teh slik je zapisan spodaj.



Diskretni kratki premiki (Inching).



Najpočasnejša hitrost (zvezni premiki).



Srednja hitrost (zvezni premiki).



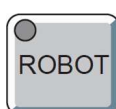
Največja hitrost (zvezni premiki).

### Tipke za izbiro robota

Sistem sestavlja dva robota, ki sta povezana na enoten krmilnik. **Robot št. 1 je definiran kot gospodar (Master), robot št. 2 pa kot služabnik (Slave). Zato robot št. 1 vedno nosi obdelovanec, robot št. 2 pa orodje za delo na obdelovancu.** Premikamo ju lahko vsakega posebej in tudi sinhrono. Način izbiranja, kateri robot bomo premikali oz. ali ju bomo premikali sinhrono, je odvisen od naslednje nastavitve. Ta se določa pri ustvarjanju novega programa za premikanje robota (JOB), kjer uporabnik določi (GROUP SET), ali bo program vodil samo en robot (R1 ali R2) ali pa oba (R1+R2:R1).



V primeru, da je program zapisan za vodenje samo enega robota (GROUP SET je R1 ali R2), je potrebno spodnjim pritiskom na gumbe dodati pritisk na tipko SHIFT. **Na zaslonu izbira GROUP SET ni nikjer izpisana!**



Izbiramo vodenje robota št. 1 ali št. 2. V primeru, da pri ustvarjanju novega programa nismo izbrali skupine R1+R2:R1, je za vodenje posameznega robota potrebno pritisniti ROBOT + SHIFT.

V statusni vrstici se ob ustreznih pritiskih na gumba ROBOT ali 7 (z ali brez skupnega pritiska tipke SHIFT) spreminjajo sličice, ki imajo naslednji pomen:



Vodimo robot št. 1.



Vodimo robot št. 2.



Sinhrono vodimo tako robot št. 1 kot št. 2 (tipka 7).



S pritiskom izbiramo vodenje enega robota ali sinhrono vodenje obeh robotov skupaj. V primeru, da pri ustvarjanju novega programa nismo izbrali skupine R1+R2:R1, je za preklp vodenja posameznega robota potrebno pritisniti tipko 7 skupaj s tipko SHIFT.



S pritiskom izbiramo sinhrono ali posamično interpolacijo premikanja pri učenju sinhronnega dela.

#### 4.4.2 Zaslón učne enote

Zaslón je občutljiv na dotik in razdeljen na 5 delov.



1	Področje izbire menujev	2	Statusna vrstica
3	Splošno področje	4	Uporabniško področje
5	Glavni menu		

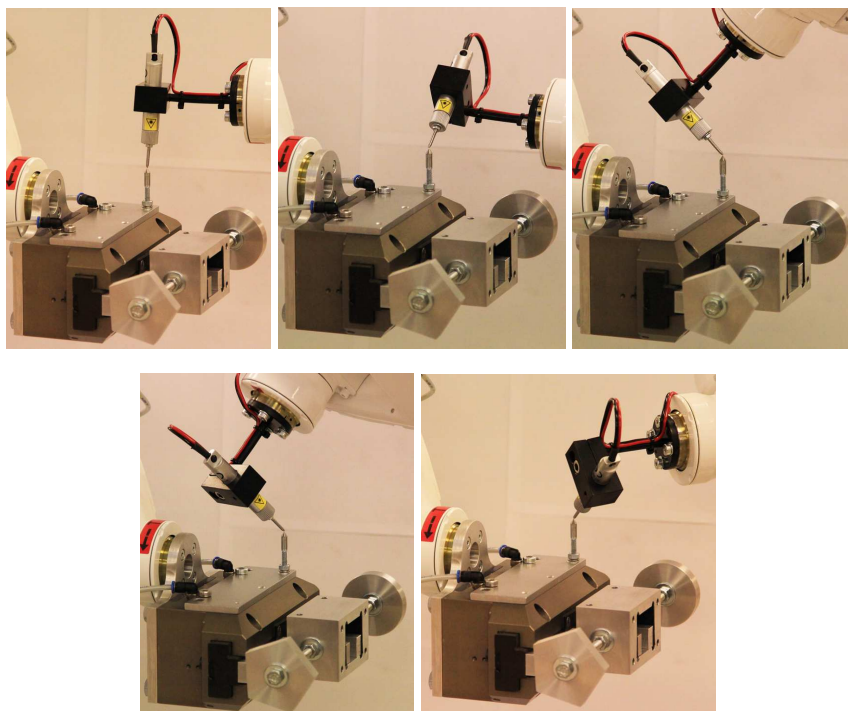
Slika 4.9: Razdelitev zaslona

## 4.5 Izvedba 1. dela vaje - kalibracija orodja

Da robot, z nameščenim orodjem, pravilno naredi linearen ali krožen premik, je potrebno definirati nov vrh oz. koordinatni sistem orodja (TCP, ang. Tool Center Point) glede na lego zadnjega segmenta robota. **Postopku pravimo kalibracija orodja.** Postopek od uporabnika zahteva, da vrh orodja postavi v isto točko v prostoru v petih različnih orientacijah orodja. Primer ustreznih leg prikazuje slika 4.10.

Pred začetkom 1. dela naloge je potrebno ustvariti nov program (**JOB**):

- V glavnem meniju pritisnete gumb **JOB** in nato **CREATE NEW JOB**. Odpre se okno za definiranje parametrov programa.
- V polju **JOB NAME (\*\*\*)** pritisnete tipko **SELECT** in odprete okno za definiranje imena programa. Za potrditev imena pritisnete tipko **ENTER**.
- V izbiri (pomik s smernimi tipkami) **GROUP SET** s tipko **SELECT** izberete **R1+R2:R1**.
- Parametre novega programa potrdite s tipko **EXECUTE**.



Slika 4.10: Leg špice kazalnika pri kalibraciji novega orodja

### Postopek nove kalibracije orodja

- Robot št. 1 s tipkami za premikanje postavite tako, da je konica na vrhu prijemala dosegljiva špici kazalnika na robotu št. 2.
- **Med kalibracijo novega vrha robota premikajte samo robot št. 2!**
- S tipkami za premikanje robota premikajte robot št. 2 tako, da konico kazalnika staknete s špico na robotu št. 1 (Slika 4.10).
- V glavnem meniju izberete **ROBOT** in nato še **TOOL**.
- **Kot številko orodja izberite 02** (za premik uporabite smerne tipke).
- S tipko **SELECT** potrdite izbrano številko orodja.
- Odpre se okno, kjer je zapisano ime orodja (bi že moralo biti **R2SpicaVSP**) in podatki o legi ter masi orodja.
- V področju izbire menujev izberete opcijo **UTILITY** in nato iz zavesnega menija še **CALIBRATION**.
- V področju izbire menujev izberete opcijo **DATA** in nato še **CLEAR DATA**, da zbrisete prejšnje podatke. Odločitev je potrebno potrditi s pritiskom gumba **YES**, ki se pojavi na zaslonu.
- V polju (\*\*\*) pritisnete tipko **SELECT** in izberete **R2:ROBOT2**.
- Ker sta konici robota št. 1 in robota št. 2 že staknjeni, pritisnete tipko **MODIFY**, da zagori njena lučka, in potrdite s tipko **ENTER**.
- Izberete točko učenja **TC2**, kar storite tako, da pritisnete tipko **SELECT** in iz menija s smernimi tipkami izberete **TC2**.
- Robot št. 2 premaknete tako, da se s konico kazalnika dotika špice na robotu št. 1, vendar v drugi orientaciji kot prej.
- Ko je lega zadovoljiva, kalibracijsko točko **TC2** potrdite s pritiskom tipk **MODIFY** in **ENTER**.
- Ta postopek ponovite še za kalibracijske točke **TC3**, **TC4** in **TC5**. V pomoč pri določanju orientacij orodja na robotu št. 2 naj bo slika 4.10.
- Ko zaključite učenje kalibracijskih točk za kalibracijo novega vrha robota št. 2 na konici kazalnika, pritisnete tipko **COMPLETE**. Tako se avtomatsko izračuna lega novega orodja glede na lego zadnjega segmenta robota.



### Postopek preverjanja kvalitete kalibracije orodja na obeh robotih

- Izberete, kateri robot boste premikali (tipka **ROBOT** na učni enoti).
- Robot št. 1 oz. 2 postavite prosto v prostor. S tipko **COORD** izberete premikanje robota v koordinatnem sistemu orodja. Spreminjanje koordinatnih sistemov spremljate v statusni vrstici.
- Določiti morate, v točno katerem koordinatnem sistemu orodja boste robot premikali. S skupnim pritiskom tipk **SHIFT** in **COORD** odprete seznam vseh definiranih orodij. S smernimi tipkami se postavite na **R1 TOOL01 za premikanje robota št. 1** oz. **R2 TOOL02 za premikanje robota št. 2**. Nazaj v prejšnje okno se vrnete z enako kombinacijo tipk.
- Kvaliteto kalibracije preverite z rotacijami (tipke  $(\overleftarrow{X}, \overrightarrow{X})$ ,  $(\overleftarrow{Y}, \overrightarrow{Y})$  in  $(\overleftarrow{Z}, \overrightarrow{Z})$ ) orodja okrog koordinatnih osi izbranega koordinatnega sistema orodja. Če konica kazalnika pri tem v prostoru navidezno miruje, je kvaliteta kalibracije konice zadovoljiva, drugače postopek ponovite.

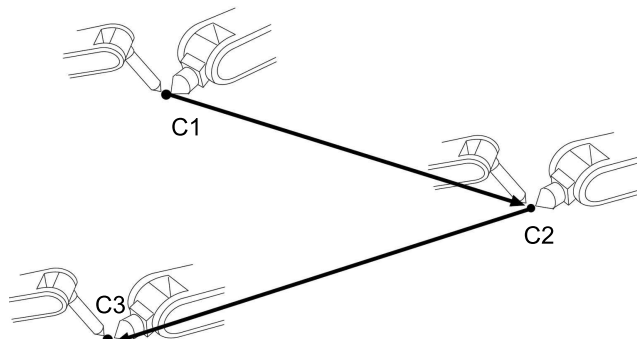
## 4.6 Izvedba 2. dela vaje - kalibracija robotov

Da se lahko robota med seboj sinhrono premikata, je potrebno opraviti postopek skupne kalibracije v prostoru. S tem definiramo lego baznega k.s. robota št. 2 v baznem k.s. robota št. 1.

Predhodno je potrebno imeti skalibrirana in izbrana ustrezna orodja na robotu št. 1 in št. 2. V našem primeru je orodje v obliki konice z imenom R1SpicaVSP že skalibrirano, konico kazalnika z imenom R2SpicaVSP pa ste skalibrirali v 1. delu naloge.

### Postopek nove kalibracije med robotoma

- Postopek kalibracije med robotoma zahteva, da skalibrirani konici orodij staknete v treh točkah v prostoru (Slika 4.11).
- V glavnem meniju izberete **ROBOT** in še **ROBOT CALIB**.
- S smernimi tipkami **izberete številko kalibracije 01** in jo potrdite s tipko **SELECT**. Odpre se okno **ROBOT CALIBRATION**.
- V področju izbire menujev izberete opcijo **DATA** in nato še **CLEAR DATA**, da zbrisete prejšnje podatke. Odločitev je potrebno potrditi s pritiskom gumba **YES**, ki se pojavi na zaslonu.



Slika 4.11: Kalibracijske točke v prostoru za kalibracijo med robotoma

- Ko se nahajate v polju (\*\*\*) , pritisnete tipko **SELECT** in izberete opcijo **R1+R2** (Slika 4.12).
- Kot kaže slika 4.11, v skupnem delovnem prostoru staknete konici orodij in te točke potrdite kot kalibracijske točke **C1**, **C2** in **C3**.
- Vsako točko shranite s kombinacijo tipk **MODIFY** in **ENTER**, točke pa izbirate s pritiski na tipko **SELECT**.
- Ko zaključite shranjevanje točk, pritisnite tipko **COMPLETE**.



Slika 4.12: Okno za medsebojno kalibracijo robotov

#### Priporočila za uspešnejšo kalibracijo robotov

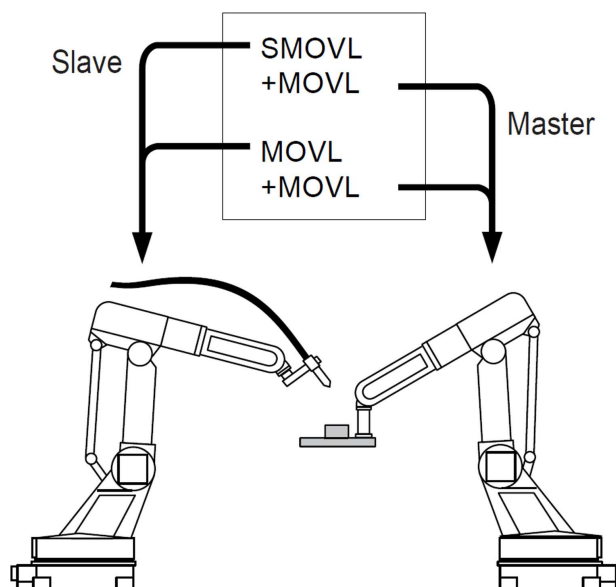
- Pri učenju točk C2 in C3 je orientacija orodij enaka kot pri točki C1.
- Priporočena minimalna razdalja med točkami C1-C2, C2-C3 in C1-C3 je 1 m, če robotski delovni prostor to dopušča.
- Točke C1, C2, C3 postavite v obliko trikotnika in ne v ravno črto.

**Z asistentom ali demonstratorjem preverite kvaliteto kalibracije obeh robot (tipka 7).**

#### 4.7 Izvedba 3. dela vaje - ukazi za sinhrono premikanje dveh robotov

Slika 4.13 kaže dva robota v relaciji **gospodar (Master)** in **služabnik (Slave)**. **Gospodar vedno nosi obdelovanec, služabnik pa z orodji opravlja razna dela na obdelovancu.** Zapisan je tudi primer ukaza za linearen premik, ko služabnik sledi premikanju gospodarja (**SMOVL +MOVL**) in neodvisno premikanje obeh (**MOVL +MOVL**).

**Zgornja vrstica v ukazu za premik robotov se nanaša na služabnika, spodnja pa na gospodarja.** Služabnik je tisti, ki mora sinhronizirati svoje premike glede na gospodarja. Zato v prvi vrstici lahko nastopa ukaz s črko S pred ukazom za premik (**SMOVL, SMOVC**).



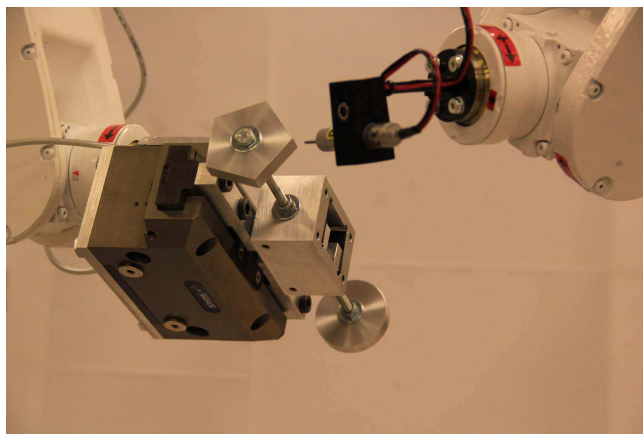
Slika 4.13: Robot št. 1 je gospodar (Master), št. 2 pa služabnik (Slave).

##### Koraki za izvedbo vaje

- Ker je za ponazoritev nesinhronega in sinhronega dela dveh robotov potrebno zapisati nekaj vrstic programa, odprite okno za urejanje programa, ki ste ga ustvarili v 1. delu vaje. Izberite izbiro **JOB** in še enkrat **JOB**.
- V odprtem oknu je spodaj zapisan ukaz **MOVJ VJ=0.78 PL=0**  
**+MOVJ VJ=0.78**

Ta dvojni ukaz pomeni, da se bosta oba robota neodvisno premikala v načeno lego s sinhronim premikanjem po sklepih (J pomeni Joint) oz. v notranjih koordinatah.

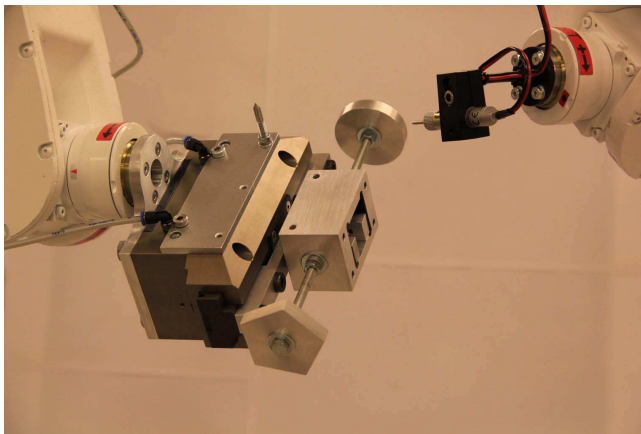
- Želja je, da konica orodja na robotu št. 2 potuje po premici, kar zahteva premik robota v zunanjih koordinatah oz. nadzorovano premikanje vrha robota. Zato je potrebno način premikanja spremeniti. To storite s pritiskom na tipko **MOTION TYPE**. Ko pritisnete tipko, se za trenutno izbran robot (številka v statusni vrstici) spremeni način premikanja. Nastavitev je potrebno spremeniti za oba robota, da je v spodnjem delu zaslona zapisano **MOVL V=11.0 PL=0**  
**+MOVL V=11.0**
- Robotu št. 1 rotirajte 6. sklep tako, da bo orientacija vrha robota podobna orientaciji na sliki 4.14. Tudi robot št. 2 postavite v podobno lego, kot jo prikazuje slika 4.14



Slika 4.14: Lega 1

- Ko je lega obeh robotov zadovoljiva, pritisnete tipko **ENTER**. S tem v program dodate ukaz, ki določa nesinhron premik robotov v lego št. 1.
- Spremeniti je potrebno tudi hitrost premikanja robotov iz  $V=11.0$  v  $V=100.0$  (enote so cm/min), in to za oba robota. S kurzorjem **označite celotno prvo vrstico** (pomik v desno) in pritisnete tipko **SELECT**. S tem kurzor prestavite v spodnji del zaslona, kjer se s smernimi tipkami premikate po parametrih ukazov.
- Označite  $V=11.0$  za robot št. 2 (Slave), pritisnete **SELECT** in v prikazano polje vpišete **100** ter potrdite s tipko **ENTER**.
- Označite  $V=11.0$  za robot št. 1 (Master), pritisnete **SELECT** in v prikazano polje vpišete **100** ter potrdite s tipko **ENTER**.
- Spremembo hitrosti premikanja za oba robota potrdite s tipko **ENTER**.

- Rotirajte 6. sklep robota št. 1 (Master) približno tako, kot prikazuje slika 4.15. Podobno storite za robot št. 2.



Slika 4.15: Lega 2

- Lego skupaj z ukazi za nesinhron premik potrdite s pritiskom na tipko **ENTER** (kurzor mora označevati številko vrstice programa).
- Za razliko od prejšnjih ukazov za premikanje obeh robotov **želimo v naslednjem koraku sinhron premik robota št. 2 glede na premikanje robota št. 1**. Za spremembo je potrebno pritisniti tipko **4** na učni enoti. S tem v spodnjem delu zaslona spremenite obliko ukaza v **SMOVL V=100.0 PL=0 +MOVL**

**Ukaz določa sinhron premik služabnika glede na premik gospodarja, pri čemer hitrost premikanja določa hitrost premika služabnika.**

- S pritiskom na tipko **ENTER** ukaz dodate v program.

### Koraki za testiranje 3. dela vaje

- Postavite se v **prvo vrstico (0001) programa** in pritisnite tipko **FWD**, da se robota premakneta v **lego 1**.
- Postavite se v **drugo vrstico (0002) programa** in pritisnite tipko **FWD**, da se robota premakneta v **lego 2**.

**Ukaz MOVL določa obema robotoma, da naj svoja definirana vrha iz lege 1 v lego 2 premakneta po svoji premici (pri robotu št. 1 to ni očitno). Kazalnik na robotu št. 2 pa se premakne iz lege 1 v lego 2 po premici. To ni sinhron premik glede na premik robota št. 1!**

- Postavite se v **prvo vrstico (0001) programa** in pritisnite tipko **FWD**, da se robota premakneta **v lego 1**.
- Postavite se v **tretjo vrstico (0003) programa** in pritisnite tipko **FWD**, da se robota premakneta **v lego 2**.

**Ukaz SMOVL določa robotu št. 2 (Slave), da naj iz lege 1 v lego 2 sledi navidezni premici med lego 1 in lego 2 na objektu robota gospodar. Tak način premikanja definirajmo kot sinhrono premikanje robota služabnik (št. 2) glede na gospodarja (št. 1.)**

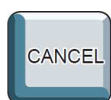
#### 4.8 Izvedba 4. dela vaje - usklajeno premikanje dveh robotov

Glavni cilj vaje je sinhrono premikanje dveh robotov pri namišljenem varjenju roba petkotnika in kroga. Situacijo prikazuje slika 4.3. Za premikanje po robu petkotnika uporabljajte linearne premike (MOVL in SMOVL) konice orodja na robotu št. 2, na krogu pa krožne premike (MOVJ in SMOVJ). Za hitro spremembo konfiguracij robotov uporabite tudi premikanje robota po sklepih (MOVJ). Priporočljivo je, da ustvarite nov program (stran 49) ter da z ustreznim ukazom ob ustreznem času prižigate ter ugašate laserski kazalnik.

##### Tipke za izvedbo vaje



Tipka za potrjevanje oz. dodajanje novih ukazov, potrjevanje vnešenih vrednosti parametrov itn.



Prekine trenutni status sistema. Najpogosteje se tipko uporablja za prekinitve sporočila napake.



Pritisk na tipko izvede program v trenutni vrstici v smeri naprej.



Pritisk na tipko izvede program v trenutni vrstici v smeri nazaj.



+



Kombinacija teh dveh tipk sproži izvajanje programa v celoti (dovolj držanje ene tipke).



Zbrišemo že vnešen ukaz v programu. Ob pritisku na to tipko na njej zagori lučka. Ukaz potrdimo s tipko **ENTER**.



