Laborator 4

Transformările Base și Tool

As. ing. Alexandru Dumitrache
As. ing. Raluca Tudorie
www.scr.cimr.pub.ro

Transformarea Base

În laboratoarele anterioare, toate locațiile învățate au fost exprimate în sistemul de coordonate World al robotului. Această abordare s-a dovedit dificilă atunci când axele de lucru (ale paletei) nu au fost perfect aliniate cu axele World.

O soluție elegantă este definirea unui sistem de coordonate local, specific task-ului efectuat (de exemplu, pe paletă), și învățarea locațiilor de lucru în acest sistem.

Exemplu

Fie punctul `p` exprimat în sistemul de coordonate local `bs` (Fig. 4.1). Punctul `p` are coordonatele omogene $[dx\ dy\ 0\ 1]^T$:

```
SET p = TRANS(dx, dy, 0)
```

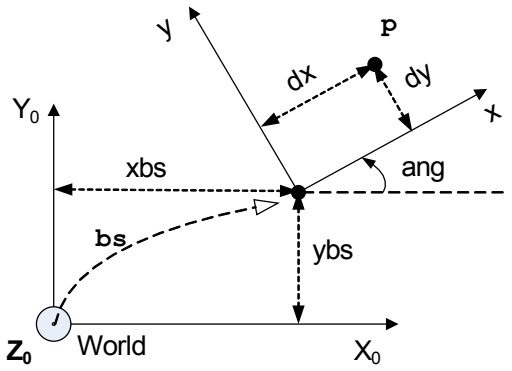

Figura 4.1: *Un punct învățat într-un sistem de coordonate local*

Raportat la sistemul World $X_0Y_0Z_0$, sistemul `bs` are originea în punctul `(xbs,ybs,0)` și este rotit în jurul axei Z cu unghiul `ang`:

```
SET bs = TRANS(xbs, ybs, 0):RZ(ang)
```

Dacă dorim să exprimăm punctul `p` în sistemul de referință World, îl vom înmulți la stânga cu matricea omogenă a sistemului local `bs`. În V^+ vom folosi operatorul de compunere a transformărilor:

```
SET p.world = bs:p
```

Punctul `p.world` rezultat va avea coordonatele:

- `x = xbs + dx * COS(ang) - dy * SIN(ang)`
- `y = ybs + dx * SIN(ang) + dy * COS(ang)`
- `z = 0`

Un sistem de coordonate poate fi definit cu funcția `FRAME` (Fig. 4.2):

```
SET bs = FRAME(a, b, c, d)
```

unde a, b, c și d sunt patru locații învățate de utilizator (de exemplu, folosind comanda monitor `HERE`). În calculul sistemului de coordonate se consideră doar componenta de translație (X, Y, Z), orientarea acestora fiind ignorată.

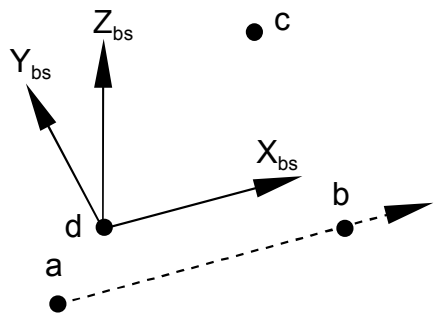

Figura 4.2: *Definirea unui sistem de coordonate folosind* ***FRAME***

Sistemul de coordonate definit cu `FRAME` este calculat astfel:

- Originea sistemului `bs` este în punctul `d`;
- Vectorul $a \rightarrow b$ determină axa X;
- Punctele a, b și c determină planul XY;
- Direcția axei Y este aleasă în planul (a, b, c), perpendiculară pe axa X, în sensul indicat de vectorul $a \rightarrow c$;
- Axa Z este $cross(X, Y)$[1], astfel încât sistemul rezultat respectă regula mâinii drepte.

Pseudocod în notație Matlab/Octave pentru funcția `FRAME`:

```
function f = FRAME(a, b, c, d)
   % a, b, c, d: vectori coloană 3x1 [x;y;z]

   x = b - a;
   y_tmp = c - a;
   z = cross(x, y_tmp);
   y = cross(z, x);

   x = x / norm(x);
   y = y / norm(y);
   z = z / norm(z);

   % Matricea de rotație este [x y z];
   % Se convertește la yaw/pitch/roll:
   [yaw, pitch, roll] = rotation_matrix_to_euler_angles([x y z])

   f = TRANS(d(1), d(2), d(3), yaw, pitch, roll)
end
```

[1]Produs vectorial

Fiind definit sistemul de coordonate local, în variabila de tip transformare `bs`, învățarea unui punct `loc` în acest sistem se face prin:

```
HERE bs:loc
```

sau, echivalent:

```
SET loc = INVERSE(bs):HERE
```

Pentru a deplasa robotul în punctul astfel învățat:

```
APPRO bs:loc, 100
MOVES bs:loc
```

Observație: Există și instrucțiunea `BASE`, însă este limitată la translații pe X, Y, Z și rotație în jurul lui Z.

Exercițiu

Fie un punct `loc.world` învățat în sistemul de coordonate *World* al robotului. După definirea sistemului de coordonate local `bs` se dorește trecerea punctului în noul sistem de coordonate. Cum se poate determina noul punct (denumit `loc.bs`) fără a fi necesară reînvățarea acestuia cu `HERE` ?

Paletizare pe plan înclinat

Transformarea Base permite lucrul pe plan înclinat. Vom modifica problema de paletizare din laboratorul 3 pentru a exemplifica acest lucru.

Planul înclinat este unic definit prin 3 puncte. Se va determina un sistem de coordonate cu axele paralele cu x_p, y_p și z_p (Fig. 4.3), dar cu originea diferită.

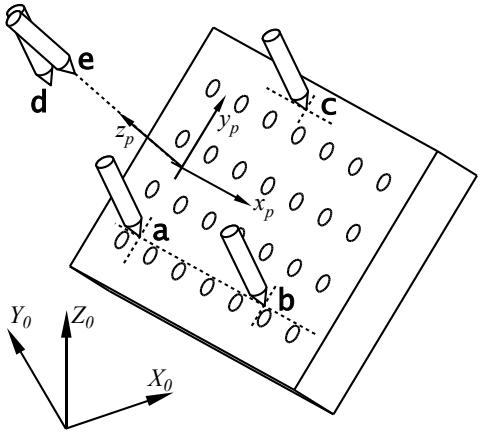

Figura 4.3: *Punctele învățate pentru paletizarea pe plan înclinat*

Procedura de învățare:

- Se fixează pointer-ul conic în gripper-ul robotului articulat vertical;
- Se învață punctele `a`, `b` și `c` pe paletă, conform Fig. 4.3. Vârful pointerului va atinge paleta, orientarea acestuia va fi aproximativ perpendiculară pe plan, dar aceeași pentru toate punctele, iar punctele `a` și `b` vor defini axa x_p a paletei;
- Se învață punctul `d` la o distanță suficient de mare de paletă (100 mm). Acesta va defini originea sistemului de coordonate `bs`, paralel cu paleta;
- Se definește sistemul de coordonate `bs` prin comenzile:

  ```
  .do set bs = frame(a, b, c, d)
  ```

- Se verifică direcția axei Z a sistemului `bs`:

  ```
  .here bs:test
  ```

  Dacă unghiul *pitch* al punctului `test` este apropiat de 180 °, iar componentele de translație sunt egale cu 0, este în regulă. Dacă unghiul *pitch* este apropiat de 0, înseamnă că axa Z_{bs} "intră" în paletă. Deoarece dorim ca Z_{bs} să "iasă" din paletă, vom inversa punctele `a` și `b`:

  ```
  ; doar dacă pitch-ul în pct. "test" este apropiat de 0:
  .do set bs = frame(b, a, c, d)
  ```

- Se calculează punctul `e`, în care tool-ul robotului devine perpendicular pe planul învățat, și în plus are aceeași origine cu `d`:

```
.do set e = bs:ry(180)
```

- Verificare: punctele `d` și `e` ar trebui să aibă poziția identică, iar unghiul *pitch* să fie ușor diferit.

```
.listl d, e
```