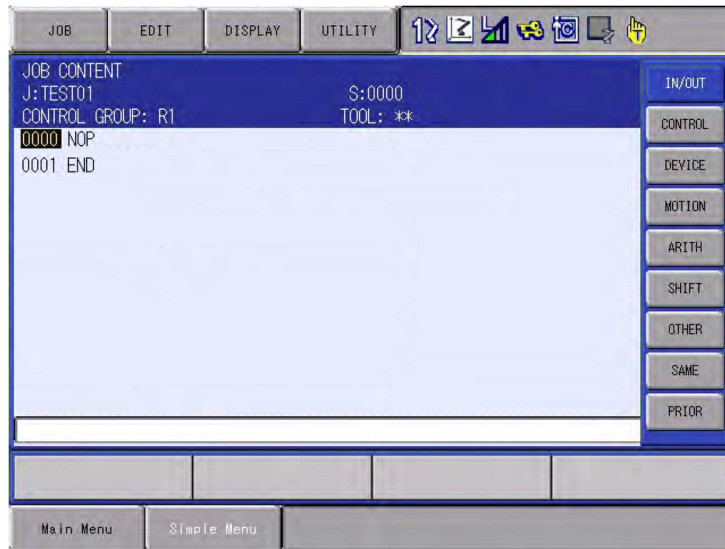
Dodamo nov ukaz v program. Ob pritisku na to tipko na njej zagori lučka. Ukaz potrdimo s tipko **ENTER**.



Spremenimo že obstoječ ukaz v programu. Ob pritisku na to tipko na njej zagori lučka. Ukaz potrdimo s tipko **ENTER**.



Pritisk na tipko odpre prvi nivo zbirke ukazov (Slika 4.16). Po menutih se sprehajamo s smernimi tipkami, ukaz pa v program vnesemo s pritiskom na tipko **SELECT**. Za prehod v druga območja zaslona s prstom pritisnemo na željen del zaslona ali pa pritisnemo tipko **AREA**.



Slika 4.16: Odprt prvi nivo zbirke ukazov

### Potrebni ukazi za izvedbo vaje

**DOUT OT#(18) ON //LASER**  
**DOUT OT#(18) OFF //LASER**

Z ukazom določimo, kateri digitalni izhod postavimo na visok oz. nizek nivo. Za kazalnik uporabimo **naslov 18**. Parameter **ON** pomeni, da nivo digitalnega izhoda dvignemo, z **OFF** pa ga spustimo. Za spremembo številke izhoda uporabimo postopek za spreminjanje parametrov ukazov.

---

<b>MOVJ</b>	Interpolacijo po sklepih uporabljamo, ko ni potrebno kalibriranega orodja na vrhu robota premakniti po znani krivulji proti naslednji shranjeni legi.
-------------	---

---

<b>MOVL</b>	Z ukazom določimo, da izbrano orodje na vrhu robota iz ene v drugo točko premaknemo po premici.
-------------	---

---

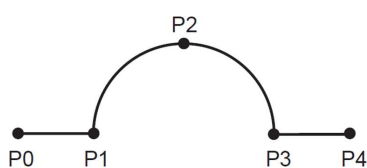
<b>SMOVL</b>	Z ukazom določimo, da izbrano orodje na vrhu robota služabnik (Slave) iz ene v drugo točko premakne po premici, pri čemer sledi premikanju robota gospodar (Master).
--------------	--

---



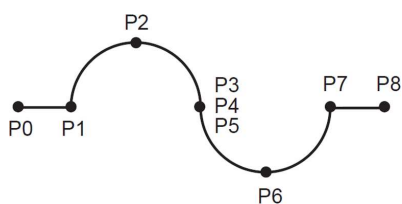
**MOVC** Z ukazom določimo, da izbrano orodje na vrhu robota opiše krožni premik. Krožni premik določimo, kot kažeta sliki 4.17 in 4.18.

**SMOVC** Z ukazom določimo, da izbrano orodje na vrhu robota služabnik (Slave) opiše krožni premik, pri čemer sledi premikanju robota gospodar (Master). Krožni premik določimo po sliki 4.17 in 4.18.



Slika 4.17: Polkrog

**P0 ... MOVJ ali MOVL**  
**P1 ... MOVC (linearen premik)**  
**P2 ... MOVC**  
**P3 ... MOVC**  
**P4 ... MOVJ ali MOVL**



Slika 4.18: Več polkrogov

**P0 ... MOVJ ali MOVL**  
**P1 ... MOVC (linearen premik)**  
**P2 ... MOVC**  
**P3 ... MOVC**  
**P4 ... MOVJ ali MOVL**  
**P5 ... MOVC**  
**P6 ... MOVC**  
**P7 ... MOVC**  
**P8 ... MOVJ ali MOVL**

## OPOZORILO!

Pri programiranju je potrebna posebna previdnost, da robota ne trčita, saj s tem lahko povzročimo poškodbe delovne opreme.

Pri testiranju napisanega programa moramo le-tega preveriti od vrstice do vrstice oz. po korakih, saj le na ta način zagotovimo varno delo z robotom.

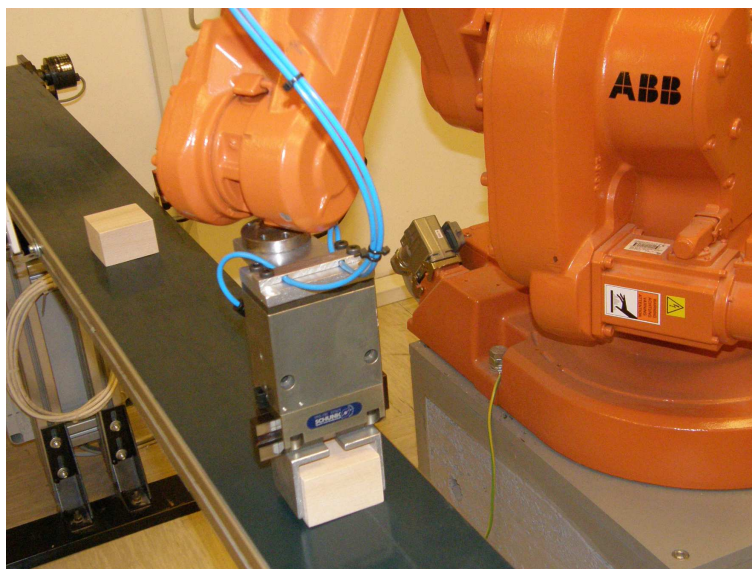


## Poglavje 5

# ROBOT ABB IRB 1600 IN TEKOČI TRAK

### 5.1 Cilj vaje

Pri nalogi uporabite industrijski robot ABB IRB 1600, nanj pritrjeno pnevmatsko prijemalo, tekoči trak in kocko. Z uporabo preprostih ukazov napišite robotski program, da robot s police ob tekočem traku pobere kocko in jo postavi na premikajoči se tekoči trak. Robot naj sinhrono sledi gibanju tekočega traku, pobere kocko in jo postavi na izhodiščno pozicijo.

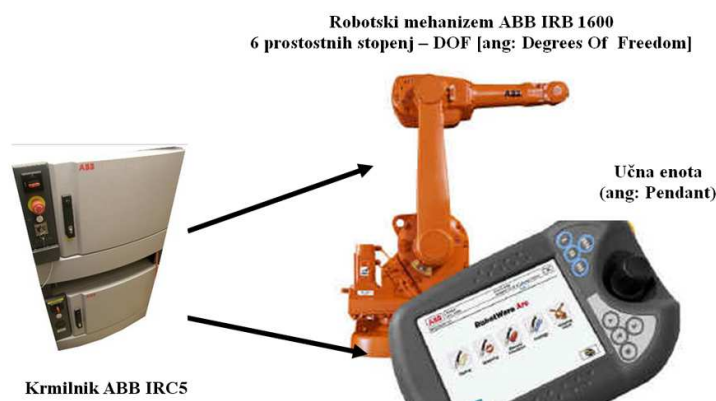


Slika 5.1. Robot ABB IRB 1600 s prijemalom s tekočega traku pobira leseno kocko.

## 5.2 Struktura sistema

Struktura osnovnega robotskega sistema robota ABB IRB 1600 je prikazana na sliki 5.2. **Osnovne komponente tega sistema so:**

- **krmilnik** ABB IRC5 s svojimi vhodno/izhodnimi enotami (5.2.1),
- vsaj en **robotski mehanizem** (5.2.2),
- **učna enota** za vodenje in programiranje robotskega sistema (5.2.3).



Slika 5.2: Struktura sistema

Najosnovnejše karakteristike sistema oz. mehanizma so:

- **mehanizem ima 6 rotacijskih prostostnih stopenj,**
- nosilnost robota pri največji hitrosti je 7 kg,
- ponovljivost pozicije je  $\pm 0.05$  mm,
- masa mehanizma je 250 kg.

Vendar pa za izpeljavo vaje osnovna struktura robotskega sistema ni dovolj. Tako potrebujemo še naslednje komponente, ki so pritrjene na robotski mehanizem ali povezane z robotskim krmilnikom.

- enkoderska kartica,
- izvor stisnjenega zraka (pnevmatika),
- elektro-pnevmatski ventil povezan z digitalnim izhodom (24 V) robotskega krmilnika,
- pnevmatsko prijemalo povezano z elektro-pnevmatskim ventilom in ga uporabljamo za prijem objektov (kocke),
- tekoči trak z asinhronskim pogonskim motorjem, ki se ga zaganja ter ustavlja s kontaktorjem povezanim z digitalnim izhodom (24 V) robotskega krmilnika,
- senzorja pozicije (enkoder) in optični senzor prisotnosti objekta na tekočem traku.

### 5.2.1 Krmilnik ABB IRC5

A ... Glavno stikalo.

B ... Velik rdeč gumb → ustavi gibanje robota.

C ... Signalna lučka, da so motorji napajani.

D ... Ključ za izbiro načina delovanja sistema:



avtomatski način,



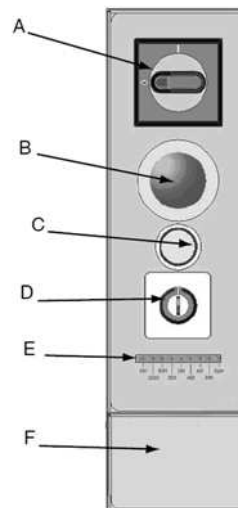
ročni način vodenja pri zmanjšani hitrosti (največja 250 mm/s),



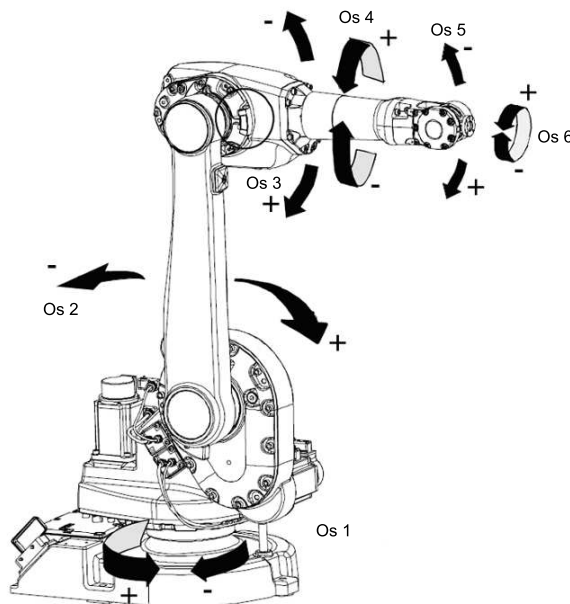
ročni način vodenja z neomejeno hitrostjo gibanja vrha.

E ... Vrsta LED kaže stanje varnostne verige.

F ... Vhodi (LAN, USB).



### 5.2.2 Robotski mehanizem ABB IRB 1600



Slika 5.3: Robotski mehanizem

### 5.2.3 Ročna učna enota

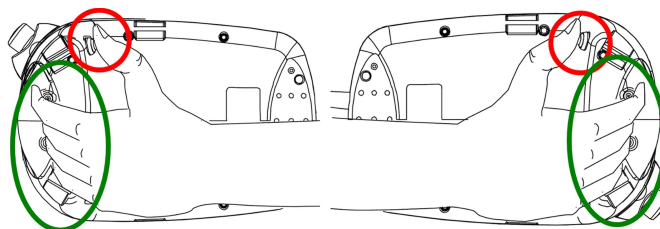
Na sliki 5.4 je prikazana ročna učna enota robota ABB IRB 1600, ki je priključena na njegov krmilnik ABB IRC5. Oznake pomenijo:

- A ... Zaslona, občutljiv na dotik.
- B ... Tipke, katerih funkcije lahko uporabnik določa sam.
- C ... Gumb zasilne zaustavitve robota.
- D ... Krmilna palica (ang. joystick).
- E ... *Hold-to-run* gumb - na zadnji strani učne enote (zagon programa v učnem načinu vodenja pri neomejeni hitrosti gibanja vrha robota).
- F ... Tipke za izvajanje programa.

**Na zadnji strani ročne učne enote je potrebno s prsti stisniti varnostno stikalo, sicer premikanje robota ni mogoče.**

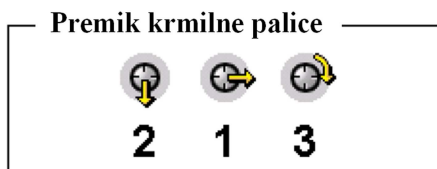


Slika 5.4: Učna enota

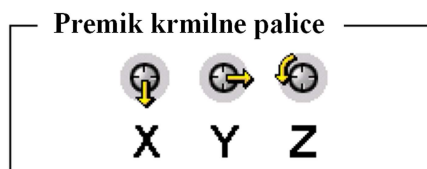


Slika 5.5. Prijem učne enote za desničarje in levičarje s ponazorjenim *Hold-to-run* gumbom

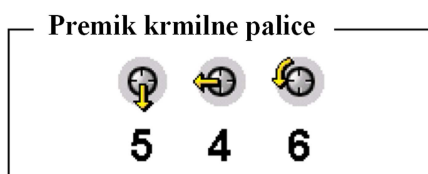
### Definirane smeri premika krmilne palice



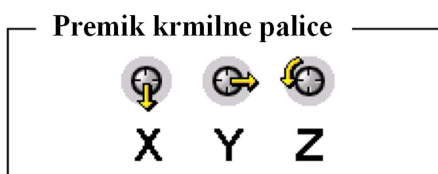
Premik robota po posameznih oseh: osi 1 do 3.



Translacija v smeri osi izbranega koordinatnega sistema.







Premik robota po posameznih oseh: osi 4 do 6.



Rotacija okrog osi izbranega koordinatnega sistema.

**Preden prvič premikate robot s krmilno palico, se morate zavedati, da robot doseže velike hitrosti in lahko hitro poškoduje človeka ali opremo. Pomembno je, da krmilno palico premikate z občutkom in z vedenjem, da je premik krmilne palice proporcionalen hitrosti premikanja!**

### Funkcijske tipke

-  → zagon tekočega traku
-  → zaustavitev tekočega traku
-  → stisk prijemala
-  → spust prijemala

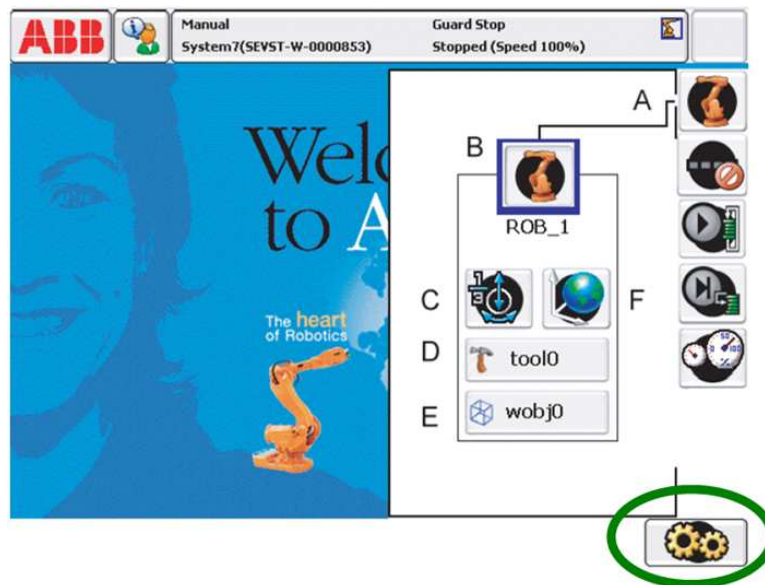


Slika 5.6. Tipke na ročni učni enoti za zagon oz. zaustavljanje tekočega traku in manipulacijo prijemala

**Funkcijo teh tipk lahko uporabnik nastavi po želji, vendar so funkcije za namen opravljanja laboratorijskih vaj določene vnaprej!**

### 5.3 Uporabniški vmesnik za premikanje robota

#### Način premikanja robota s krmilno palico (ikona C)



Slika 5.7: Menu za nastavljanje načina gibanja robota

- A ... Odpiranje menija za nastavljanje načina gibanja robota.
- B ... Izbira robota (v našem primeru samo ROB\_1).
- C ... Določanje načina gibanja robota.
- D ... Izbiranje koordinatnega sistema orodja.
- E ... Izbiranje zunanjega koordinatnega sistema.
- F ... Izbira koordinatnega sistema v katerem premikamo vrh robota.

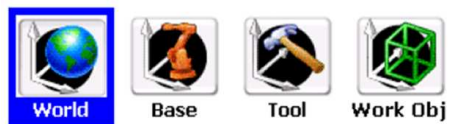


Slika 5.8: Izbira načina za premikanje robota

- Axis 1-3** → Rotiranje osi 1 do 3 (z uporabo krmilne palice).
- Axis 4-6** → Rotiranje osi 4 do 6 (z uporabo krmilne palice).
- Linear** → Translacijsko premikanje robota po ravni črti v izbranem koordinatnem sistemu (ikona F).
- Reorient** → Rotacijsko premikanje robota okrog osi izbranega koordinatnega sistema (ikona F).

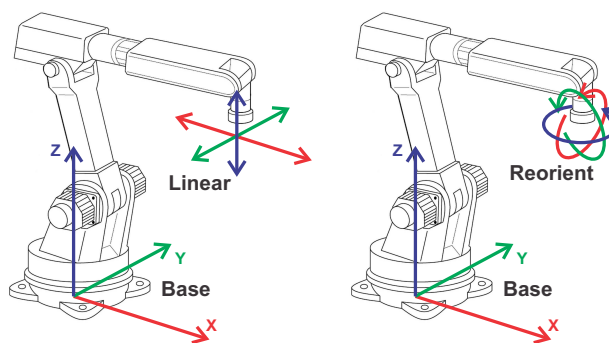


### Izbira koordinatnega sistema (ikona F)



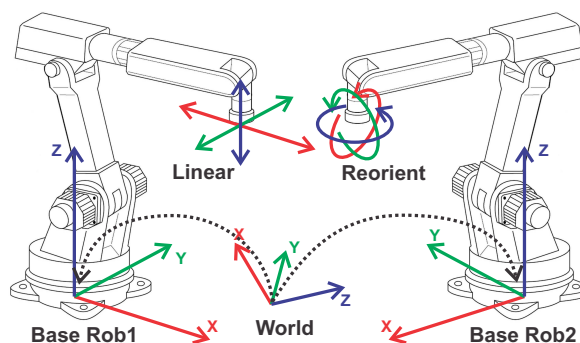
Slika 5.9: Izbiranje koordinatnega sistema za premikanje robota

Z izbiro **Base** določimo, da se vrh robota pri izbiri *Linear* (ikona C) translira vzdolž osi baznega koordinatnega sistema robota ali z izbiro *Reorient* (ikona C) rotira okrog osi baznega koordinatnega sistema (Slika 5.10).

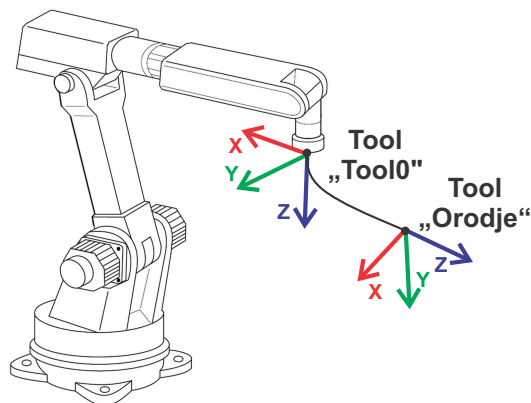


Slika 5.10: Premikanje vrha robota v baznem koordinatnem sistemu

Izbiri **World** uporabimo, če imamo v sistem vključena vsaj dva robotska mehanizma in jima določimo skupen zunanji koordinatni sistem. Glede na osi tega koordinatnega sistema lahko vrh obeh robotov linearno transliramo oziroma okrog njih vrh robota rotiramo (Slika 5.11). V našem primeru, ko je na krmilnik priključen samo en robotski mehanizem, je World koordinatni sistem istoležen z baznim (Base) koordinatnim sistemom robota.

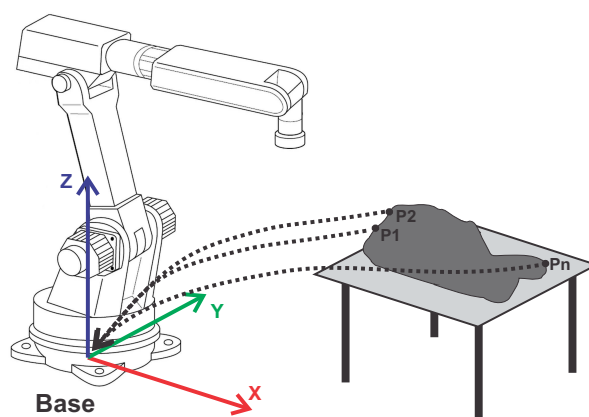


Slika 5.11: Uporaba World koordinatnega sistema



Slika 5.12: Premikanje vrha robota v koordinatnem sistemu orodja

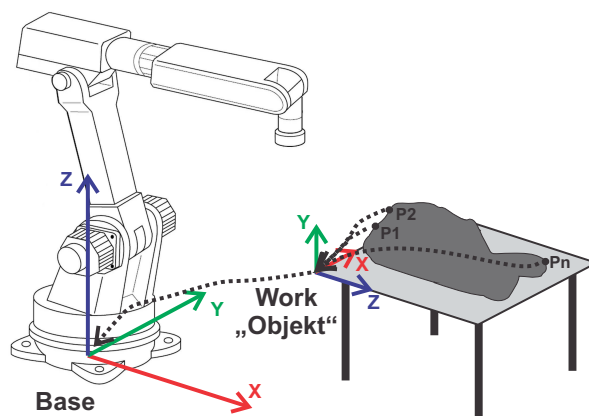
Izbiro **Tool** uporabimo, če želimo premikati (translacija in rotacija) vrh robot v koordinatnem sistemu orodja (Slika 5.12). V vrhu robota (zadnji sklep) je v osnovi def niran koordinatni sistem z imenom *Tool0*, katerega lahko izberemo v izbiri D na sliki 5.7. Vendar lahko uporabnik s postopkom kalibracije (presega okvir vaje) določi nov vrh robota, kar lahko vidimo na sliki 5.12 z imenom *Orodje*. V postopku kalibracije uporabnik določi ime novemu koordinatnemu sistemu orodja in to ime se pojavi v izbiri D na sliki 5.7. Če uporabnik izbere to novo ime (*Orodje* namesto *Tool0*) in ima izbran način premikanja robota **Tool**, potem se vrh robot premika glede na osi novega koordinatnega sistema na vrhu robota z imenom *Orodje*.