- Dacă punctul `e` este calculat corect, robotul se reorientează cu gripper-ul perpendicular pe planul înclinat:

```
.speed 10
.do moves e
```

- În continuare, robotul va fi controlat pe modul *Tool* doar prin translație și rotație în jurul lui Z_{tool}. Translația pe X_{tool} și Y_{tool} va deplasa TCP-ul (Tool Center Point) paralel cu planul învățat. Se îndepărtează pointer-ul conic din gripper, se prinde piesa de lucru și se învață prima poziție de pe paletă:

```
.here bs:pal
```

Programul de paletizare este 90% identic cu `palet.ideal` din laboratorul 3. Se rulează numai pe roboții articulați vertical (Viper).

Se cunosc:

- `#safe` - locație în afara spațiului de lucru;
- `st` - locație la baza stivei verticale;
- `bs` - sistemul de coordoate local al paletei;
- `pal` - poziția primei piese pe paletă, în sistemul `bs`.
- `dx, dy, dz, n = nl * nn * nc` - din laboratorul 3.

Calculul poziției în care va fi așezată o piesă se va face cu:

```
SET place = bs:SHIFT(pal BY i*dx, j*dy, k*dz)
```

Observație: Pentru a se evita coliziunea cu paleta înclinată, mișcarea de la `pick` la `place` și înapoi se face printr-un punct intermediar `#aux`.

```
.PROGRAM pal.inc()
    ; Laboratorul 4 - Paletizare pe plan inclinat

    GLOBAL #safe, st, bs, pal ; locatii robot:
    AUTO dx,dy,dz,nl,nc,nn
    AUTO i,j,k,p,r,nr.piese
    AUTO pick, place

    nl = 3
    nc = 2
    nn = 2
    nr.piese = nl*nc*nn

    dx = 1.25 * 25.4 * 2
    dy = dx * 1.5
    dz = 4.3

    SPEED 100 ALWAYS
    OPEN
    MOVE #safe
    BREAK

    LEFTY
    ABOVE
    NOFLIP

    FOR p = 1 TO nr.piese     ; p = indicele piesei curente
        r = nr.piese - p + 1  ; r = nr. pieselor din stiva

        i = (p-1) MOD nl
        j = INT((p-1)/nl) MOD nc
        k = INT((p-1)/(nl*nc))
        TYPE i, ", ", j, ", ", k

        SET pick = SHIFT(st BY 0,0,(r-1)*dz)
        SET place = bs:SHIFT(pal BY i*dx, j*dy, k*dz)
        CALL pick.place(pick, place)
    END

    MOVE #safe        ; intoarcere in #safe
.END
```

```
.PROGRAM pick.place(pick, place)
    ; Laboratorul 4 - Subrutina pick.place

    AUTO z.pick, z.place
    z.pick = 100
    z.place = 100

    PARAMETER HAND.TIME = 0.2

    OPEN
    MOVE #aux
    BREAK

    APPRO pick, z.pick
    BREAK
    SPEED 50
    MOVES pick
    CLOSEI
    SPEED 30
    DEPARTS z.pick
    BREAK

    MOVE #aux
    BREAK

    APPRO place, z.place
    BREAK
    SPEED 20
    APPROS place, 0.5  ; marginea de siguranta
    OPENI
    SPEED 50
    DEPARTS z.place
    BREAK
.END
```

Transformarea Tool

Până acum, operațiile de mișcare efectuate cu robotul au fost definite prin puncte învățate și eventual alterate printr-o translație (așa-numitele shiftări). În aceste situații nu a fost necesară cunoașterea exactă a poziției punctului condus de către robot.

Ce se întâmplă dacă dorim să executăm, de exemplu, o mișcare de înșurubare? Trebuie să ne asigurăm că axa de rotație coincide cu axa șurubului. Aceasta înseamnă că în urma rotației (în jurul axei Z_T), șurubul (axul, piulița, piesa cilindrică, etc) nu are voie să se deplaseze în lateral. Dacă gripper-ul nu este perfect simetric, nu merge.

Dar dacă avem atașat pe robot un instrument pentru etanșarea unor piese cu silicon, de exemplu? Sau orice altă unealtă care are un vârf ascuțit (electrod de sudură) care trebuie să își schimbe orientarea în timpul operației? Cunoaștem traiectoria pe care trebuie s-o urmeze vârful uneltei (o secvență de coordonate $X/Y/Z/Yaw/Pitch/Roll$, extrasă dintr-un model CAD al piesei de lucru). Problema devine dificilă în special din cauza necesității de a modifica orientarea sculei.

Dacă punctul condus ar fi exact în vârful uneltei, problema ar fi rezolvată.

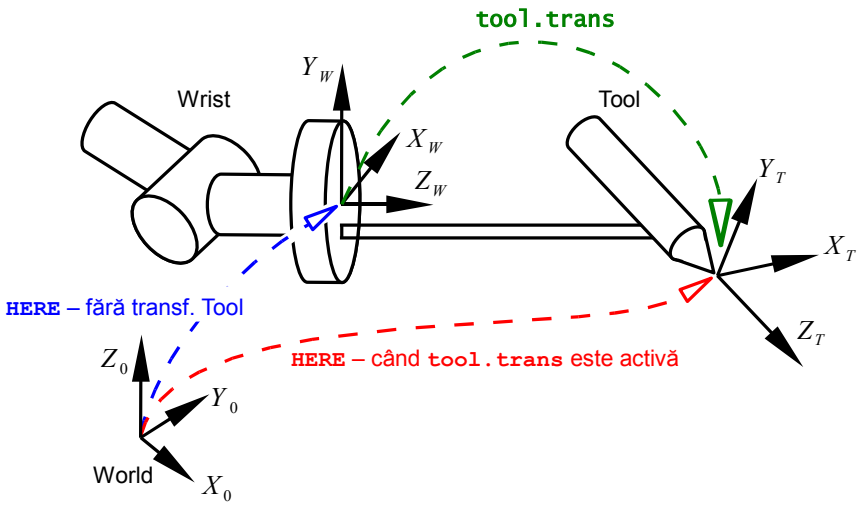

Figura 4.4: *Transformarea Tool*

Aici ne ajută transformarea Tool. Rolul ei este de a muta sistemul de coordonate asociat uneltei (Tool) din încheietura robotului (poziția implicită) într-o locație convenabilă (setată de utilizator) - Fig. 4.4. Se poate modifica atât poziția, cât și orientarea sistemului Tool.

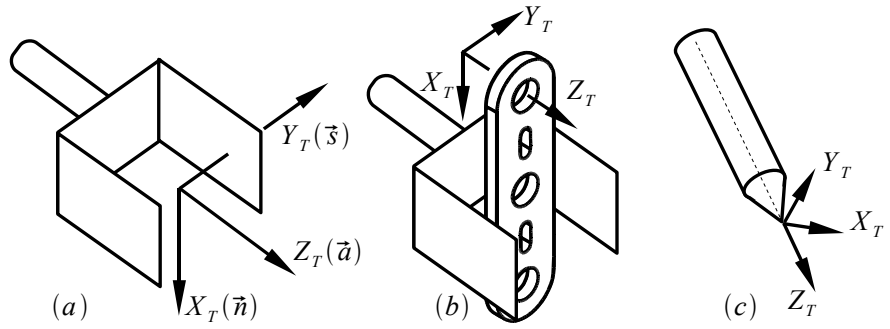

Figura 4.5: *Exemple de alegere a sistemului de coordonate Tool: (a) - Gripper cu 2 degete - $(\vec{n}, \vec{s}, \vec{a})$; (b) Piesă cu o gaură circulară; (c) Instrument cu vârf ascuțit (pointer conic)*

Ce înseamnă locație convenabilă? Pentru problema de înșurubare, sistemul de coordonate Tool va fi ales astfel încât axa Z_T să coincidă cu axa șurubului din gripper-ul robotului (piuliței, piesei cilindrice etc). Pentru problema utilizării unei unelte cu vârf ascuțit, originea sistemului de coordonate va fi în vârful uneltei, iar axa Z_T va indica direcția acesteia. Prin convenție, prin deplasarea uneltei în sensul pozitiv al axei Z_T, aceasta se va apropia de piesă.