Slika 5.13. Lega ciljnih točk vrha robota def nirana v baznem koordinatnem sistemu

Izbiro **Work** uporabimo, če želimo def nirati koordinatni sistem, ki pripada delovnemu objektu. Najlažje je pomen orisati na primeru.

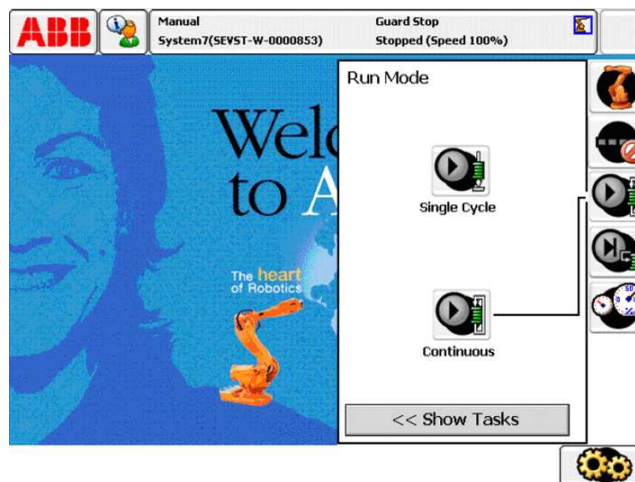
Na sliki 5.13 je robot s svojim baznim koordinatnim sistemom. V delovnem prostoru robota je postavljena miza in na mizo pritrjen delovni objekt. Naloga robota je, da z vrhom sledi robu objekta. Najprej postavimo vrh robota v začetno točko ( $P1$ ) in shranimo trenutno lego vrha robota. Nato robot premaknemo v naslednjo željeno lego in jo shranimo kot točko  $P2$ . Tako nadaljujemo do točke  $Pn$ . Če uporabnik ne določi drugače, so vse shranjene točke definirane v baznem koordinatnem sistemu robota. Nevarnost takega načina dela je, da se miza premakne in z njo tudi delovni objekt. To povzroči, da shranjene lege ne odražajo več dejanskega stanja, razen v primeru ponovne postavitve mize v izhodiščno lego. V nasprotnem primeru je potrebno prav vse shranjene točke shraniti ponovno, kar pa je velika izguba časa.



Slika 5.14: Uporaba koordinatnega sistema, ki pripada objektu

Na sliki 5.14 je robot s svojim baznim koordinatnim sistemom. V delovnem prostoru robota je postavljena miza in na mizo pritrjen delovni objekt. Naloga robota je, da z vrhom sledi robu objekta. S postopkom kalibracije definiramo nov koordinatni sistem z imenom *Objekt*, katerega ime se pojavi v izbiri E na sliki 5.7. Lega koordinatnega sistema *Objekt* je definirana v baznem koordinatnem sistemu robota. Sledi definiranje ciljnih točk vrha robota ( $P1$ ,  $P2$ , ...  $Pn$ ), vendar preden shranimo prvo, z izbiro E na sliki 5.7, definiramo, da so vse lege vrha robota definirane glede na nov koordinatni sistem *Objekt*. V primeru, da se miza zatem premakne, je potrebno na novo definirati samo koordinatni sistem, ki pripada objektu in ga poimenovati *Objekt*.

### Določanje načina izvajanja programa

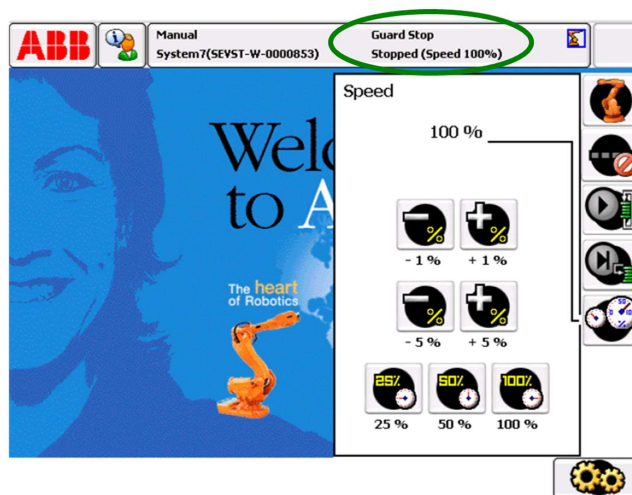


Slika 5.15: Izbira števila ponovitev programa

**Single Cycle** → Program se izvede samo enkrat.

**Continuous** → Program se izvaja brez ustavljanja.

### Določanje hitrosti gibanja robota med izvajanjem programa

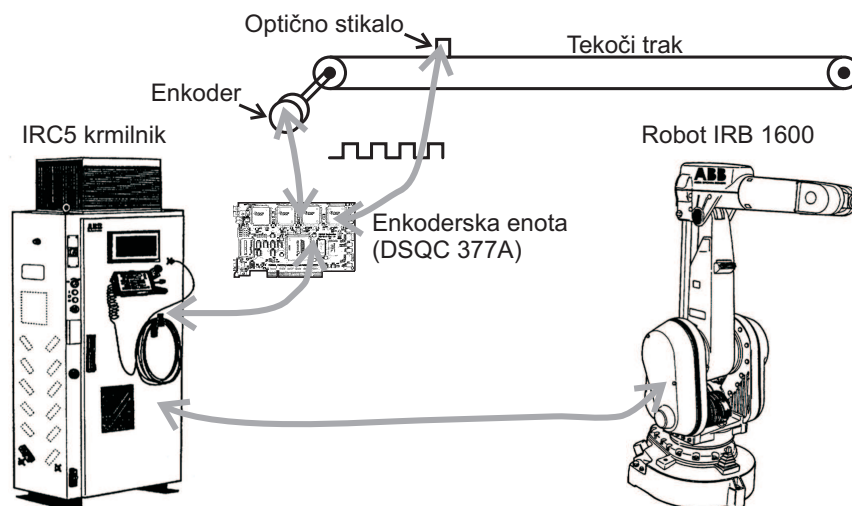


Slika 5.16: Izbira hitrosti gibanja robota

Nastavitve vpliva na hitrosti, ki so nastavljene v programu kot parametri ukazov za gibanje. Če je v programu nastavljena hitrost 100 mm/s in izbrana omejitev hitrosti 10 %, potem se robot premika s hitrostjo 10 mm/s.

## 5.4 Tekoči trak

### 5.4.1 Povezava z robotskim krmilnikom

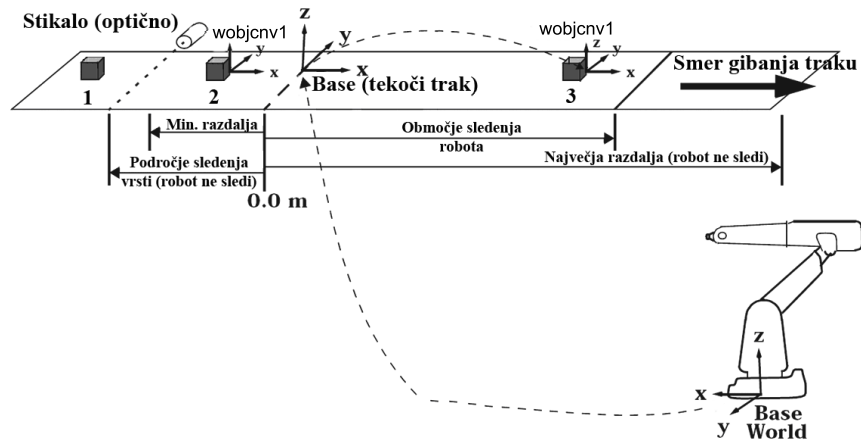


Slika 5.17: Sistem robot in tekoči trak

Srce sistema je krmilnik IRC5, na katerega je priključen robotski mehanizem IRB 1600. Kot zunanji sistem nastopa tekoči trak, ki je opremljen z enkoderjem in optičnim stikalom. Da je krmilnik in posledično robotski mehanizem zmožen slediti objektom na tekočem traku, je v krmilnik vgrajena namenska kartica, imenovana enkoderska enota z oznako DSQC 377A. Nanjo sta priključena tako enkoder kot tudi optično stikalo tekočega traku. Enkoder s svojim izhodom, v obliki dveh fazno zamaknjenih pravokotnih impulzov, omogoča, da enkoderska enota izračuna pozicijo in določi smer gibanja objekta na tekočem traku (uporabljeni tekoči trak ima le eno smer vrtenja). Optično stikalo pa enkoderski enoti sporoča, da je objekt prekinil žarek in je prisoten na tekočem traku. Na tekočem traku je lahko do 255 objektov in prav vsem enkoderska enota lahko sledi, vendar robotski mehanizem lahko dela le s prvim objektom v vrsti.

### 5.4.2 Definiranje lege tekočega traku in relacij med koordinatnimi sistemi

Za namene opravljanja laboratorijskih vaj so vsi parametri tekočega traku že definirani. Mednje štejemo vse razdalje (Slika 5.18) in tudi podane koordinatne sisteme. Takoj za optičnim senzorjem je območje, kjer enkoderska enota sledi objektu, vendar pa krmilnik ne dovoli robotu manipulacije z objektom.



Slika 5.18: Def niranje tekočega traku in relacije med koordinatnimi sistemi

Ker tekoči trak predstavlja zunanjo robotsko enoto, ima tudi tekoči trak svoj bazni koordinatni sistem. Na sliki 5.18 je imenovan **Base (tekoči trak)**. Glede na označbe na sliki 5.18 enkoderska enota za objekt št. 1 ne ve, da se nahaja na tekočem traku (v tem trenutku objekt št. 2 in 3 ne obstajata). Ko objekt prekine žarek optičnega stikala, mu enkoderska enota prične slediti oz. pozna lego koordinatnega sistema objekta (imenovan **wobjcnv1**), kar na sliki 5.18 ponazarja objekt št. 2. Ta lega je def nirana s signalom optičnega enkoderja, kar pomeni def niranje lege glede na bazni koordinatni sistem tekočega traku. Slednje najlažje razberemo z opazovanjem objekta št. 3.

Lega objekta št. 3 je tako def nirana glede na bazni koordinatni sistem tekočega traku (**Base (tekoči trak)**), ta pa je def niran glede na bazni koordinatni sistem robota (**Base**). Krmilnik lahko v vsakem trenutku preračuna lego koordinatnega sistema objekta (**wobjcnv1**) št. 3 glede na bazni koordinatni sistem robota (**Base**).

Če hočemo, da robotski mehanizem sledi objektu, ki se premika s tekočim trakom, je potrebno lego (imenujmo jo P1) vrha robota (v našem primeru kar **Tool0**), def nirati glede na koordinatni sistem objekta na tekočem traku (**wobjcnv1**). V robotskem programu namreč def niramo, da naj se vrh robot (**Tool0**) postavi v shranjeno lego (P1), ki je def nirana glede na koordinatni sistem objekta na tekočem traku (**wobjcnv1**). Krmilnik ciklično preračunava ciljno lego (P1) vrha robota, ki je def nirana glede na koordinatni sistem objekta (**wobjcnv1**), glede na bazni koordinatni sistem robota (**Base**). Če se koordinatni sistem objekta (**wobjcnv1**) premika s tekočim trakom, krmilnik v vsakem ciklu preračuna drugo lego vrha robota (P1) glede na bazni koordinatni sistem robota (**Base**) in to izgleda kot sledenje vrha robota premikajočemu objektu na tekočem traku.

**Razlaga oznak na sliki 5.18:**

**Base (World)** → Bazni koordinatni sistem robota (v našem primeru je World k.s. istoležen z Base k.s.).

**Base (tekoči trak)** → Bazni koordinatni sistem tekočega traku (predhodno postavljen; upoštevana samo os X).

**Work** → Lokalni koordinatni sistem objekta na tekočem traku.

**Področje sledenja vrsti** → Razdalja def nira točko 0.0 m na tekočem traku. Def nirana je glede na položaj optičnega stikala. Znotraj tega področja enkoderska enota sledi vsem objektom, robot pa ne more z njimi manipulirati. Ker je področje v negativni smeri osi  $x$  osi baznega koordinatnega sistema tekočega traku, je koordinata  $x$  pozicije objekta v tem področju negativna.

**Območje sledenja robota** → Področje dela robotskega mehanizma.

**Največja razdalja** → Največja razdaja, kjer sistem še sledi objektu na tekočem traku, vendar robot s tem objektom ne more več manipulirati. Izven tega področja objekt avtomatsko izgubi sledenje. Razdalja ima smisel, ko ima tekoči trak možnost vrtenja v obe smeri.

**Najmanjša razdalja** → Najmanjša razdaja, kjer sistem še sledi objektu na tekočem traku, vendar robot s tem objektom ne more več manipulirati. Izven tega področja objekt avtomatsko izgubi sledenje. Razdalja ima smisel, ko ima trak možnost vrtenja v obe smeri.

**Objekt št. 1** → Objekt še ni prekinil optičnega stikala, zato mu enkoderska enota še ne sledi.

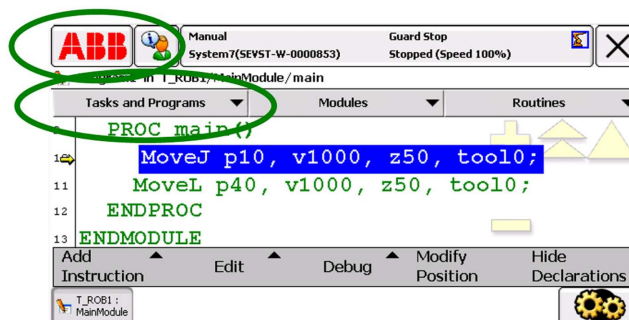
**Objekt št. 2** → Objekt je v področju sledenja, vendar izven področja sledenja robota. Če na tekočem traku ni objekta št. 3, potem robot čaka, da objekt št. 2 preide mejo 0.0 m. Zatem sledi premikanju objekta št. 2.

**Objekt št. 3** → Pozicija objekta na tekočem traku, preračunano glede na bazni koordinatni sistem robota, je znana in robot lahko sledi objektu.

## 5.5 Kreiranje, pisanje, urejanje in zaganjanje programov

### Kreiranje novega programa

Pritisnemo gumb **ABB** (levo zgoraj) in izberemo menu **Program Editor**. Odpre se okno s slike 5.19 in izberemo **Task and Programs**.

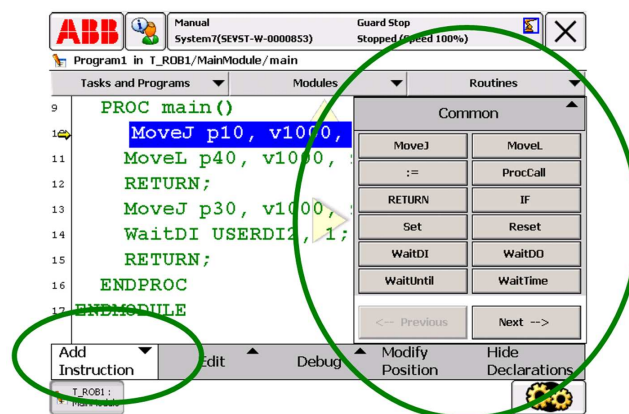


Slika 5.19: Postopek za odpiranje novega programa

V novem oknu izberemo **File** → **New Program...** in nato pritisnemo gumb **Don't Save**. Vpišemo želeno ime programa in ga potrdimo z gumbom **OK**. Odpre se okno s prazno funkcijo `main()`, znotraj katere dodajamo ukaze.

### Dodajanje ukazov v program

S pritiskom na gumb **Add Instruction** se na desni strani zaslona odpre nabor ukazov, ki so razporejeni v skupine. Na sliki 5.20 so prikazani najpogosteje uporabljeni ukazi (**Common**). **Do drugih skupin prihajamo tako, da kliknemo v polje Common**. Nov ukaz je dodan za trenutno izbrano vrstico.

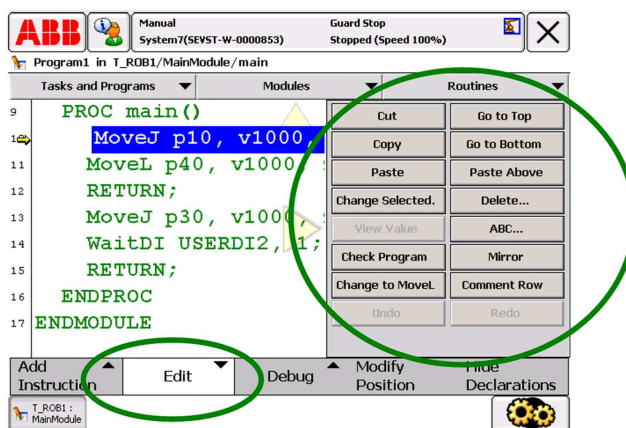


Slika 5.20: Dodajanje ukazov v program



### Manipuliranje in popravljanje vrstic programa

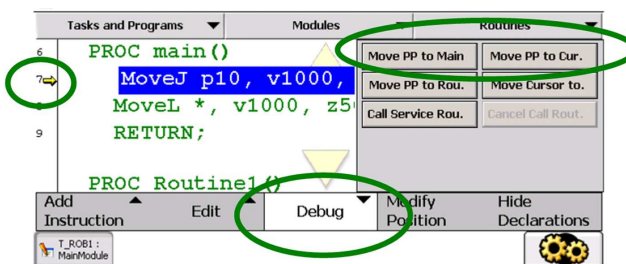
Do ukazov za popravljanje oz. manipulacijo vrstic pridemo s pritiskom gumba **Edit**. Desno se odpre nabor ukazov za manipulacijo. Operiramo z vrstico, ki je trenutno izbrana. *Posamezne parametre programskih ukazov spreminjamo s klikom na parameter ukaza, recimo v1500, v vrstici MoveJ p20, v1500.*



Slika 5.21: Izbire za manipuliranje in popravljanje vrstic programa

### Pozicioniranje kazalca v programu

Za zagon programa ni dovolj označena vrstica, ampak se izvrševanje programa prične od vrstice 5.22, ki je označena s kazalcem. Ob kreiranju programa kazalec še ni postavljen, zato ga je potrebno v program še postaviti. To storimo s spodaj opisanimi izbirama.



Slika 5.22: Pozicioniranje kazalca v programu

**Move PP\* to Main** → Kazalec postavimo na začetek funkcije main().

**Move PP\* to Cur.** → Kazalec postavimo na trenutno izbrano vrstico.

\* PP pomeni Program Pointer.

### 5.5.1 Koraki za izvedbo vaje z vzorčnim programom

- 1) Kreirajte nov program s poljubnim imenom in napišite program.
- 2) Komentarjev ne prepisujte, jih pa skrbno preberite. **Program in poudarjene besede so mišljene kot namig!**

```
Proc main()
  'Spremljanje konfiguracije med linearnimi gibi izključimo
  ConFL\Off; <nabor ukazov Settings + klik na ukaz + Optional
  Argument → Use \Off>
  'Aktiviranje zunanje mehanske enote CNV1 (tekoči trak)
  ActUnit CNV1; <nabor ukazov Motion & Proc.>
  'Zagon tekočega traku - ukaz Set postavi vrednost
  'spremenljivke/izhoda (doTekociTrak) na visoki nivo)
  'Digitalni izhod določimo v izbiri Expression.
  Set doTekociTrak; <nabor ukazov Common>
```

- 3) Zaženite do sedaj napisan program, ki mora pognati tekoči trak.





S kazalcem (PP) se postavite na prvo vrstico programa.

**Pazite na izbrano hitrost (25 %) gibanja robota med izvajanjem programa (Slika 5.16)!**

Pritisnite in držite večjo tipko na zadnji strani učne enote, pritisnite tipko za štart programa (glejte spodaj) in pritisnite ter držite še majhen gumb na zadnji strani učne enote.



Slika 5.23: Tipke na ročni učni enoti za preverjanje gibanja robota po programu

-  → Izvajanje programa od postavljenega kazalca do konca.
-  → Izvajanje programa koračno od trenutne pozicije kazalca nazaj.
-  → Izvajanje programa koračno od trenutne pozicije kazalca naprej.
-  → Zaustavitev izvajanja programa.

- 4) Nadaljujte s pisanjem programa. Pri prvem ukazu MoveL spremenite hitrost iz  $v1000$  na  $v300$ , parameter  $z50$  na *fine* in poimenujte ime trenutne shranjene lege kot *p10*, saj označba  $z^*$  uporabniku premalo pove. Parametre spreminjate z dvojnimi klikom na posamezen parameter.

```
'Premik nad kocke s hitrostjo 300 mm/s, popolnim ujemanjem v  
'točki in z uporabo orodja 0  
MoveL p10, v300, fine, tool0; <nabor ukazov Common>
```



```
'Vrednost spremenljivke c1RemAllPObj postavimo na 1, da  
'počistimo morebitno vrsto na traku  
Set c1RemAllPObj; <nabor ukazov Common>
```

```
'Krajša pavza  
WaitTime 0.2; <nabor ukazov Common>
```

```
'Vrednost spremenljivke c1RemAllPObj postavimo nazaj na 0  
Reset c1RemAllPObj; <nabor ukazov Common>
```

```
'Preventivno odprtje prijemala  
Reset doPrijemalo; <uporabite Reset in WaitTime>
```

```
'Krajša pavza  
WaitTime 0.2; <nabor ukazov Common>
```

```
'Prijemalo nad kocko 1, njeno prijetje in odlaganje na trak  
Kocka1NaTrak; <uporabite MoveL, WaitTime, Set in Reset>
```



Prijemalo nad  
kocko

Pozicija  
prijetja kocke

Prijetje kocke

Dvig kocke

Zaradi zakasnitve, ki jo ima pnevmatika oz. prijemalo, je potrebno za ukaz za stisk prijemala (*Set doPrijemalo;*) dodati krajšo pavzo (*Wait-Time 0.2;*). V nasprotnem primeru robot že dviga prijemalo, vendar prijemalo še ni povsem stisnjeno in tako kocka ni prijeta.