Pentru gripper-ul cu două degete se recomandă folosirea sistemului de coordonate $\vec{n}, \vec{s}, \vec{a}$ (normal, slide, approach, Fig. 4.5 a).

Sistemul de coordonate din Fig. 4.5 (b) poate fi folosit pentru a efectua o rotație în jurul unei găuri oarecare de pe piesă. Acest lucru ne permite, de exemplu, montarea piesei într-un ax printr-o mișcare de înșurubare.

Ce se întâmplă dacă un robot dispune de un mecanism automat de schimbare a sculelor? Aceste scule au lungimi diferite. Este posibil ca și orientarea acestora să difere. Se poate învăța câte un punct pentru fiecare sculă, sau se poate învăța un singur punct, folosind una din scule. A doua variantă presupune cunoașterea unei transformări Tool pentru fiecare sculă în parte.

La pornirea robotului, transformarea Tool este `NULL`. Adică I_4 pentru cei cu înclinații spre algebra liniară, sau `eye(4)` pentru cei care preferă notația Matlab/Octave.

Acest lucru înseamnă că sistemul de referință asociat sculei coincide cu S.C. al ultimei articulații din lanțul cinematic. Pentru roboții din laborator, originea S.C. Tool este în centrul flanșei pentru montarea gripper-ului.

Exercițiu

Sistemul de coordonate Tool poate fi studiat cu ajutorul unui robot cu 6 grade de libertate. Folosind MCP-ul, comutați în modul Jog - Tool și efectuați mișcările de translație pe X, Y și Z. Rețineți direcția axelor.

Efectuați mișcările de rotație RX, RY și RZ și apoi efectuați din nou translație pe cele 3 axe. Ce observați?

Puteți deplasa robotul folosind doar $X+$ și $RZ+$, astfel încât gripper-ul acestuia să parcurgă (aproximativ) o traiectorie circulară?

Setarea transformării Tool în V^+

Pentru a seta transformarea Tool se folosește:

```
TOOL new.tool
```

Pentru a reseta transformarea Tool la valoarea implicită:

```
TOOL NULL
```

`TOOL` poate fi instrucțiune program, comandă monitor, și poate fi apelată și ca funcție:

```
.LISTL TOOL
.TOOL TOOL:other.tool
```

Atenție! Un punct învățat cu o transformare Tool este valabil doar cu acea transformare activă. Nerespectarea acestei reguli duce de obicei la coliziuni.

Următoarele două exemple sunt echivalente:

```
.TOOL tool.trans                .TOOL NULL
.MOVE loc              și       .MOVE loc:INVERSE(tool.trans)
.TOOL NULL
```

La schimbarea transformării Tool, robotul execută automat `BREAK`.

Calculul transformării Tool

Metoda generală

Avem nevoie de două puncte robot. Primul este `ref.loc`, cu semnificația următoare: când robotul se află într-o pozitie de referință (cunoscută de noi), dorim ca sistemul de coordonate Tool să fie identic cu `ref.loc`. Cu alte cuvinte, comanda `HERE` va trebui să indice `ref.loc`.

Poziția `ref.loc` se poate învăța cu ajutorul unei unelte de referință, pentru care se cunoaște transformarea Tool `old.tool`, sau se poate calcula.