Prenos kocke nad tekoči trak (pred optični senzor)

Prenos kocke tik nad tekoči trak (ni dotika)

Spust kocke na tekoči trak

Dvig prijemala

'Prijemalo postavimo v področje nad senzorjem.  
 MoveL **pxx**, v300, fine, tool0; <nabor ukazov Common>

----- DELO S TEKOČIM TRAKOM -----

'Robot čaka, da prvi objekt v vrsti na tekočem traku preide 0.0 m.

WaitWObj wobjcnv1; <nabor ukazov Motion Adv.>

'Prvi objekt je v območju sledenja robota, zato lahko kocko primete. Upoštevajte navodila za delo z objekti na tekočem traku med sledenjem (podpoglavje 5.6).

**KockaSTraku;** <uporabite MoveL, WaitTime, Set in Reset>



'Prijeti objekt je potrebno izključiti iz sledenja.

DropWObj wobjcnv1; <nabor ukazov Motion Adv.>

----- DELO S TEKOČIM TRAKOM ZAKLJUČIMO -----

'Odnesemo kocko na izhodiščno pozicijo. Pazimo, da ne trčimo v stojalo.

**Začetna pozicija** <MoveL, WaitTime, Set in Reset>

'Ustavimo tekoči trak.

Reset doTekociTrak; <nabor ukazov Common>

'Deaktiviranje zunanje mehanske enote CNV1 (tekoči trak).

DeactUnit CNV1; <nabor ukazov Motion & Proc.>

## OPOZORILO!

Pred zagonom celotnega programa pri neomejenih hitrostih je potrebno zagotoviti, da v delovnem prostoru ni osebe ali predmeta, s katerim bi lahko robot trčil.

## 5.6 Koraki za učenje koordiniranega dela robota in tekočega traku

- 1) Objekt z robotom odložite na tekoči trak, ki ga v primeru delovanja ročno ustavite (Slika 5.6). *Trak naj ne teče!*
- 2) Prijemalo naj bo postavljeno nad senzor. *Trak naj ne teče!*
- 3) Vključite tekoči trak, da objekt na tekočem traku preide točko 0.0 m oz. preide v področje sledenja robota. **Trak ustavite!** Sedaj robotski krmilnik ve, kje na tekočem traku se nahaja objekt oziroma njegov koordinatni sistem (wobjcnv1). Znana je tudi lega orodja (tool0) oz. prijemala. Za sledenja vrha robota, premikajočemu se objektu na tekočem traku, je potrebno ročno nastaviti glede na kateri koordinatni sistem naj se trenutna lega prijemala shrani. Zato je potrebno ročno spremeniti koordinatni sistem **iz wobj0 v wobjcnv1** (Slika 5.7).
- 4) Robotsko prijemalo postavite nad kocko in shranite točko oz. dodate ukaz za premik robota. *V programu med sledenjem lahko uporabite samo ukaza MoveL in MoveC.*  
Primer: `MoveL pxx, v300, fine, tool0\WObj:=wobjcnv1`
- 5) Robotsko prijemalo spustite v področje prijema kocke in shranite točko oz. dodate ukaz za premik robota. *V programu med sledenjem lahko uporabite samo ukaza MoveL in MoveC.*  
Primer: `MoveL pxx, v300, fine, tool0\WObj:=wobjcnv1`
- 6) Koordinirano gibanje robota in tekočega traku je potrebno zaključiti z gibom v točko, ki ni definirana glede na koordinatni sistem objekta (wobjcnv1) na tekočem traku, ampak glede na bazni koordinatnem sistemu robota (Base). Kocko dvignete nad tekoči trak, ročno spremenite koordinatni sistem v katerem bo lega prijemala shranjena **iz wobjcnv1 v wobj0** (Slika 5.7) in shranite točko oz. ukaz za premik.  
Primer: `MoveL pxx, v300, fine, tool0`



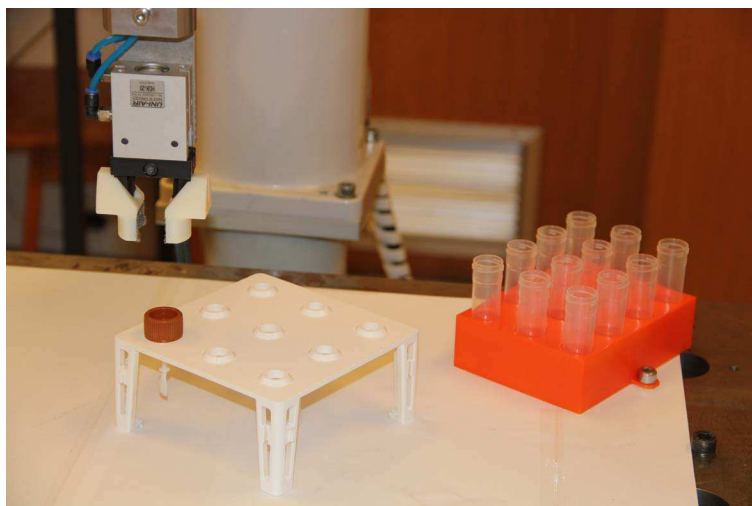
## Poglavje 6

# MANIPULACIJA OBJEKTOV Z ROBOTOM EPSON E2S651 IN ROBOTSKIM VIDOM

### 6.1 Cilj vaje

#### 1. del

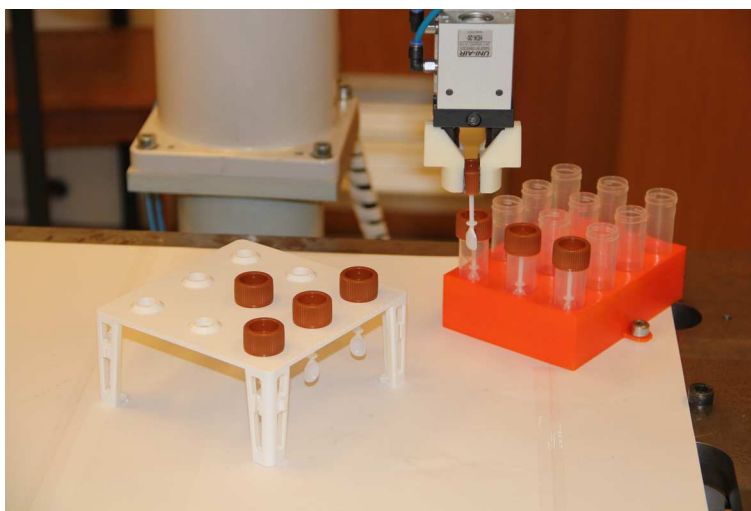
Na mizico v delovnem prostoru robota postavite pokrovček in ga z robotom oz. robotskim prijemalom primite. Prijetega odnesite nad poljubno ustje posodice in ga privijte. Po krajšem premoru ga odvijte in odnesite na začetno pozicijo. Tu ga spustite, robot pa umaknete. Začetno postavitvev kaže slika 6.1.



Slika 6.1: Začetna postavitvev v delovnem prostoru robota za 1. del naloge

## **2. del**

Z uporabo robotskega vida in robota Epson E2S651 določite pozicijo poljubno postavljenih pokrovčkov na mizici v delovnem prostoru robota (Slika 6.2) in jih privijte na ustja posodic. Pri nalogi je potrebno izvesti postopek kalibracije s katero definirate lego koordinatnega sistema kamere v referenčnem koordinatnem sistemu robota. Za izračun potrebnih transformacij uporabite okolje Matlab, za gibanje robota pa razvojno okolje Epson RC+.



Slika 6.2. Postavitev mizice s pokrovčki v delovnem prostoru robota za 2. del naloge

## **6.2 Lastnosti sistema**

Robot Epson E2S651 je robot tipa SCARA srednje velikosti. Njegove najpomembnejše lastnosti kaže tabela na sliki 6.3.

Robot med delom v industrijskem proizvodnem procesu je prikazan na sliki 6.4, kjer je uporabljen kot nosilec laserskega triangulacijskega merilnika razdalje za merjenje dimenzij ulitkov iz sive litine. Na desni sliki 6.5 pa je prikazana majhna kontrolna plošča na samem robotu, kjer najdemo priključke za zrak in električne signale ter tudi gumb za sprostitev zavore tretjega sklepa.

## **6.3 Programsko okolje Epson RC+**

Krmilnik robota je zasnovan na industrijskem osebнем računalniku. Osi robota vodijo neodvisni namenski procesorji, uporabniški vmesnik Epson RC+, ki predstavlja delovno okolje za vse robote Epson, pa teče v operacijskem sistemu Windows 2000. Vse komponente krmilnika, vključno s končnimi stopnjami ojačevalnikov, so v standardnem ohišju. Pisanje robotskih aplikacij

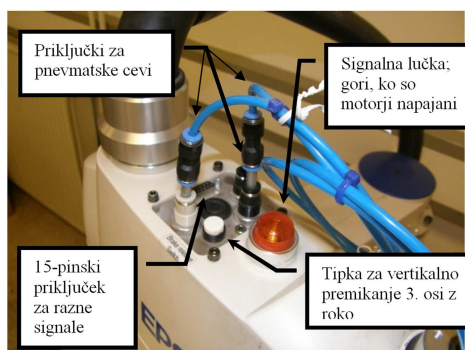


Model robota	E2S651	
Oblika	SCARA 4 Axis AC Servo	
Maksimalna obremenitev	5 kg	
Doseg osi	T1 + T2 Z U	650 mm 170/320 mm (čisti prostori: 150/300 mm) $\pm 360^\circ$
Hitrost	T1 + T2 (skupaj) Z U	<b>6300 mm/s</b> <b>1100 mm/s</b> <b>1870 °/s</b>
Ponovljivost	T1 + T2 Z U	<b><math>\pm 0.015</math> mm</b> <b><math>\pm 0.01</math> mm</b> <b><math>\pm 0.020^\circ</math></b>
Vztrajnostni moment (U os 5Kg )	0.120 kg·m <sup>2</sup>	
Masa	20 kg: (različni načini pritrditve: 22 kg)	
Uporabniška vodila	15 električnih, 3 pnevmatska, votla zadnja os	

Slika 6.3: Lastnosti robota Epson E2S651



Slika 6.4: Robot Epson E2S651



Slika 6.5: Kontrolna plošča na robotu

poteka v programskem jeziku SPEL (izpeljanka BASIC-a), ki nudi široke možnosti komunikacije in vključevanja v aplikacije, razvite s splošno namenskimi programskimi jeziki (vmesnik ActiveX, TCP/IP itd.). Izgled vmesnika RC+ prikazuje slika 6.6.

### 6.3.1 Osnove o pisanju programov

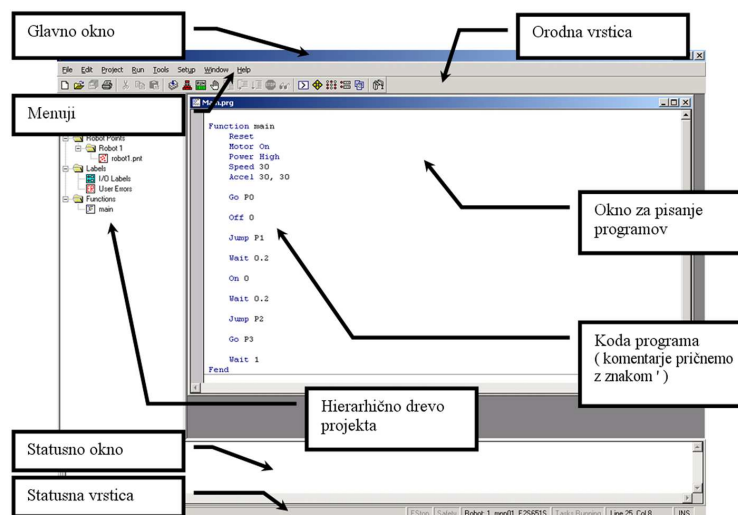
- 1) Za pisanje programa je potrebno odpreti nov projekt. To storimo tako, da izberemo menu **Project** → **New...** Pod **Select Project Folder** izberemo mapo *Projects\StudentskeVaje* in v polje **New Project Name** vpišemo poljubno ime projekta. Pazimo, da je vključena izbira *Create Main.prg*. Izbiro potrdimo z gumbom OK.

- 2) Odpremo pa lahko tudi že ustvarjeni projekt. To storimo tako, da izberemo menu **Project** → **Open...** Pod **Select Project to Open** izberemo projekt in kliknemo gumb **OK**.
- 3) Če področje za pisanje programov še ni odprto (okno z imenom Main.prg), ga odpremo tako, da v **hierarhičnem drevesu projekta dvakrat kliknemo na Main.prg**.
- 4) Za demonstracijo pisanja programa v prazno belo polje vpišemo:

```
Function Main
    On 0      ' Zapremo prijemalo.
    Wait 1.0  ' Počakamo 1 sekundo.
    Off 0     ' Odpremo prijemalo.
Fend
```

S tem smo ustvarili Main funkcijo. Med Function Main in Fend pa vpišujemo ukaze. *Za primer smo vnesli ukaz za zapiranje in odpiranje prijemala, vmes pa počakamo 1 sekundo.*

- 5) Napisan program zaženemo tako, da kliknemo ikono, prikazano na sliki 6.7, nato pa še gumb **Start main**.



Slika 6.6: Glavno okno programskega okolja Epson RC+

### 6.3.2 Vključitev pogonskih motorjev

Ob zagonu programskega okolja Epson RC+ pogonski motorji nimajo napajanja. Napajanje vključimo tako, da kliknemo na ikono, ki je označena na sliki 6.8, nato pa se odpre okno s slike 6.9.

#### Izbire oziroma gumbi v kontrolnem oknu

- **EStop: (OFF/ON)**  
→ Napis signalizira, ali je pritisnjena tipka (**črna škatla na mizi s kovinskim gumbom v sredini**) zasilnega izklopa.
- **Safe Guard: (OFF/ON)**  
→ Napis signalizira, ali je nekdo vstopil v varnostno kletko robota.
- izbira **Robot:** → Izberemo robot (v našem primeru je samo eden).
- gumb **Reset** → Resetiramo stanje po napaki ali pritisnjeni zasilni tipki.
- gumb **Help** → S pritiskom prikličemo pomoč.
- gumb **Close** → S pritiskom zapremo okno.

#### **Power**

- **LOW** → Pogonski motorji delujejo v načinu **nizke** moči.
- **HIGH** → Pogonski motorji delujejo v načinu **visoke** moči.

#### **Motor**

- **OFF** → Izključimo napajanje pogonskih motorjev (*v tem stanju lahko robot premikamo z roko*).
- **ON** → Vključimo napajanje pogonskih motorjev (*robot lahko premikamo le preko uporabniškega vmesnika*).

**V fazi programiranja in testiranja mora biti zaradi vaše varnosti in morebitnih poškodb opreme vedno vklopljena opcija Power LOW!**

Opcija Power High je namenjena doseganju polne dinamike robota v industrijskem okolju.

### 6.3.3 Upravljanje z digitalnimi vhodi in izhodi

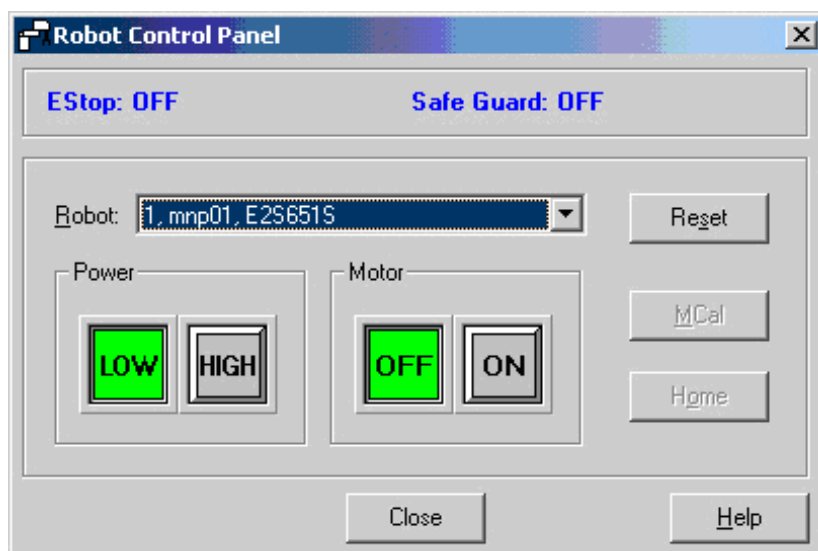
Krmilnik v osnovi omogoča spremljanje in upoštevanje 16 digitalnih vhodnih signalov in 16 digitalnih izhodnih signalov. Za namen izvajanja prijemanja



Slika 6.7: Zaganjanje napisanih programov



Slika 6.8: Orodna vrstica in ikona za odpiranje kontrolnega okna



Slika 6.9: Kontrolno okno robota

pokrovčkov je na robot nameščeno pnevmatsko prijemalo. Pnevmatiski cilindri prijemala krmilimo z monostabilnim elektropnevmatskim ventilom, ki ga krmilimo z enim digitalnim izhodom iz robotskega krmilnika. Po pritisku na ikono s slike 6.10 se odpre okno s slike 6.11.

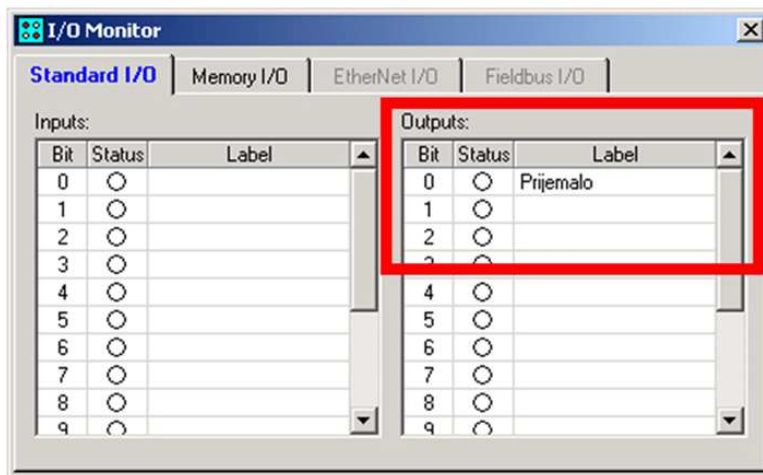


Slika 6.10. Orodna vrstica in ikona za odprtje okna upravljanja z digitalnimi vhodi in izhodi

### Okno za upravljanje z digitalnimi vhodi in izhodi

- **Standard I/O** → Zavihek, kjer upravljamo z digitalnimi vhodi in izhodi.
- **Inputs** → Digitalni vhodi (*biti od 0 do 15*).
- **Outputs** → Digitalni izhodi (*biti od 0 do 15*).
- **Memory I/O** → Zavihek za upravljanje z biti spomina (512 bitov), uporabljeni za komunikacijo med opravili (task) programa.

Prijemalo oziroma njegov elektropnevmatski ventil je priklopljen na **digitalni izhod številka 0**. Ta signal vklopimo oziroma izklopimo z **dvakratnim klikom na krogec v stolpcu Status** (Slika 6.11). Prijemalo je odprto, ko je krogec prazen, in zaprto ob polnem krogecu.



Slika 6.11: Okno za upravljanje z digitalnimi vhodi in izhodi

### 6.3.4 Okno za premikanje robota in učenje točk

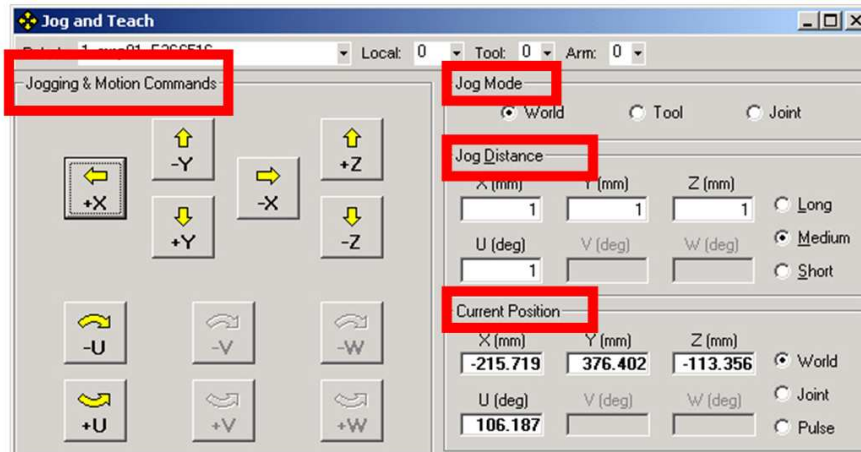
Robot premikamo na različne načine, spremljamo trenutno lego in shranjujemo lego vrha robota v oknu, katerega odpremo s pritiskom na gumb s slike 6.12.



Slika 6.12: Orodna vrstica in ikona za okno za premikanje robota in učenje točk

#### Ukazi za premikanje robota

- **Jog Mode** → Izberemo koordinatni sistem za premikanje robota (*World*, *Tool*, *Joint*).
- **Jog Distance** → Izberemo korak premika, kjer prednastavljene korake izberemo s pritiskom na ustrezen krogec (Long, Medium ali Short). V poljih lahko vrednosti tudi spreminjamo.
- **Current Position**
  - Izpisuje se trenutna lega izbranega orodja (npr. *Tool 0*) v referenčnem koordinatnem sistemu (izbira **World**).
  - Izpisujejo se trenutni koti v sklepih v stopinjah (izbira **Joint**).
  - Izpisujejo se trenutni pulzi enkoderjev v sklepih (izbira **Pulse**).
- **Jogging & Motion Commands**
  - S puščicami vodimo robot v izbrani smeri koordinatnega sistema (*Jog Mode = World ali Tool*) ali ga rotiramo po sklepih (*Jog Mode = Joint*).



Slika 6.13: Ukazi za premikanje robota

### Ukazi za učenje točk

- **Points** → Razdelek omogoča shranjevanje trenutnih leg vrha robota v datoteko (*Point File*); največje število točk v eni datoteki je 1000. V okno Point #: ročno vpišemo številko točke ali pa jo nastavimo z drsnikom spodaj. Trenutne vrednosti lege X, Y, Z, U in Local lahko poljubno spreminjamo. **Točko shranimo** s pritiskom na gumb **Teach P#**, **zbrišemo** pa jo z **Delete P#**.



Slika 6.14: Ukazi za učenje točk

Robot lahko v shranjene lege premikamo z izbirami na sliki 6.14, skrajno levo zgoraj. Hitrost premika nastavimo z izbiro *Speed*, način giba z izbiro *Motion Command*, gibanje pa štartamo z gumbom **Execute**.

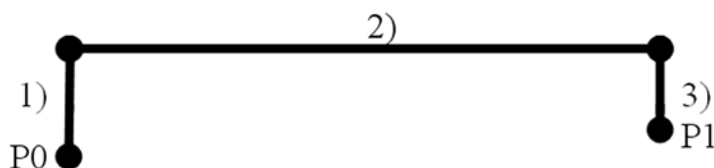
## 6.4 1. del naloge

V 1. delu naloge spoznate ukaze in postopek za prijem pokrovčka, privitje le-tega na ustje posodice, njegovo odvitje s posodice ter spust pokrovčka na začetno lego. Ta postopek je namreč večji del programa, ki je v 2. delu naloge že napisan in ga le zaženete.

### 6.4.1 Ukazi, potrebni za izvedbo naloge

**Jump** → premik vrha robota iz točke v točko (recimo iz P0 v P1), ki ga razdelimo na:

- 1) translacija navzgor z nespremenjeno orientacijo;
- 2) sinhrono premikanje sklepov, pri čemer se orientacija vrha robota spremeni, če je drugačna v ciljni točki;
- 3) translacija navzdol v ciljno lego z ustrezno orientacijo vrha robota.



Slika 6.15: Premikanje vrha robota pri uporabi ukaza Jump

**Go** → premik vrha robota iz točke v točko (P0 v P1) s sinhronim premikanjem sklepov. Sinhrono premikanje pomeni, da se vsi sklepi, katerih premik je potreben, začnejo premikati hkrati in tudi hkrati končajo.

*Primer: Za dosego končne lege se mora prvi segment zarotirati za 213°, tretji pa translirati za 8 mm. Rotacijo prvega sklepa robot opravlja z nastavljenno hitrostjo v programu (največji premik), hitrost translacije pa prilagodi času, ki je potreben za izvršitev rotacije prvega sklepa.*



Slika 6.16: Premikanje vrha robota, ko je uporabljen ukaz Go

**Motor P** → Parameter (P) **On** vključi napajanje motorjev, parameter **Off** pa napajanje izključi (*primer: Motor On*).

**Power P** → Nastavitev moči delovanja motorjev. Parameter (P) **Low** nastavi nizek režim, parameter **High** pa večji režim moči.

**Speed N** → Največja hitrost (N) gibanja v odstotkih (od 1 % do 100 %) glede na maksimum (*primer: Speed 40*).

**Accel N,M** → Največji pospešek (N) in pojemek (M) v odstotkih (od 1 % do 100 %) glede na maksimum (*primer: Accel 30, 10*).

**On N** → Dvig digitalnega signala na naslovu N (*primer: On 0*).

**Off N** → Spust digitalnega signala na naslovu N (*primer Off 0*).

**Wait N** → Ustavitev izvajanja programa za N sekund (*primer: Wait 1.3*).

#### 6.4.2 Koraki za izvedbo naloge

**Napišite program (primer spodaj) za premik robota iz točke P0 v P1, prijem pokrovčka in njegov prenos nad posodico v točki P2. Sledi privijanje v točko P3. Po krajši pavzi vrstni red nalog obrnite in odložite pokrovček na izhodiščno pozicijo, robot pa naj se vrne v izhodiščno točko P0. Vsaka operacija prijemanja in spuščanja pokrovčka zahteva kratke premore, zato uporabite ukaz Wait.**

```
Function main
  Motor On      ' Vključitev napajanja motorjev.
  Power Low     ' Nizka moč delovanja motorjev.
  Speed 20      ' Hitrost gibanja na 20% največje.
  Accel 20, 20  ' Pospešek in pojemek na 20% največjega.
  Off 0         ' Preventivno odprtje prijemala.
  Wait 0.2     ' Pavza 0.2 sekunde.

  Go P0        ' Premik robota v točko P0.
  Jump P1      ' Premik robota v točko P1.
  .
  .
  .

Fend
```



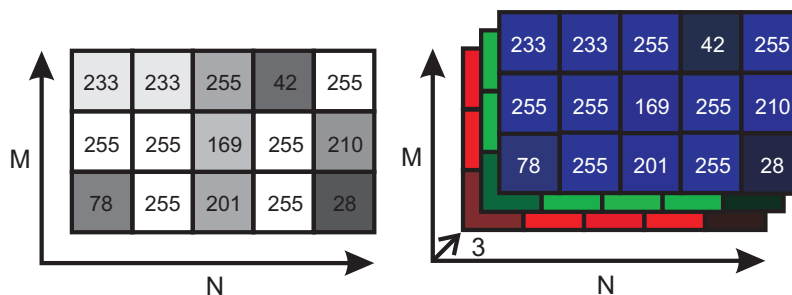
- 1) Ustvarite nov projekt (Projects\StudentskeVaje) in vpišite njegovo ime v polje, ki ga razvojno okolje ponudi.
- 2) V oknu za programiranje zapišete `Function main` in dve vrstici nižje še `End`.
- 3) Robot ne sme imeti vključenega napajanja motorjev (Slika 6.9).
- 4) Odprite okno Jog and Teach s klikom na ustrezno ikono (Slika 6.12).
- 5) Odprite okno I/O Monitor (Slika 6.11) in preverite, če je prijemalo odprto (izhodni naslov 0 ne sme biti izbran).
- 6) Na mizo v delovnem prostoru robota postavite pokrovček.
- 7) Robot ročno postavite v poljubno lego v delovnem prostoru robota.
- 8) V oknu Jog and Teach (Slika 6.14) shranite lego robota kot točko P0.
- 9) Robot z roko postavite tako, da lahko prime pokrovček (lega, ki ji sledi stisk prijemala). Za vertikalni premik tretjega segmenta sprostite zavoro s pritiskom na bel gumb (Slika 6.5).
- 10) V oknu Jog and Teach (Slika 6.14) shranite lego robota kot točko P1.
- 11) V oknu I/O Monitor (Slika 6.11) kliknite na naslov izhodnega digitalnega signala 0 za stisk prijemala.
- 12) Robot s prijetim pokrovčkom z roko premaknite tik nad ustje posodice. Za vertikalni premik tretjega segmenta sprostite zavoro s pritiskom na bel gumb (Slika 6.5).
- 13) V oknu Jog and Teach (Slika 6.14) shranite lego robota kot točko P2.
- 14) Zadnjo os robota s prijetim pokrovčkom z roko rotirajte (vrtite) in vertikalno translirajte (pomik navzdol), da privijete pokrovček. Za vertikalni premik tretjega segmenta sprostite zavoro s pritiskom na bel gumb (Slika 6.5).
- 15) V oknu Jog and Teach (Slika 6.14) shranite lego robota kot točko P3.
- 16) V oknu I/O Monitor (Slika 6.11) kliknite na naslov izhodnega digitalnega signala 0 za spust pokrovčka.
- 17) Robot z roko umaknite iznad ustja posodice.

## 6.5 Matlab in umetni vid

Programsko okolje Matlab s svojo široko paleto funkcij in orodij omogoča tudi učinkovito delo z umetnim vidom oziroma obdelavo slik ter vzorcev. Pri vaji sta pomembni predvsem naslednji orodji:

- 1) **Image Acquisition Toolbox** → skrbi za zajem slike in videa ter strojno podporo.
- 2) **Image Processing Toolbox** → obdelava slike, razpoznavanje vzorcev, transformacije, ugotavljanje najrazličnejših lastnosti slik itd.

Matlab je v osnovi namenjen za numerično računanje, še posebej manipulaciji matričnih funkcij. Sliko lahko predstavimo kot matriko. Sivinska je zapisana kot  $M \times N$  matrika (Slika 6.17(a)), barvna pa kot  $M \times N \times 3$  matrika, kjer tretja dimenzija predstavlja posamezno RGB komponento (Slika 6.17(b)).



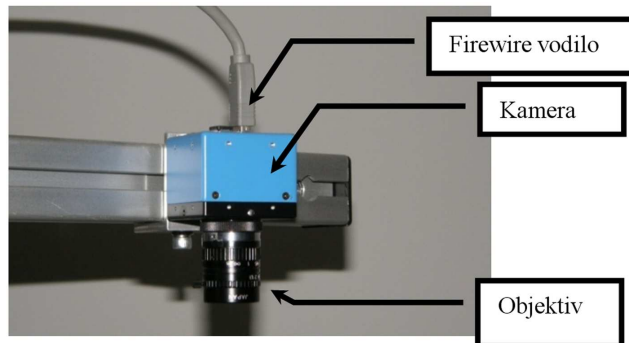
(a) Matrika sivinske slike dimenzije  $M \times N$  (b) Matrika barvne slike dimenzije  $M \times N \times 3$

Slika 6.17: Primera matrik sivinske in barvne slike

### 6.5.1 Video sistem

Karakteristike video sistema s slike 6.18 so:

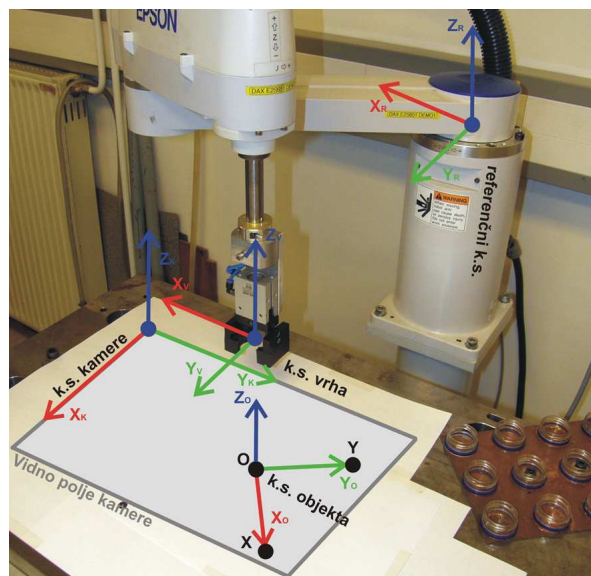
- kamera The Imaging source DMK 21AF04 s Firewire vodilom,
- slikovni senzor CCD velikosti  $640 \times 480$  slikovnih elementov,
- 256 sivinskih odtenkov,
- digitalen zajem in prenos slike,
- objektiv z 8 mm goriščne razdalje.



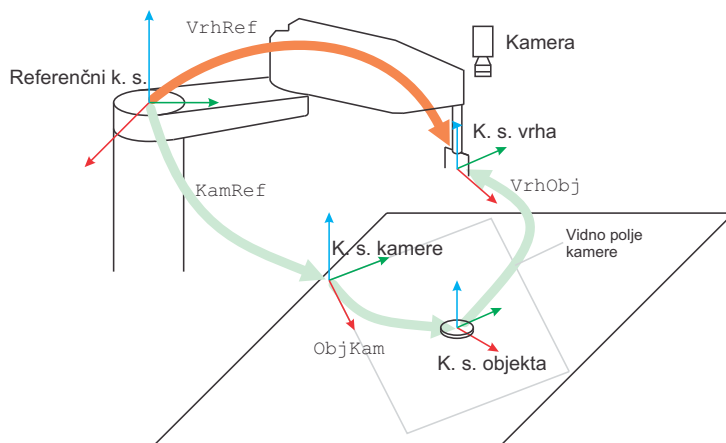
Slika 6.18: Video sistem nad mizo robota

## 6.5.2 2. del naloge

Na sliki 6.19 so predstavljeni vsi nastopajoči koordinatni sistemi. Najosnovnejši je **referenčni koordinatni sistem robota**, ki leži v prvem sklepu robota Epson. V verigi robota nastopa še **koordinatni sistem vrha robota**, ki je pripet na zadnji sklep robota. Ker v sistem hočemo vključiti še informacije dobljene s slike video kamere, v ogliščje njenega vidnega polja pripravimo **koordinatni sistem kamere**. V vidnem polju kamere bodo postavljeni objekti, ki imajo svojo lego (pozicijo in orientacijo). Zato zaradi splošnosti tudi objektu v vidnem polju kamere pripravimo **koordinatni sistem objekta**.



Slika 6.19: Postavitev koordinatnih sistemov



Slika 6.20: Transformacije med koordinatnimi sistemi

Na sliki 6.20 so prikazane transformacije med omenjenimi koordinatnimi sistemi. Najbolj očitna je transformacija, ki ponazarja **lego koordinatnega sistema vrha v referenčnem koordinatnem sistemu robota** in je odvisna od spremenljivk (koti in pozicije) v sklepih robota.

Lego vrha robota lahko izrazimo oz. izračunamo tudi drugače, vendar je potrebno poznati naslednje homogene transformacijske matrike:

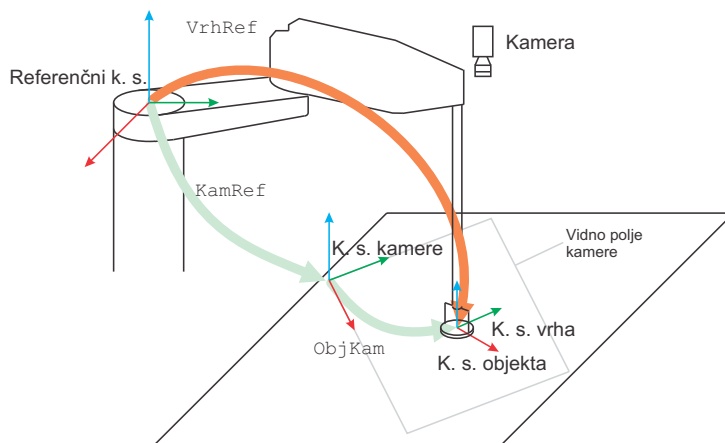
- lego koordinatnega sistema kamere v referenčnem koordinatnem sistemu robota (KamRef),
- lego koordinatnega sistema objekta v koordinatnem sistemu video kamere (ObjKam)
- lego vrha robota v koordinatnem sistemu objekta (VrhObj).

Omenjeno enakost zapišemo v obliki matrične enačbe:

$$\mathbf{VrhRef} = \mathbf{KamRef} \bullet \mathbf{ObjKam} \bullet \mathbf{VrhObj}$$

Razlaga okrajšav uporabljenih koordinatnih sistemov:

- VrhRef** → Lega vrha robota v referenčnem koordinatnem sistemu.  
*Lego odčitamo v okolju Epson RC+ (okno na sliki 6.14).*
- KamRef** → Lega koordinatnega sistema kamere v referenčnem koordinatnem sistemu robota.
- ObjKam** → Lega koordinatnega sistema objekta v koordinatnem sistemu robota.
- VrhObj** → Lega koordinatnega sistema vrha robota (prijemala) v koordinatnem sistemu objekta.



Slika 6.21: Transformacije med koordinatnimi sistemi pri prijemu objekta

Ko robot prime objekt, želimo poravnavo koordinatnega sistema vrha in koordinatnega sistema objekta. V tem primeru transformacija  $VrhObj$  izgine oz. matematično gledano matrika postane enotska matrika (po diagonali 1).

V tem primeru se enačba s prejšnje strani malenkostno poenostavi:

$$VrhRef = KamRef \bullet ObjKam \bullet I \quad (6.1)$$

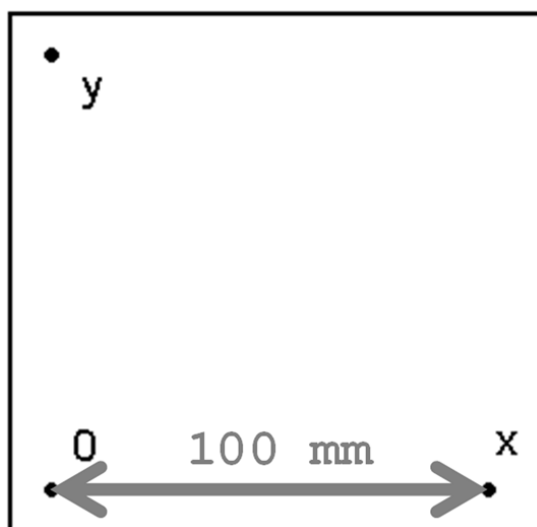
Enačba 6.1 opisuje lego koordinatnega sistema vrha robota v referenčnem koordinatnem sistemu ( $VrhRef$ ) ob prijemu objekta. Uporabna je v primeru, če poznamo lego koordinatnega sistema objekta v koordinatnem sistemu kamere ( $ObjKam$ ) in lego koordinatnega sistema kamere v referenčnem koordinatnem sistemu robota ( $KamRef$ ). Tako lahko izračunamo lego koordinatnega sistema objekta v referenčnem koordinatnem sistemu robota in v to lego lahko pošljemo vrh robota ( $VrhRef$ ).

**Če hočemo robot premakniti v neko lego ( $VrhRef$ ) v vidnem polju kamere in tako prijeti objekt ( $VrhObj = I$ ), je potrebno poznati obe transformaciji na desni strani enačbe ( $KamRef$  in  $ObjKam$ ). Vendar pa ob pričetku izvajanja naloge ni poznana ne ena ne druga lega.**

V nadaljevanju bomo pokazali, da je določitev  $ObjKam$  iz slike preprosto opravilo. Enako velja za določitev  $VrhRef$ . Na podlagi teh dveh določitev lahko iz enačbe 6.1 izrazimo kot neznanke  $KamRef$  in jo tako izračunamo. Šele nato lahko zgornjo enačbo uporabljamo za izračun  $VrhRef$ , ki premakne robot v lego za prijem objekta, katerega lego smo določili iz slike kamere.

### Določitev matrike ObjKam

Za določitev lege koordinatnega sistema objekta v koordinatnem sistemu kamere (**ObjKam**) uporabimo kalibracijsko ploščo s tremi črnimi pikami (Slika 6.22). Ena pika predstavlja koordinatno izhodišče  $O$ , ostali dve pa ležita na  $x$  in  $y$  osi koordinatnega sistema objekta.



Slika 6.22: Kalibracijska plošča s točkami, ki predstavljajo k.s. objekta.

**Plošča predstavlja objekt s koordinatnim sistemom in izhodiščem v točki  $O$ .**

**Razdalja med točkami  $O$  in  $X$  ter  $O$  in  $Y$  znaša 100 mm in jo uporabimo za pretvorbo števila slikovnih elementov v milimetre.**

### **Postopek za določitev matrike ObjKam**

Postopek za določanje matrike **ObjKam** je opisan v spodnjih točkah. Pri delu uporabite programsko okolje Matlab.

- 1) V okolju Matlab je potrebno izbrati delovno mapo ...\**Epson\_kamera**\ in v ukazni vrstici pognati Camera Calibration Toolbox. To storite z ukazom:

» **calib\_gui**

Toolbox je namenjen ugotavljanju parametrov kamere oz. uporabljenega objektiva (uporaba kalibracijskih vzorcev). Po potrditvi ukaza *calib\_gui* se

odpre okno, kjer kliknete gumb *Standard* (*all the images are stored in memory*). Postopek je že opravljen, zato kliknete gumb *Load* in s tem naložite rezultate zadnje kalibracije. Trenutno prikazano okno zaprete s klikom na gumb *Exit*.

- 2) V naslednjem koraku je potrebno zajeti sliko s kamere in z ravnokar naloženimi parametri odstraniti popačenje objektiva. To storite tako, da v ukazno vrstico okolja Matlab vpišete:

➤ `snapcalib`

Če je kamera pravilno priključena, se na zaslonu izpišejo parametri kamere, takoj zatem pa se pojavi okno z živo sliko. **Postavite kalibracijski list v vidno polje kamere, da so na sliki vidne vse tri točke. Sliko zajamete s pritiskom na tipko** . Slika se shrani kot `calib_image.tif`, opravi se transformacija odstranjevanja popačenja objektiva in popravljena slika se shrani kot `calib_image_rect.tif`.

- 3) Na zajeti in uprakovljeni bitni sliki je potrebno ročno pokazati, kje se pike koordinatnega sistema objekta ( $O, X, Y$ ) nahajajo. V ukazni vrstici okolja Matlab izvršite ukaz:

➤ `select_dots`

Na prikazani bitni sliki pokažete pike z dvojnimi klikom v okroglo področje posamezne pike. **Vrstni red dvojnih klikom mora biti sledeč: najprej dvojni klik na točko  $O$ , zatem dvojni klik na točko  $X$  in še dvojni klik na točko  $Y$ .** Pri izbiranju točk ni potrebna velika točnost, pomembno je le, da kliknete v območje pike, saj program sam izračuna središče. Po izbranih vseh treh točkah program središča točk obarva z rdečo piko. Če središča odstopajo preveč izven belega področja pik, postopek iz te točke ponovite.

Koordinate točk v koordinatnem sistemu kamere se shranijo v vektor  $\mathbf{g}$  (Enačba 6.2). **Vrednosti v vektorju  $\mathbf{g}$  so v enotah slikovni elementi, zato jih je potrebno naknadno pretvoriti v milimetre.**

$$\mathbf{g} = \begin{bmatrix} (O_x, O_y) \\ (X_x, X_y) \\ (Y_x, Y_y) \end{bmatrix} \quad (6.2)$$

- 4) V Matlab-u odprete datoteko (File→Open...) **zamaski.m** in jo dopolnite:

```
% Razdalja v slikovnih elementih med zajeto O in X točko.  
Ox = g(1,1);      % Izluščimo koordinato x točke O.
```

```

Oy = g(1,2);      % Izluščimo koordinato y točke O.
Xx = g(2,1);      % Izluščimo koordinato x točke X.
Xy = g(2,2);      % Izluščimo koordinato y točke X.

% Izračun števila slikovnih elementov med točkama O in X.
d = sqrt( ( Xx - Ox )^2 + (Xy - Oy )^2);

% Izračunamo faktor pretvorbe št. slikovnih elementov v
% 1 mm. Razdaljo 100 mm med pikami delimo s številom
% slikovnih elementov med točkami.
faktor = 100/d;

% Koordinate točk O,X,Y iz slikovnih elementov pretvorimo v
% metrične enote (v našem primeru v milimetre).
g_mm = g*faktor;
Ox_mm = g_mm(1,1); % Izluščimo koordinato x točke O v mm.
Oy_mm = g_mm(1,2); % Izluščimo koordinato y točke O v mm.
Xx_mm = g_mm(2,1); % Izluščimo koordinato x točke X v mm.
Xy_mm = g_mm(2,2); % Izluščimo koordinato y točke X v mm.

% Izračunamo kot zasuka koordinatnega sistema Objekta
% glede na k.s. Kamere (6.23).
fiOK = atan2( (Xy_mm - Oy_mm) , (Xx_mm - Ox_mm) );

% Zapišemo transformacijsko matriko ObjKam - translacija
% D(Ox_mm, Oy_mm) in rotacija okrog osi Z za kot fiOK.
ObjKam = [cos(fiOK), -sin(fiOK), 0, Ox_mm; sin(fiOK),
          cos(fiOK), 0, Oy_mm; 0, 0, 1, 0; 0, 0, 0, 1];

```

Zgornji zapis v matrični obliki izgleda tako:

$$\mathbf{ObjKam} = \begin{bmatrix} \cos(\mathbf{fiOK}) & -\sin(\mathbf{fiOK}) & \mathbf{0} & \mathbf{Ox\_mm} \\ \sin(\mathbf{fiOK}) & \cos(\mathbf{fiOK}) & \mathbf{0} & \mathbf{Oy\_mm} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \quad (6.3)$$

Enačba 6.3 predstavlja transformacijsko matriko **ObjKam** in opisuje lego objekta oz. njegovega koordinatnega sistema (list s pikami) v koordinatnem sistemu kamere (Slika 6.23).