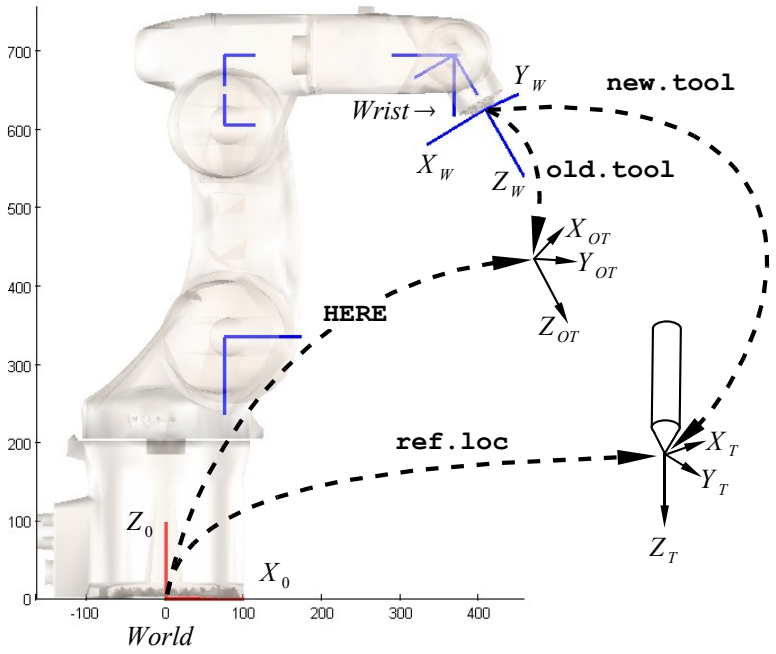

Figura 4.6: *Calculul transformării Tool*

Metoda de calcul a transformării Tool este:

- Cunoaștem transformarea Tool pentru o unealtă de referință, `old.tool`.
- Se determină poziția `ref.loc` (prin învățare cu ajutorul sculei de referință, sau prin calcul).
- Se poziționează robotul cu vârful sculei noi (cea pentru care dorim să învățăm transformarea Tool), în poziția `ref.loc`. Poziția indicată de `HERE` va fi diferită de `ref.loc` (Fig. 4.6).

- Se determină noua transformare Tool cu comanda:

```
.DO SET new.tool = old.tool:INVERSE(HERE):ref.loc
```

- Verificare:

```
.TOOL new.tool
.HERE new.loc
```

  Punctul `new.loc` trebuie să fie identic cu `ref.loc`.

Dacă avem o unealtă de referință pentru care cunoaștem transformarea Tool `old.tool`, putem învăța punctul `ref.loc` cu comenzile:

```
.TOOL old.tool
; se deplasează robotul cu vârful sculei de referință în ref.loc
.HERE ref.loc
```

Inițial, transformarea Tool este `NULL`, iar sistemul Tool este identic cu sistemul de coordonate al articulației terminale (Wrist).

Dacă nu avem nici o unealtă de referință, a cărei transformare Tool să fie cunoscută, o posibilitate pentru învățarea punctului `ref.loc` ar fi să se demonteze gripper-ul, pentru a putea duce punctul din centrul flanșei în dreptul unui reper fixat.

Preferăm să nu demontăm gripper-ul; în acest caz vom seta `old.tool = NULL` și vom determina punctul `ref.loc` prin alte procedee.

În continuare vom particulariza metoda pentru două cazuri relativ ușoare, pe roboții SCARA. Vom alinia piesa de tip I din gripper cu axa X_{world} a robotului, apoi vom muta punctul condus exact în centrul piesei, atunci când punctul de prindere este excentric, sau gripper-ul este asimetric.

Metodele particulare funcționează și la Viper atâta timp cât pitch-ul este exact $180\,^\circ$ (vezi instrucțiunea `ALIGN`).

Ajustarea orientării

Această metodă permite reorientarea sistemului de coordonate Tool, și poate fi utilă la determinarea vectorilor $\vec{n}, \vec{s}, \vec{a}$ pentru gripper, ca în Fig. 4.5 (a).

Se orientează gripper-ul paralel cu axele World, adică Y_{tool} dorit va fi paralel cu Y_{world} și în același sens, iar X_{tool} și Z_{tool} paraleli cu X_{world} și respectiv Y_{world}, dar având sensuri opuse (pentru că unghiul $pitch = 180\,^\circ$).

Pentru robotul Viper, se poziționează gripper-ul vertical pe suprafața de lucru ($pitch = 180\,^\circ$):

```
.DO ALIGN    ; poziționare la pitch=180, doar pentru Viper
```

Alinierea cu axele World poate fi făcută din ochi (aproximativ), sau poate fi făcută cu precizie folosind o piesă de tip I. Se alege un ax vertical fixat, și se învață 2 puncte, `a` și `b`, cu cele 2 găuri extreme în ax. Punctele `a` și `b` vor avea aceeași orientare:

```
.HERE a
; deplasare robot folosind MCP, doar pe translație
.HERE b
```