### Določitev matrike KamRef

Da lahko s pomočjo robotskega vida z robotom prijemamo objekte v vidnem polju kamere in delovnem prostoru robota, je potrebno lego objektov definirati v referenčnem koordinatnem sistemu robota. Glede na enačbo 6.1 za izračun željene lege objekta v referenčnem koordinatnem sistemu robota, potrebujemo še **transformacijsko matriko KamRef**.

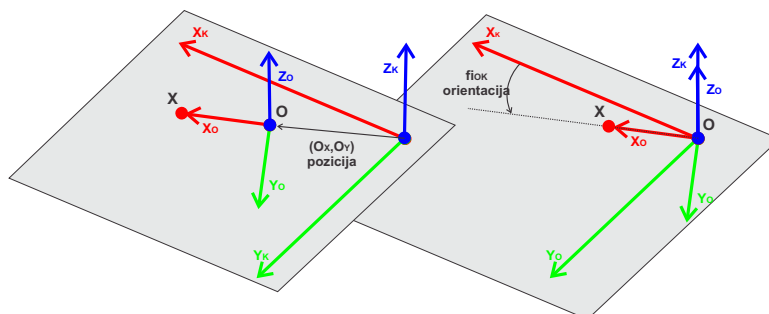


### Postopek za določitev matrice KamRef

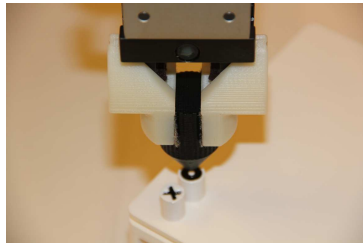
Lega objekta oziroma koordinatnega sistema objekta je zapisana v transformacijski matriki **ObjKam**. Za določitev lege koordinatnega sistema kamere v referenčnem koordinatnem sistemu robota ni mogoče najti postopka, ki bi lego določil, kot smo to storili za lego objekta v koordinatnem sistemu kamere. Izkaže se, da lahko postopek za določitev lege objekta v koordinatnem sistemu kamere, v rahlo spremenjeni obliki, uporabimo za določitev matrice **VrhRef**. Ta ob prijemu objekta predstavlja lego objekta v referenčnem koordinatnem sistemu robota. Na ta način lahko z izračunom določimo matriko **KamRef**, ki predstavlja lego kamere v referenčnem koordinatnem sistemu robota.

- 1) **V robotsko prijemalo primete konico** (Slika 6.24). Z robotom jo točno pozicionirajte glede na točki  $O$  in  $X$  koordinatnega sistema objekta na listu papirja (Slika 6.22). Pomembno je, da lista v času določanja transformacijskih matrik ne premikate! V okolju Epson RC+ v oknu Jog and Teach **preberete in si zapišete trenutno pozicijo točk  $O(O_x, O_y)$  in  $X(X_x, X_y)$  v World koordinatnem sistemu** (Slika 6.13).
- 2) Odčitana točka  $O$  predstavlja **pozicijo** koordinatnega sistema objekta v referenčnem koordinatnem sistemu robota (Slika 6.25, levo). **Orientacijo** koordinatnega sistema objekta v referenčnem koordinatnem sistemu robota pa določa kot (f OR) med osema  $X_R$  in  $X_O$  (Slika 6.25, desno).
- 3) V okolju Matlab je že odprta datoteka **zamaski.m**, zato nadaljujete:

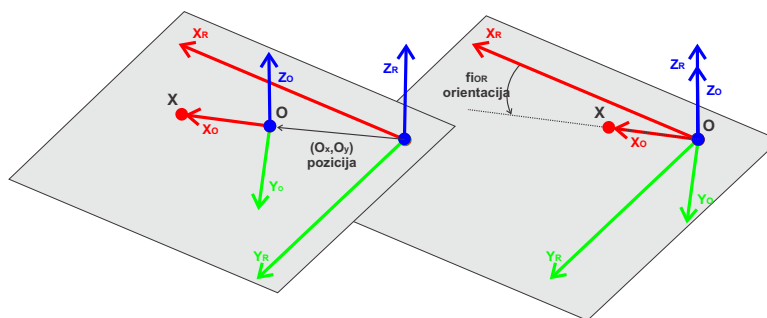
```
% V ustrezna polja vpišemo prebrane vrednosti koordinate
% O(Ox,Oy) in X(Xx,Xy).
Ox_r = vpišemo prebrano koordinato;
Oy_r = vpišemo prebrano koordinato;
Xx_r = vpišemo prebrano koordinato;
Xy_r = vpišemo prebrano koordinato;
```



Slika 6.23: Lega koordinatnega sistema objekta v koordinatnem sistemu kamere



Slika 6.24. V ustje robota nameščena konica, ki smo jo pozicionirali v središče točke na osi X



Slika 6.25. Lega koordinatnega sistema objekta v referenčnem koordinatnem sistemu robota

```
% Izračunamo kot zasuka koordinatnega sistema Objekta
% glede na referenčni k.s. (Slika 6.25, desno).
fiOR = atan2( (Xy_r - Oy_r) , (Xx_r - Ox_r) );

% Zapišemo transformacijsko matriko Vrh- translacija
% D(Ox_r, Oy_r) in rotacija okrog osi Z za kot fiOR.
VrhRef = [cos(fiOR), -sin(fiOR), 0, Ox_r; sin(fiOR), cos(fiOR), 0, Oy_r;
          0, 0, 1, 0; 0, 0, 0, 1];
```

Zapis v matrični obliki izgleda tako:

$$\mathbf{VrhRef} = \begin{bmatrix} \cos(\text{fiOR}) & -\sin(\text{fiOR}) & 0 & O_{x_r} \\ \sin(\text{fiOR}) & \cos(\text{fiOR}) & 0 & O_{y_r} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.4)$$

Enačba 6.4 predstavlja transformacijsko matriko **VrhRef** in opisuje lego objekta oziroma koordinatnega sistema objekta v referenčnem koordinatnem sistemu robota.

- 4) V tem trenutku sta določeni obe transformaciji **ObjKam** in **VrhRef**, zato lahko po spodnjem postopku izračunamo lego koordinatnega sistema kamere v referenčnem koordinatnem sistemu robota, ki jo predstavlja transformacijska matrika **KamRef**. Izhodišče je enačba 6.1.

$$\begin{aligned}\mathbf{VrhRef} &= \mathbf{KamRef} \bullet \mathbf{ObjKam} / z \text{ desne množimo z } (\mathbf{ObjKam})^{-1} \\ \mathbf{VrhRef} \bullet (\mathbf{ObjKam})^{-1} &= \mathbf{KamRef} \bullet \mathbf{ObjKam} \bullet (\mathbf{ObjKam})^{-1} \\ \mathbf{KamRef} &= \mathbf{VrhRef} \bullet (\mathbf{ObjKam})^{-1}\end{aligned}$$

V .m datoteko dopišete potrebne vrstice.

```
% Zapišemo izračun transformacije KamRef
KamRef = VrhRef * inv(ObjKam);

% Središča objektov na sliki pretvorimo v metrične enote
obj = centers * faktor;

% Določitev koordinat prepoznanih objektov v referenčnem
% koordinatnem sistemu. Ker imamo opraviti samo s pozicijami
% središč pokrovčkov, ni potrebno izvajati rotacij. Zato
% transformacijske matrike vsebujejo samo translacijski del.
for ii = 1:size(centers,1)
    ObjKam = [1,0,0,obj(ii,1); 0,1,0,obj(ii,2); 0,0,1,Z; 0,0,0,1];
    VrhRef(:, :, ii) = KamRef * ObjKam;
end

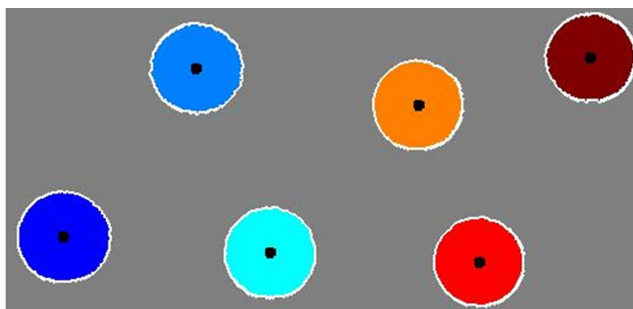
% Iz serije transformacijskih matrik Vrh izluščimo samo
% koordinate središč pokrovčkov v referenčnem k.s.
koord_zamaskov;
```

- 5) **Datoteko zamaski.m shranimo!**

### 6.5.3 Koraki za izvajanje naloge

- 1) **Vsi postopki določanja transformacijskih matrik **ObjKam** in **KamRef** ter vse vrstice programske kode v datoteki **zamaski.m** morajo biti opravljeni oziroma zapisani!**
- 2) **V vidno polje kamere postavite mizico z nekaj pokrovčki** (recimo 7), še prej pa odstranite kalibracijsko konico, umaknete robot iz vidnega področja kamere in odstranite list s koordinatnim sistemom objekta.
- 3) V Matlab ukazni vrstici zažete ukaz:  
`>> extobject`

Odpre se okno kamere z živo sliko za boljše pozicioniranje mizice s pokrovčki. Ko ste s pozicijo mizice s pokrovčki zadovoljni, pritisnete `[Enter]`. Čez čas se v ukazni vrstici okolja Matlab izpiše število razpoznanih objektov in izriše se slika 6.26. **Če niste zadovoljni z razpoznavo objektov, ta korak ponovite.**



Slika 6.26: Iz zajete slike razbrani pokrovčki ter označena njihova središča

- 4) V ukazni vrstici okolja Matlab poženete datoteko(*zamaski.m*).  
`>> zamaski [Enter]`
- 5) V Epson RC+ okolju v mapi Projects\ odprete že pripravljen projekt z imenom **Epson\_kamera**.

**Ob zagonu programa poskrbite, da robot deluje z nižjo močjo in njegova hitrost gibanja naj bo le 10% (prvi dve vrstici programa).**

```
#define dPower Low
#define dSpeedAccelProcent 10
```

- 6) S tipko `[F5]` projekt prevedete (Slika 6.7) in ga zaženete s klikom na gumb **Start tcpipcomm**.

## OPOZORILO!

Pred zagonom programa je potrebno zagotoviti, da v delovnem prostoru ni osebe ali predmeta, s katerim bi lahko robot trčil.

- 7) V ukazni vrstici okolja Matlab izvršite ukaz (štarta premikanje robota):

`>> zamaski_tcp [Enter]`

## Poglavje 7

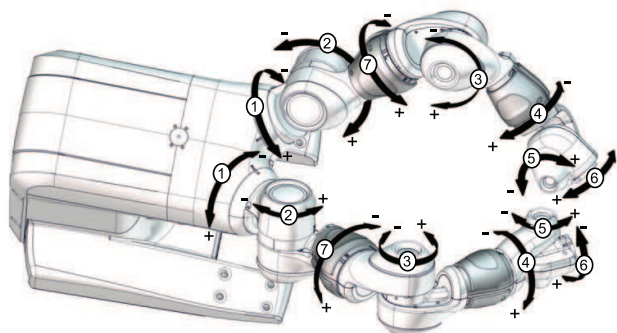
# SODELUJOČI ROBOT ABB YUMI IN ROBOTSKI VID

### 7.1 Cilj vaje

Pri tej vaji boste uporabili sodelujočega robota ABB YuMi z nameščenimi SmartGripper prijemali, ki imajo vgrajeno kamero Cognex In-Sight serije 7000. Z uporabo programskega paketa RobotStudio ter integrirane podpore za robotski vid boste robota naučili prepoznati lego objektov. Na podlagi prepoznane lege boste napisali program, ki bo izvedel ustrezno manipulacijo objektov. Pri tem si boste pomagali s kinestetičnem vodenjem robota (angl. *lead-through*).

### 7.2 Struktra sistema

Robot IRB 14000-0.5/0.5 (YuMi) proizvajalca ABB je predstavnik t.i. sodelujočih robotov. Tak tip robota je narejen za varno delo skupaj s človekom brez dodatnih varnostnih elementov (varnostnih ograj, svetlobnih zaves ...). Robot je dvoročni industrijski robot z integriranim krmilnikom IRC5. Posamezna roka antropomorfne oblike ima sedem prostostnih stopenj. Robot je namenjen manipulaciji z manjšimi objekti, kot je na primer sestavljanje elektronskih naprav. Osnovni podatki robota so podani v spodnji tabeli, na sliki 7.1 pa je predstavljena konfiguracija robota.



Slika 7.1: Konfiguracija robota IRB 14000-0.5/0.5

Tip	ABB IRB 14000-0.5/0.5	
Doseg posamezne roke	0.5 m	
Nosilnost posamezne roke	500 g	
Maksimalna hitrost vrha	1.5 m/s	
Ponovljivost pozicioniranja	$\pm 0.02$ mm	
Maksimalna hitrost	Os 1	180 °/s
	Os 2	180 °/s
	Os 3	180 °/s
	Os 4	400 °/s
	Os 5	400 °/s
	Os 6	400 °/s
	Os 7	180 °/s
Delovni prostor	Os 1	-168.5° / +168.5°
	Os 2	-143.5° / +43.5°
	Os 3	-123.5° / +80°
	Os 4	-290° / +290°
	Os 5	-88° / +138°
	Os 6	-229° / +229°
	Os 7	-168.5° / +168.5°
Teža	38 kg	
Zavore	1, 2, 3 in 7 os	

Posamezna roka je opremljena z integriranim prijemalom. Prijemalo je sestavljeno iz dveh prstov, ki jih poganjajo servo motorji. Dodan je še en pnevmatski sesek, s katerim se lahko preko vakuuma oziroma komprimiranega zraka manipulira z objekti. Prijemalo na levi roki ima dodatno integrirano kamero proizvajalca Cognex. Prijemalo je prikazano na sliki 7.2.



Slika 7.2: Robotsko prijemalo SmartGripper

Robota se programira z uporabo ročne učne naprave *FlexPendant* (slika 7.3). Naprava omogoča premikanje robota, upravljanje s prijemali, pisanje, popravljanje in poganjanje programov ter podobno. Ker je IRB 14000 deklariran kot sodelujoči robot za premikanje ni potrebno držati varnostne tipke na spodnji strani ročne učne naprave kot je to v navadi pri premikanju ostalih robotov proizvajalca ABB. Več o uporabi ročne učne naprave *FlexPendant* je razloženo v poglavju Industrijski robot ABB IRB 1600-7/145.



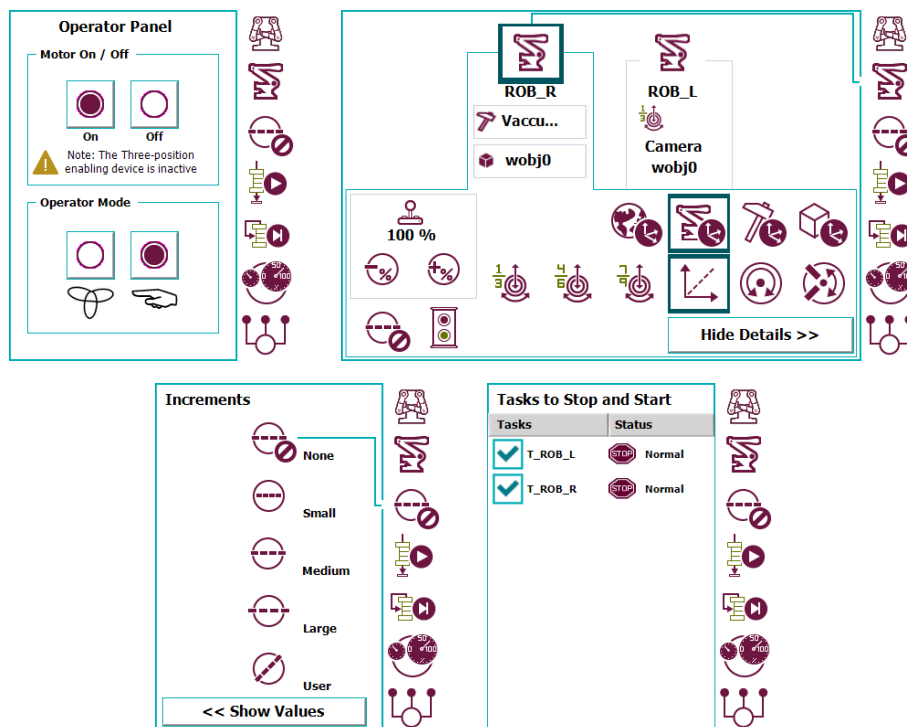
Slika 7.3: Ročna učna naprava *FlexPendant*

## 7.3 Dvoročna manipulacija

### 7.3.1 Ročno vodenje robota

Ročno vodenje je način vodenja robota s pomočjo krmilne palice. Robota lahko ročno vodite glede na položaje sklepov ali glede na različne koordinatne sisteme. Izbrano delovanje gibanja in/ali koordinatnega sistema določa način premikanja robota. V linearnem načinu gibanja se točka središča orodja (TCP - angl. *tool center point*) premika po ravni črti v prostoru, oz. v smeri osi izbranega koordinatnega sistema. V načinu gibanja “os-za-osjo” pa se premika do-

ločena os robota. Ker ima krmilna palica tri prostostne stopnje gibanja, lahko naenkrat spreminjate pozicijo le treh koordinat. Ker ima vsaka roka 7 prostostnih stopenj, lahko poleg premikanja vrha premikate tudi sam mehanizem, medtem ko se vrh ne premika.

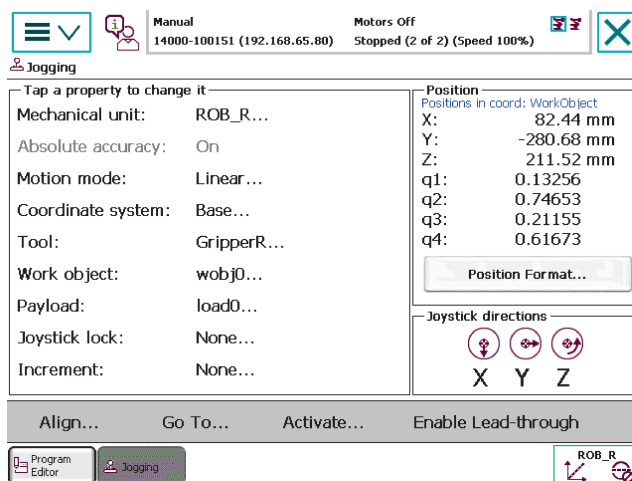


Slika 7.4. Hitri meni: Kontrolna plošča (zgoraj levo), izbira načina vodenja (zgoraj desno), inkrementalno vodenje (spodaj levo), izbira aktivnega robota (spodaj desno)

Hitri meni, ki je dostopen preko tipke v spodnjem desnem kotu, nudi osnovno funkcionalnost za premikanje robota. Prvi meni (slika 7.4 zgoraj levo) je kontrolna plošča, s katero prižigate/ugašate motorje, ter izbirate avtomatski oziroma ročni način izvajanja. Naslednji meni nam podaja izbiro načina vodenja (slika 7.4 zgoraj desno). Izberete lahko robotsko roko, ki jo želite premikati, izbirate, katere sklepe želite premikati in način premikanja vrha (spreminje pozicije, orientacije ali pa celotne konfiguracije). V tem meniju lahko tudi izberete uporabljeno orodje ter uporabniški koordinatni sistem. Naslednji meni določa hitrost premikanja: pri hitrostnem vodenju je hitrost premikanja proporcionalna odmiku krmilne palice, pri inkrementalnem vodenju pa je premikanje pogojeno z dolžino inkrementa. V naslednjih menijih izbirate način izvajanja programa (program se izvede enkrat ali pa se ponavlja), način premikanja po



programu, ter hitrosti, ki omejujejo gibanja robota med učenjem. V zadnjem meniju izberete, kateri robot je aktiven (slika 7.4 spodaj desno). Na podlagi te izbire morate definirati ustrezne programe; n.p.r. če imate izbrani obe roki, morate ustvariti dva programa, en program za posamezno roko, če pa imate izbrano samo levo roko, pa samo program za levo roko.



Slika 7.5. Meni za nastavljanje parametrov vodenja (*Jogging*): izbira robotske roke, načina premikanja, koordinatnih sistemov, orodja, delovnih objektov; meni omogoča tudi izbiro vodenja robota z roko (*Enable Lead-through*).

Robot ABB IRB 14000 omogoča tudi vodenje robota z roko (angl. *Lead through*). V tem načinu lahko robotsko roko primemo ter jo postavimo v ustrezno lego; pri tem za premikanje ni potrebno uporabiti krmilne palice. Robotski krmilnik na podlagi modela mehanizma preračunava ustrezne navore za motorje, ki kompenzirajo vpliv vztrajnosti, coriolisovih in centripetalnih sil, statičnega in dinamičnega trenja ter gravitacije. Vsak dodaten navor na motorje, kot je na primer potisk operaterja, predstavlja dodatno komponento navora, ki jo mora robotski mehanizem uestezno izničiti. Ta način vodenja se vklopi z izbiro opcije *Enable Lead-through* preko menija *ABB > Jogging*. Preden se robotski program požene, je potrebno ta način vodenja izklopiti (*ABB > Jogging > Disable Lead-through*).

### 7.3.2 Kreiranje delovnega objekta

Delovni objekt vsebuje informacijo o novem uporabniškem koordinatnem sistemu. Ta se uporablja za poenostavitev programiranja, ko urejate programe, zaradi prenavstavitve določenih opravil, procesov, objektov, itd. Ustvarite jih za poenostavitev premikanja vzdolž površin objektov. Ker jih lahko ustvarite več,

je potrebno vedno izbrati tistega, glede na katerega izvajate gibanje. Aktiven koordinatni sistem izberete v meniju *ABB > Jogging > Workobject*.

Nov uporabniški koordinatni sistem def nirate z *ABB > Jogging > Workobject > New*. V novem oknu nastavite ime koordinatnega sistema (*Name*), doseg (*Scope*) nastavite na globalni, tip shranjevanja (*Storage Type*) pa nastavite na *persistent*. Ostale nastavitve pustite nespremenjene.

Nov koordinatni sistem je potrebno def nirati. To storite z dostopom *ABB > Jogging > Workobject > Edit > Define*. Nato izberete želeno metodo kalibracije: *3 Points*. Robota ročno vodite v tri določitvene točke, pri čemer prva točka določa izhodišče novega koordinatnega sistema, druga točka smer osi x, pravokotnica na os x, ki teče skozi tretjo točko pa določa os y. Ko robota postavite v ustrezno točko, potrdite pozicijo z *Modify Position*. Med samim def niranje ni priporočljivo spreminjati orientacije prijemala. Določitev kalibracije zaključite s potrditvijo *OK*.

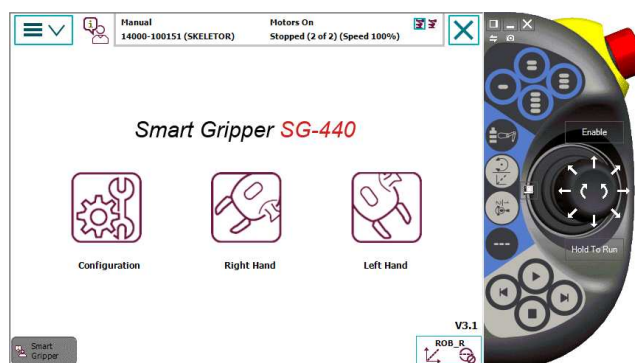
Sintaksa uporabniškega koordinatnega sistema je sledeča:

```
wobj:= [FALSE, TRUE, "", [[0,0,0], [1,0,0,0]],
        [[0,0,0], [1,0,0,0]]];
```

Pri tem prva kombinacija  $[[0,0,0], [1,0,0,0]]$  predstavlja pozicijo in orientacijo novega koordinatnega sistema glede na bazni koordinatni sistem robota, druga kombinacija  $[[0,0,0], [1,0,0,0]]$  pa predstavlja lego objekta, def nirano v novem koordinatnem sistemu wobj.

### 7.3.3 Uporaba prijemala

Robotski roki sta opremljeni z integriranimi robotskimi prijemali. Prijemalo je sestavljeno iz servo motorjev za premikanje prstov, pnevmatike za uporabo vakuumu oziroma izpihavanje ter kamere.



Slika 7.6: Meni za upravljanje z robotskimi prijemali *SmartGripper*

Za prijemala obstaja poseben meni, do katerega se dostopa preko *ABB > Smart Grippers*. V tem meniju sta na voljo posamezni prijemali ter konf guracija. Ob kliku na levo ali desno prijemalo se odpre dodatni meni, kjer lahko nastavite hitrost premikanja prstov ter želeno silo prijemanja. Dodatno lahko ročno premikate prijemala (*Jog+/Jog-*), premikate prijemalo do željene pozicije (*Move to*) oziroma prijemate do nastavljene sile (*Grip to*) ter vklapljate/izklapljate vakuum oziroma zrak. Po vsakem zagonu robota je potrebno izvesti kalibracijo prstov, tako da se jih z *Jog-* stisne ter nato izbere gumb *Calibrate*. Indikator ob gumbu sporoča stanje prijemala (zeleno - prijemalo je kalibrirano, rdeča - prijemalo je potrebno kalibrirati). V program se lahko vključi sledečo kodo, ki v primeru, da prijemala niso kalibrirana, izvede inicializacijo (nastavi maksimalno hitrost na 20 mm/s ter silo prijemanja na 10 N) ter kalibracijo.

```
IF Hand_IsCalibrated() THEN
    !roka je ze kalibrirana
ELSE
    Hand_Initialize \maxSpd:=20, \holdForce:=10,
    \Calibrate;
ENDIF
```

Prijemalo se krmili z ukazi, ki so v programu na voljo preko *Add instruction*, kjer izberete *SmartGripper*. Za premikanje prstov so na voljo sledeči ukazi:

- *Hand\_JogInward* se uporablja za premikanja prijemala navznoter; prijemalo se ne ustavi, dokler ne doseže mehanske omejitve;
- *Hand\_JogOutward* se uporablja za premikanja prijemala navzven; prijemalo se ne ustavi, dokler ne doseže mehanske omejitve;
- *Hand\_MoveTo* premakne prijemalo do nastavljene pozicije;
- *Hand\_GripInward* se uporablja za premikanje prijemala navznoter; nastavlja se lahko dodatne parametre: sila držanja *\holdForce*, željena pozicija *\targetPos*;
- *Hand\_GripOutward* se uporablja za premikanje prijemala navzven; nastavlja se lahko dodatne parametre: sila držanja *\holdForce*, željena pozicija *\targetPos*.

Za manipulacijo pnevmatike se uporabljajo sledeči ukazi:

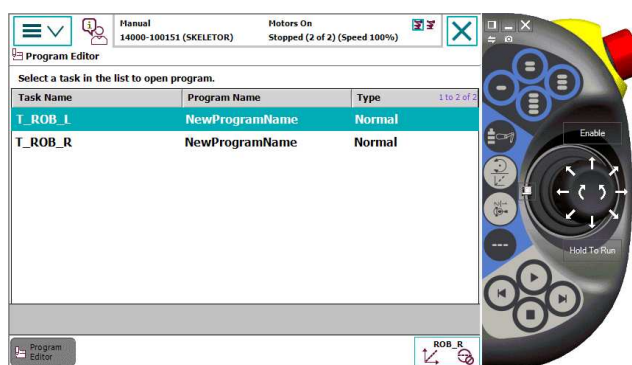
- *Hand\_TurnOnBlow1* vklopi pihanje na kanalu 1;
- *Hand\_TurnOffBlow1* izklopi pihanje na kanalu 1;
- *Hand\_TurnOnVacuum1* vklopi vakuum na kanalu 1;

- Hand\_TurnOffVacuum1 izklopi vakuum na kanalu 1.

Podobni ukazi veljajo tudi za drugi kanal. V programu RobotStudio so ukazi na voljo v zavihku *RAPID > Instruction > SmartGripper*.

### 7.3.4 Program RAPID

Robotski program je sestavljen iz niza ukazov, ki opisujejo kam in na kakšen način naj se robot premakne. Ker je robot IRB 14000 sestavljen iz dveh robotskih rok sta potrebna dva ločena programa. Dostop do programov je preko *ABB > Program Editor*. Tu lahko izberete program za levo (T\_ROB\_L) ali desno roko (T\_ROB\_R).



Slika 7.7: Pregled programov, ki so naloženi za posamezno robotsko roko

Nov program ustvarite z izbiro *File > New Program*, že napisan program pa naložite z *File > Load Program*. Program shranite z *File > Save Program As*. Za učenje programa morate robota najprej postaviti v željeno lego. Nato željeno lego shranite z izbiro *Add Instruction*, v dodatnem meniju pa izberete ukaz, ki def nira način premika robota v to točko. Največkrat uporabljeni ukazi so

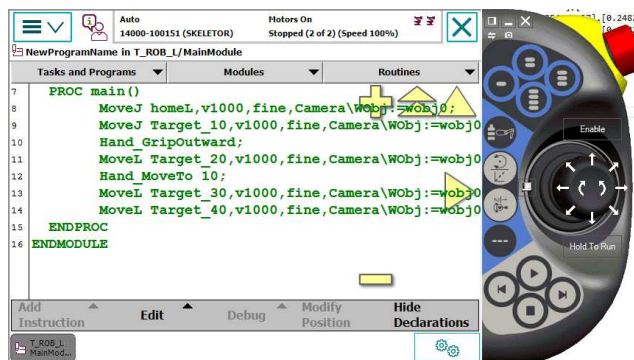
- 1 MoveJ - koordinirano gibanje po sklepkih;
- 2 MoveL - linearno gibanje vrha robota;
- 3 MoveC - krožno gibanje vrha robota.

Sintaksa posameznega ukaza je sledeča:

```
[MotionType] [Name], [Speed], [Zone], [Tool], [WorkObject];
```

Posamezne komponente so

- 1 [MotionType] def nira način premikanja robota;



Slika 7.8: Primer kratkega programa za levo roko

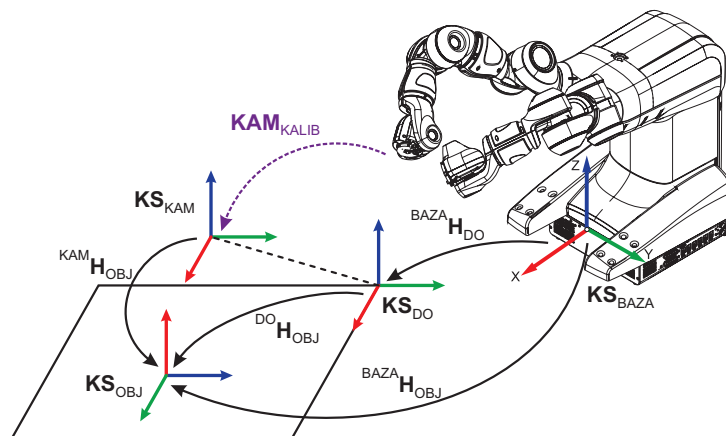
- 2 [Name] je ime točke, ki defnira lego robota; če ime ni defnirano, je nadomeščeno z zvezdico \*;
- 3 [Speed] defnira hitrost robota;
- 4 [Zone] defnira, kako natančno naj robot pride/obide točko; fine določa, da robot pride točno v točko in se tam ustavi;
- 5 [Tool] defnira TCP, katerega naj robot postavi v izbrano točko;
- 6 [WorkObject] defnira koordinatni sistem, glede na katerega je defnirana točka.

## 7.4 Robotski vid

Robot ABB IRB 14000-0.5/0.5 ima v levem prijemalu integrirano industrijsko kamero Cognex In-Sight serije 7000. V industriji se robotski vid uporablja za iskanje in pregled objektov, merjenje objektov ter preverjanje posameznih značilk.

Kamera, nameščena v prijemalu robota, lahko zajema slike s hitrostjo do 102 slike na sekundo z resolucijo  $800 \times 600$  slikovnih pik. Dodatno ima integrirano osvetlitev z LED diodo. Kameri je mogoče nastaviti čas odprtja zaslona, način in interval proženja, intenziteto osvetlitve, ipd. Do nastavitve kamere dostopate preko programskega okolja EasyBuilder, ki je integriran v programski paket RobotStudio.

Predno lahko uporabite kamero, je le-to potrebno kalibrirati s pomočjo različnih kalibracijskih metod. S kalibracijo se določi transformacijo med slikovnimi točkami in milimetri, obenem pa se določi tudi koordinatni sistem kamere, ki je pripet na samo kalibracijsko mrežo. Da se podatke iz kamere lahko uporabi za robota, je potrebno defnirati še uporabniški koordinatni sistem, ki soupada s koordinatnim sistemom kamere. Tako je lega prepoznanega objekta



Slika 7.9. Postavitev koordinatnih sistemov. Koordinatni sistem kamere  $\mathbf{KS}_{KAM}$  in koordinatni sistem delovnega objekta  $\mathbf{KS}_{DO}$  sta soležna, kar nakazuje črtkana črta.

v koordinatnem sistemu kamere enaka legi objekta v uporabniškem koordinatnem sistemu. Relacije med koordinatnimi sistemi so prikazane na sliki 7.9.

**Primer:** Program za prepoznavo objekta s kamero najde objekt na poziciji  ${}^{KAM}\mathbf{KS}_{OBJ} = [0.30.150]$  glede na izhodišče kamere  $\mathbf{KS}_{KAM}$ . Ker je uporabniški sistem  $\mathbf{KS}_{DO}$  poravnani s koordinatnim sistemom kamere  $\mathbf{KS}_{KAM}$  velja, da je objekt na poziciji  ${}^{DO}\mathbf{KS}_{OBJ} = [0.30.150]$  glede na izhodišče delovnega objekta  $\mathbf{KS}_{DO}$ . Ker je delovni objekt definiran glede na bazni koordinatni sistem robota, je lega objekta določena

$${}^{BAZA}\mathbf{H}_{OBJ} = {}^{BAZA}\mathbf{H}_{DO} {}^{DO}\mathbf{H}_{OBJ}.$$

## 7.5 Prepoznavanje in manipulacija objektov

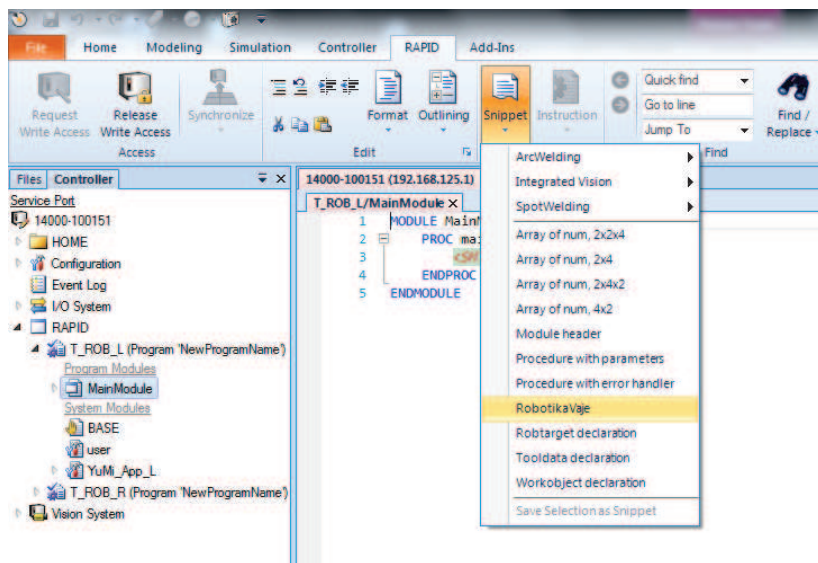
Vaja je sestavljena iz več sklopov:

- priprava programa RAPID;
- kalibracija kamere ter povezava robota in kamere;
- priprava programa robotskega vida za prepoznavo objekta;
- def niranje prijemanja objekta;
- def niranje točk odlaganja objekta;
- nadgradnja programa za prepoznavo in manipulacijo treh objektov.

### 7.5.1 Priprava programa

Pri tej vaji boste uporabljali samo levo robotsko roko, ker je na njej nameščena kamera. Preko hitrega menija nastavite kot aktivno roko samo levo roko, ter za njo ustvarite nov program (*ABB > Program editor > Task and programs > File > New program*). Program shranite v mapo HOME\Programi\ROBOTIKA. Nato se povežete na krmilnik robota s programsko opremo RobotStudio. To storite tako, da poženete program RobotStudio in v zavihku *Controller* kliknete *Add Controller*. Nato izberete še opcijo *Request Write Access*, na ročni učni napravi pa izberete *Grant*. S tem dovolite, da uporabnik RobotStudia pridobi pravice za spreminjanje programa.

Na levi strani v zavihku *Controller > RAPID* odprite drevesno strukturo za levo roko (*T\_ROB\_L*); pri tem se vam odpre seznam uporabljenih modulov. Izberite *MainModule*. Vrstici *PROC Main()* in *ENDPROC* nadomestite s predpripravljenim kodo - t.i. *Snippet*. Do te kode dostopate preko zavihka *RAPID*, kjer izberete *Snippet > ROBVaje*.



Slika 7.10: Drevesna struktura naloženih programov na krmilniku ter Snippet meni

Program boste ustrezno spreminjali med nadaljevanjem vaje. Program boste spreminjali tam, kjer je označeno z značko

```
!##### STUDENT #####
. . .
!#####
```

Program naložite na krmilnik tako, da kliknete na gumb *Apply* v zavihku *RAPID*.

#### Koraki za izvedbo naloge:

- za aktivno roko izberite samo levo roko: *hitri meni > Tasks to Stop and Start*;
- ustvarite nov program za levo roko: *ABB > Program editor > Task and programs > File > New program*;
- povežite RobotStudio in krmilnik robota: *Controller > Add Controller*;
- zahtevajte dovoljenje za pisanje v RobotStudio: *Request Write Access*;
- namesto `PROC Main ... ENDPROC` vstavite predpripravljeno kodo za prijemanje objektov: *RAPID > Snippet > ROBVaje*;
- program naložite na krmilnik *RAPID > Apply*.

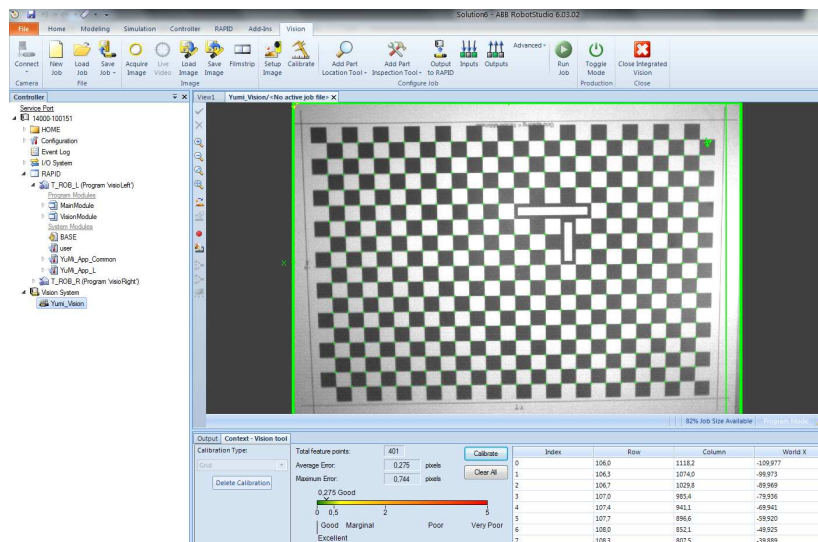
### 7.5.2 Kalibracija kamere

Ko se povežete na robotski krmilnik, se vam pri drevesni strukturi pojavi možnost *Vision System*. Če kliknete na puščico poleg imena, se vam pokaže aktivna kamera - v našem primeru je to kamera z imenom *Yumi\_Vision*. Zavihek *Vision* se vam odpre, če kliknete DMK na *Vision System* in izberete *Integrated Vision*. S kamero se povežete, če kliknete na gumb *Connect*, ki se nahaja na levi strani v zavihku *Vision*. Ko je kamera povezana, lahko naredite posnetek s klikom na *Acquire Image*. Nastavitve kamere vam omogočajo, da nastavite optimalno sliko. Najpomembnejši nastavitvi sta čas odprtja zaslonke (angl. *Exposure*) in intenziteta osvetlitve (angl. *Light Intensity*). Do nastavitve dostopa preko gumba *Setup Image*. Priporočljivo je zmanjšati osvetlitev, ali pa jo kar izklopiti (*Light Control Mode* nastavite na *Disabled*).

Najprej ustvarite nov program za uporabo kamere (angl. *Vision Job*). To storite z *New Job* v zavihku *Vision*. Kliknite *Yes* ter program shranite na samo kamero ter ga poimenujte (klik na *Save Job*). Ta program boste med izvedbo vaje naložili v delovni pomnilnik kamere. V RAPID kodi zamenjajte ime programa `mojrobotskivid.job` z imenom vašega programa.

Pred uporabo je potrebno kamero kalibrirati, kjer se določi pretvorbo med slikovnimi pikami in milimetri ter korekcijo zaradi ukrivljenosti leče oziroma projekcije. Pri tem boste uporabili natisnjeno šahovnico, ki jo postavite na mizo. Levo robotsko roko z uporabo ročne učne naprave postavite v tako lego, da vidite celotno šahovnico. Ko ste zadovoljni s postavitvijo, to shranite v vašem RAPID programu. To storite tako, da na ročni učni enoti izberete vrstico





Slika 7.11: Primer kalibracije kamere

MoveJ položajKamere, v300, fine, GripperL;

ter kliknete *Modify Position*. Ta točka vam predstavlja lego kamere, v kateri boste morali postaviti robota pred uporabo kamere.

Kalibracijo se izbere s klikom na gumb *Calibrate* v RobotStudio. Odpre se vam okno, kjer izberete tip kalibracije. Izberete *Grid* ter preverite, če je razdalja med kockami nastavljena na 10 mm. Nato kliknete *Next*. V naslednjem koraku program za kalibracijo prepozna posamezna polja, ki se vam izpišejo na zaslonu. Če so polja lepo vidna, nadaljujete s kalibracijo. V nasprotnem primeru ustrezno popravite postavite robota/mreže oziroma parametre slike. Kalibracijo poženete s klikom na gumb *Calibrate*. Grafčni prikaz vam pokaže kvaliteto kalibracije. Če je rezultat zadovoljiv, zaključite z izbiro *Finish*. Program shranite z gumbom *Save Job*.

#### Koraki za izvedbo naloge:

- odprite zavihek *Vision: Controller > Integrated Vision*;
- povežite se s kamero: *Controller > Connect*;
- kreirajte nov program robotskega vida: *Vision > New Job*;
- program robotskega vida poimenujte in shranite: *Vision > Save Job*;

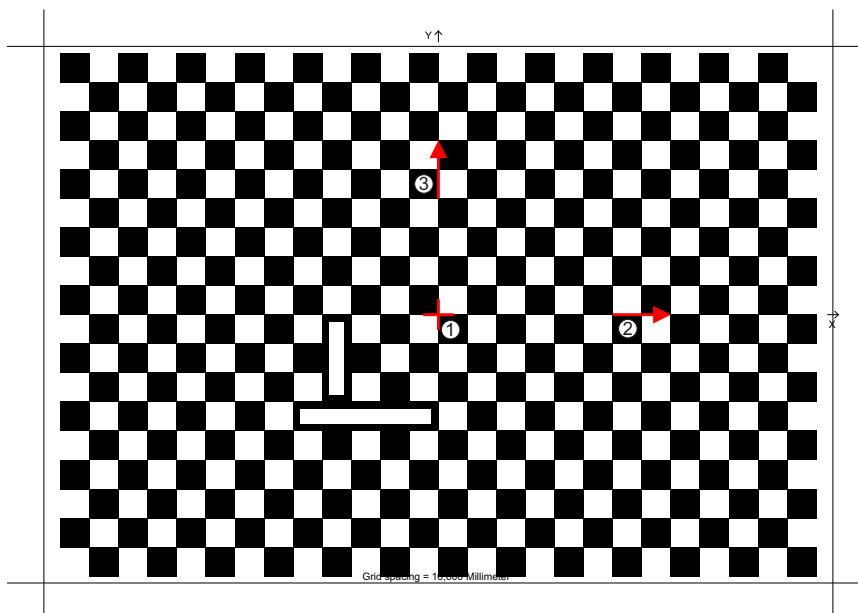
- v predpripravljeni kodi nadomestite `mojrobotskivid.jobz` imenom vašega programa;
- levo robotsko roko postavite tako, da vidite celotno šahovnico; sliko pogledate z *Vision > Acquire Image*;
- shranite točko v robotski program: v programu se postavite na točko položaja `wobjKamere` ter izberete *Modify Position*;
- naredite kalibracijo: *Vision > Calibrate*.

### 7.5.3 Povezava kamere in robota

Po kalibraciji kamere so vse izhodne vrednosti programov za obdelavo slike podane v koordinatnem sistemu kamere. Da lahko robot pravilno interpretira podatke s kamere, ga je potrebno naučiti lego koordinatnega sistema kalibrirane kamere. S tem se def nira skupen koordinatni sistem, ki povezuje kamero in robota.

To storite tako, da def nirate delovni objekt, ki vsebuje informacije o uporabniškem koordinatnem sistemu ter koordinatnem sistemu objekta. S to kalibracijo spremenite samo uporabniški koordinatni sistem, koordinatni sistem objekta pa ostane nespremenjen. Uporabniški koordinatni sistem je določen s postavitvijo kalibracijske mreže, ki mora biti postavljena enako kot pri kalibraciji kamere. V prijemalo leve roke vpnete orodje s špico, ki jo boste uporabili za def niranje novega koordinatnega sistema (*ABB > Smart Grippers > Left Hand > Grip to [0] mm*). Kot orodje leve roke izberete `CalibSpikeL` (*ABB > Jogging > Tool > CalibSpikeL*). Delovni objekt def nirate tako, da preko menija *ABB* izberete *Jogging*, nato pa s klikom na polje poleg *Work object*: odprete seznam uporabniških koordinatnih sistemov. Izbere koordinatni sistem `wobjKamere`. Nato z *Edit > Define* začnete z def nicijo koordinatnega sistema, pri čemer izberete metodo s 3 točkami (*User method: 3 points*). Prva točka je postavljena na presečišču premic, ki opisujeta krajši stranici pravokotnikov na šahovnici, druga točka je nekje na osi v smeri osi x, tretja pa na osi y. Te točke so prikazane na sliki 7.12. Določitev kalibracije zaključite s potrditvijo *OK*.

Po končani kalibraciji še preverite, ali je delovni objekt ustrezno def niran. V hitrem meniju nastavite za delovni objekt `wobjKamere` (druga ikona pod izbranim robotom), izberete vodenje glede na delovni objekt (ikona kocke s koordinatnim sistemom) ter premikavanje po oseh (ikona grafa s črtkano črto). Če sedaj robota premikate s krmilno palico po posameznih oseh, bi se moral vrh robota premikati vzporedno glede na kalibracijsko mrežo.



Slika 7.12: Mreža za kalibracijo kamere z označenim izhodiščem

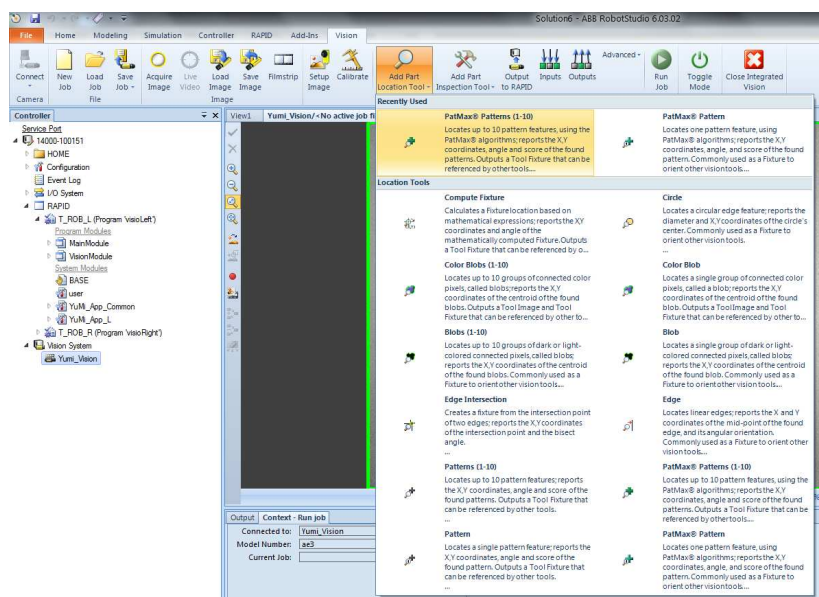
#### Koraki za izvedbo naloge:

- v levo robotsko roko vpnite kalibracijsko konico: *ABB > Smart Grippers > Left Hand > Grip to: [0] mm*
- nastavite orodje CalibSpikeL: *ABB > Jogging > Tool;*
- izberete delovni objekt wobjKamere: *ABB > Jogging > Work object;*
- definirajte točke delovnega objekta: *Edit > Define;* izberete *User method: 3 points;*
- robotsko roko postavite v središče koordinatnega sistema, ter točko User Point X 1 shranite z *Modify Position;* enako naredite še za točki na osi x (User Point X 2) in osi y (User Point Y 1);
- potrdite definicijo novega delovnega objekta z *OK > OK > OK;*
- preiskusite premikanje robota glede na delovni objekt wobjKamere.

## 7.5.4 Definiranje naloge robotskega vida

Ko zaključite s kalibracijo kamere lahko odstranite mrežo, v vidni prostor kamere pa postavite objekt, ki ga želite prepoznati. Robota postavite v položaj za slikanje: *Jogging > Go To...*, izberete položaj kamere ter pritisnete in držite *Go to* dokler se robot ne postavi v pravilno lego. Pri tem pazite, da imate izbran delovni objekt `wobj0`. Zatem preverite kvaliteto slike (*Vision > Acquire Image*); če je potrebno, ustrezno nastavite parametre kamere preko menija *Setup Image*.

Za prepoznavo objektov sta na voljo dva tipa orodij: orodja za določitev lege (*Part Location Tool*) in orodja za preverjanje objektov (*Part Inspection Tool*). Pri tej vaji vas bo zanima lega objektov, zato boste uporabili orodja za določitev lege. Dostopna so preko zavihka *Vision > Add Part Location Tool*. Med bolj pogosto uporabljenimi orodji sta *PatMax* in *Blob* ter njuni različici *PatMax(1-10)* in *Blob(1-10)*, ki omogočata prepoznavo do 10 enakih objektov. Orodje *PatMax* nam omogoča prepoznavo objektov na podlagi geometrijskih značilnk. Orodje *Blob* pa spremlja, kje se nahajajo različne intenzitete barve, ki predstavljajo opazovane objekte.



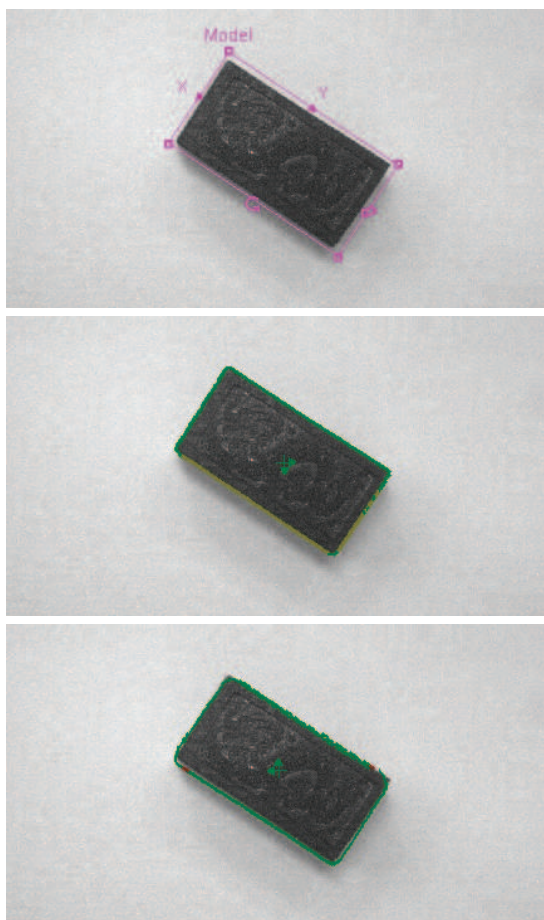
Slika 7.13: Meni z različnimi orodji za prepoznavo objektov

Pri tej nalogi boste uporabili orodje *PatMax*. To orodje se uporablja za določitev pozicije posameznih značilnk objekta. Temelji na vnaprej naučenem modelu objekta. Orodje *PatMax Pattern Tool* se uporablja za določitev enega samega

objekta, medtem ko se *PatMax Patterns (1-10)* uporablja za določitev pozicije do 10 objektov.

Ko izberete orodje *Pat Max Patterns (1-10)*, je potrebno definirati področje modela (ang. Model Region) ter področje, kjer se objekt lahko nahaja (ang. Search Region). Področje modela je lahko pravokotnik, krog, obroč ali poligon. Izbrano področje ustrezno prestavite nad model. Pri tem pazite, da izbrano področje pokriva samo pomembne značilke objekta.

Nato je potrebno definirati področje iskanja. To področje je lahko pravokotnik, krog, obroč ali poligon. Izbira naj pokriva področje, kjer se pričakuje, da bo objekt. Velikost področja vpliva na hitrost delovanja algoritma: večje kot je področje, več časa bo porabil algoritem za analizo. Ko nastavite obe ustrezni področji potrdite izbiro z *OK*.



Slika 7.14. Učenje prepoznavanja novega objekta: definiranje modela (zgoraj); učenje značilke modela (sredina); prepoznavanje objekta (spodaj)

V naslednjem oknu nastavite ime vzorca (*Tool Name*). V tem koraku vam področje modela zamenja z naučenim modelom, obenem pa vam zelen znak koordinatnega sistema kaže središče prepoznanega objekta (pozicijo v x in y smeri ter orientacijo okoli osi z). V zavihku *Context - Vision tool > Settings* nastavite parametre prepoznave objektov; med pomembnejšimi so

- število objektov (*Number to find*) - če se pričakuje več kot en objekt (1-10);
- faktor podobnosti (*Accept Threshold*) - zahtevana podobnost med modelom in objektom (0-100);
- rotacijska toleranca (*Rotation Tolerance*)- določa največji kot, za katerega je lahko zavrten najden objekt ( $\pm 0-180^\circ$ );
- skaliranje (*Scale Tolerance*)- določa dovoljen skalirni faktor med modelom in najdenim objektom (0-50);
- način iskanja (*Find Mode*) - izbira med *PatMax* in *PatQuick*; slednji je hitrejši, a manj natančen;
- horizontalni odmik (*Horizontal Offset*) - določa odmik od centra najdenega objekta v horizontalni smeri (v slikovnih točkah);
- vertikalni odmik (*Vertical Offset*) - določa odmik od centra najdenega objekta v vertikalni smeri (v slikovnih točkah).

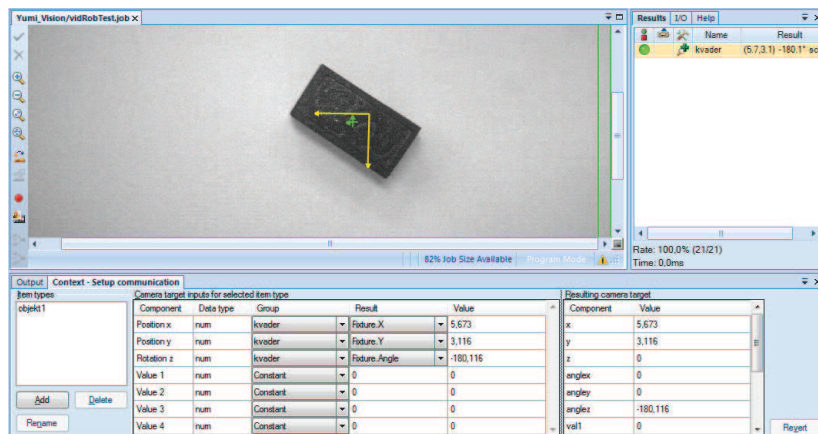
Sedaj lahko testirate delovanje algoritma pri različnih postavitvah predmeta. Če algoritem ne deluje ustrezno, ga lahko popravite s pomočjo menija v spodnjem desnem kotu. Ta meni vam omogoča tudi bolj detaljno definicijo modela (dodajanje/odstranjevanje področja, spreminjanje modela, spreminjanje področja iskanja ...). Ko ste zadovoljni z delovanjem algoritma za prepoznavo shranite trenutni program (*Vision > Save Job*).

V nadaljevanju je potrebno nastaviti povezave med orodjem za prepoznavo objektov ter robotom (RAPID programsko kodo). To se nastavi z izbiro *Vision > Output to RAPID*, ki odpre tabelo z nastavitvami (glej sliko 7.15). Le-ta povezuje izhod iz orodja za prepoznavo z RAPID spremenljivko tipa *cameratarget*. Najprej ustrezno smiselno poimenujete objekt (*Item types > Add > Rename*). V stolpcu *Group* izberete ime orodja za prepoznavo, ki ste ga nastavili. V stolpcu *Results* definirate, katere podatke želite posredovati v RAPID kodo (pozicijo v x in y smeri, ter rotacijo okoli z osi). Na koncu shranite trenutni program.

Ukaz v robotskem programu

```
CamGetResult mycamera, mycameratarget;
```

izvede zapis vrednosti, ki jih vrne program za prepoznavo nastavljen v *Output to RAPID*, v strukturo *mycameratarget*.



Slika 7.15. Primer tabele, ki povezuje podatke iz kamere s tarčno spremenljivko *cameratarget*

### Koraki za izvedbo naloge:

- robota postavite v položaj za slikanje (*ABB > Jogging > Go To...*); pri tem imejte izbran delovni objekt *wobj0*;
- v vidno polje kamere postavite en objekt;
- zajemite sliko: *Vision > Acquire Image*;
- nastavite ustrezne parametre slike (*Vision > Setup Image*):
  - *Exposure*,
  - *Light Intensity*;
- izberite orodje za prepoznavo objektov: *Vision > Add Part Location Tool > Pat Max Patterns (1-10)*;
- nastavite področje modela ter področje iskanja;
- poimenujte orodje: *Tool Name*;
- nastavite parametre orodja (*Context - Vision tool > Settings*):
  - *Number to find*: 3,
  - *Rotation Tolerance*: 90,
  - *Scale Tolerance*: 5;
- testirajte delovanje orodja pri različnih postavitvah objekta;

- nastavite povezave med orodjem za prepoznavo ter RAPID programom:  
*Vision > Output to RAPID;*
- shranite program: *Vision > Save Job.*

### 7.5.5 Prijemanje in manipulacija objekta

Pred prijemanjem morate koordinatnemu sistemu vašega delovnega objekta, ki je definiran v mywobj sistemu (.oframe - *object frame*), prirediti vrednost (lego objekta), ki jo določite pri prepoznavi objekta s kamero (.cframe - *current frame*). To stori koda

```
wobjKamere.oframe := mycameratarget.cframe;
```

Orodje, ki ga boste uporabili za pobiranje objektov, bo kar servo prijemalo, ki je nameščeno na levi robotski roki. Vrh prijemala je definiran s TCP kalibracijo z imenom GripperL, kot delovni objekt pa mora biti izbran wobjKamere. Točka prijemanja je podana z ukazom

```
MoveL prijemObjekta, v200, fine, GripperL
  \Wobj:=wobjKamere;
```

To točko je potrebno ustrezno spremeniti, da bo predstavljala lego robotskega prijemala v trenutku prijemanja. Z ročno učno napravo prezamete pravice pisanja programskemu paketu RobotStudio (izberete *Revoke*), odprete program za levo roko, postavite programski kazalec na začetek (*Debug > PP to main*) ter poženite program (f zična tipka *Play*). Program se bo izvedel do vrstice, kjer je Break. Do takrat se bo izvedla inizializacija kamere, robot se bo postavil v položaj za slikanje, kamera bo zajela sliko, program bo prepoznal objekt, njegovo pozicijo vnil v strukturi mycameratarget, lega objekta pa se bo zapisala v delovni objekt wobjKamere. Na ročni učni enoti se bo izpisala lokacija, kje se nahaja trenutno prepoznani objekt. Preverite, če je podana lega smiselna: objekt mora biti za prepoznano število mm oddaljen od izhodišča uporabniškega koordinatnega sistema wobjKamere.

Sedaj robota ročno vodite do objekta. Pri tem pazite, da objekta ne premaknete, robota pa postavite v najbolj optimalno lego za prijemanje. Ko je robot postavljen, popravite pozicijo točke prijemObjekta z ukazom *Modify Position*. Pri tem mora biti kot aktivni uporabniški koordinatni sistem izbran koordinatni sistem wobjKamere (*ABB > Jogging > Work Object*).

V nadaljevanju programa dodajte še ukaz za zapiranje prijemala. S kurzorjem se postavite na ustrezno mesto v programu, ukaz pa vstavite z izbiro *Add Instruction > SmartGripper*. Ukaz za zapiranje je Hand\_GripInward za odpiranje pa Hand\_GripOutward.



Nato s programom RobotStudio ponovno prevzamete pravice pisanja ter zakomentirajte vrstico, ki vsebuje `Break`. To storite tako, da pred `Break` postavite klicaj (!): `!Break;`. Program naložite na krmilnik z **RAPID > Apply**.

Sedaj lahko program testirate. Prevezemite pravice pisanja z ročno učno napravo (**Revoke**), postavite programski kazalec na začetek (**Debug > PP to main**) ter poženite program. Če je vse narejeno pravilno, se bo robot postavil na prepoznani objekt ter ga prijel.

Nato z uporabo ročnega vodenja robota def nirajte še ostale potrebne točke, da robot odloži prijet objekt v ustrezno škatlo. Te točke ne smejo biti def nirane v koordinatnem sistemu kamere/objekta, saj se ta spreminja glede na lego objekta. Pri tem ne pozabite uporabiti ukaz za odpiranje prijemala.

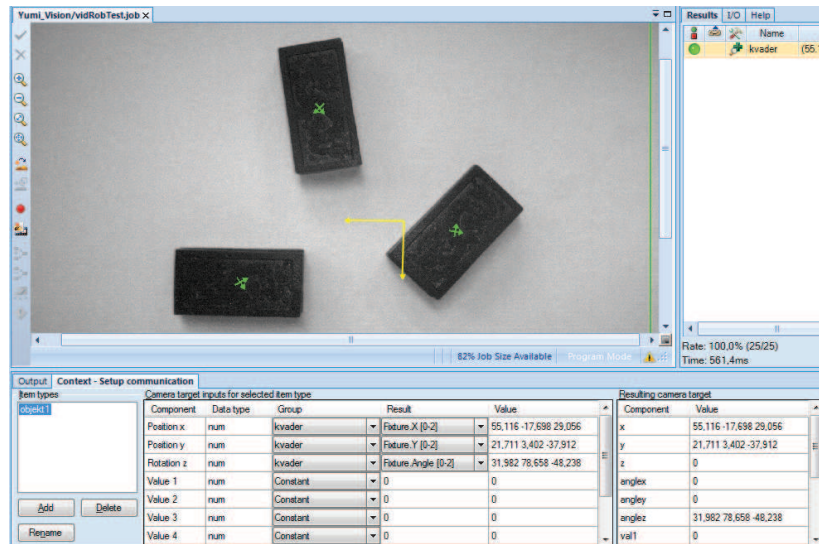
#### Koraki za izvedbo naloge:

- poženite program: **Debug > PP to main**, f zična tipka **Play**;
- premaknite robota do objekta v položaj za prijemanje; kot delovni objekt nastavite `wobjKamere`;
- s kurzorjem se postavite na vrstico `MoveL prijemObjekta ...` ter izberite **Modify Position**;
- dodajte ukaz za zapiranje prijemala: `Hand_GripInward`;
- zakomentirajte vrstico, ki vsebuje `Break`;
- ponovno poženite program;
- dodajte točke/ukaze za odlaganje objekta;
- v vidno polje kamere postavite en objekt ter poženite program.

### 7.5.6 Prepoznavna in manipulacija več objektov

Ko uspešno prepoznate, poberete in odložite en objekt, nadgradite vaš program za prepoznavo in manipulacijo več istih objektov. V nastavitvah orodja za prepoznavo *Pat Max Patterns (1-10)* ste že nastavili največje število objektov, ki jih pričakujete, na 3.. Če program prepozna več istih objektov, se lege objektov shranijo v vektor. Ukaz `CamGetResult` potem kliče en element naenkrat.

Da poberete vse elemente, je potrebno v program vključiti ustrezno zanko. Pri tem si lahko pomagata z ukazom `CamNumberOfResults`, ki vrne število prepoznanih objektov, ki so še na razpolago. Primer testne kode je podan v nadaljevanju.



Slika 7.16. Prepoznavna več objektov: na vsak prepozna objekt se pripne koordinatni sistem, izhod v RAPID program pa so posamezni elementi vektorjev.

```

VAR bool continueloop:=TRUE;
...
PROC main()
...
[del programa, ki inicializira kamero]
...
WHILE continueloop DO
    CamGetResult Yumi_Vision, mycameratarget;
    wobjKamere.oframe := mycameratarget.cframe;
    ...
    [del programa, ki pobere/odnese/odloži objekt]
    ...
    IF CamNumberOfResults(Yumi_Vision)<1 THEN
        continueloop:=FALSE;
    ENDIF
ENDWHILE
ENDPROC

```

**Koraki za izvedbo naloge:**

- v vidno polje kamere postavite tri enake objekte;
- preverite, če program za prepoznavo prepozna tri objekte;
- v robotski program dodajte WHILE zanko za pobiranje vseh objektov;
- naložite program na krmilnik *RAPID > Apply*;
- poženite program.

### 7.5.7 Zagon v avtomatskem načinu

Ko preverite vaš program, da deluje v ročnem načinu (robot prepozna, pobere in odloži tri objekte), lahko program požene v avtomatskem načinu. V tem načinu se bo robot premikal s hitrostmi, kot ste jih definirali v programu. Najprej odprete produkcijsko okno: *ABB > Production Window*. Nato v hitrem meniju v *Operator Panel* izberete avtomatski način izvajanja (*Auto*). V meniju *Task to Stop and Start* izberete samo levo roko - T\_ROB\_L. Nato v produkcijskem oknu kliknete *PP to Main* ter s tipko *Play* zaženete program.