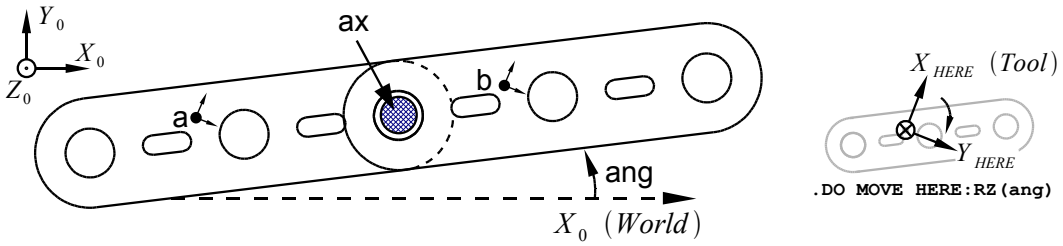

Figura 4.7: *Alinierea unei piese de tip I cu axa X_0 (World)*

Unghiul cu care piesa I a fost rotită față de axa X_{world}, în jurul axei Z_{world}, se calculează cu `ATAN2` (Fig. 4.7):

```
.DO ang = ATAN2(DY(b) - DY(a), DX(b) - DX(a))
```

Poziția piesei se va corecta prin:

```
.DO DEPARTS 50
.DO MOVE HERE:RZ(ang)
```

Observație: Unghiul de corecție `ang` a fost calculat față de Z_{world}, dar rotația a fost făcută în jurul lui Z_{tool}. Deoarece s-a presupus că unghiul *pitch* este egal cu 180° (gripper-ul orientat vertical în jos), cele 2 axe Z sunt paralele, dar au sensuri diferite. Acesta este motivul pentru care s-a folosit `RZ(ang)` și nu `RZ(-ang)` (vezi Fig. 4.7, în dreapta).

După ce piesa este aliniată cu axele World, se execută pașii următori:

- Se memorează o locație auxiliară `loc`:

  ```
  .DO SET old.tool = TOOL
  .HERE loc
  ```

- Se calculează `ref.loc` ca având originea identică cu a punctului `loc`, dar orientarea sa va fi `(0, 180, 0)`:

  ```
  .DO SET ref.loc = TRANS(DX(loc),DY(loc),DZ(loc), 0, 180, 0)
  ```

- Se calculează transformarea Tool pentru ajustarea orientării:

  ```
  .DO SET new.tool = old.tool:INVERSE(loc):ref.loc
  ```

- Verificare: `new.tool` trebuie să difere, față de `new.tool`, doar pe RZ:

  ```
  ; se compară old.tool și new.tool:
  .LISTL new.tool, old.tool

  ; sau se listează "diferența" între new.tool și old.tool
  .LISTL INVERSE(old.tool):new.tool
  ```

  Expresia `INVERSE(old.tool):new.tool` trebuie să aibă o singură componentă nenulă, pe RZ.

- Se aplică transformarea `new.tool`:

  ```
  .TOOL new.tool
  ```

Această metodă modifică orientarea axelor sistemului Tool, însă originea acestuia rămâne neschimbată.

Transformarea Tool pentru gripper excentric

Această metodă se poate folosi în cazul în care se dorește efectuarea unei operații de înșurubare, sau rotația în jurul unui punct poziționat excentric față de axa Z a flanșei. Este utilă de asemenea atunci când gripper-ul nu este centrat. Prin această metodă se poate învăța S.C. din Fig. 4.5 (b).

Metoda constă în învățarea a două locații, astfel încât orientarea acestora să difere cu 180 ° (Fig. 4.8). Poziția carteziană $World$ a punctului de rotație va fi aceeași.

Pașii necesari:

- Pentru robotul Viper, efectorul terminal se poziționează perfect vertical pe planul de lucru ($pitch = 180\,°$).

```
.DO ALIGN    ; poziționare la pitch=180 (doar pt. Viper)
```

- Se alege punctul de rotație, precum și un reper fix în spațiul de lucru (de exemplu, se fixează în gripper o piesă cu o gaură circulară și se alege drept reper un ax cu diametrul egal cu al găurii). Se deplasează robotul (folosind MCP, comutatorul Slow activat) astfel încât cele două elemente (punctul de rotație și reperul) să coincidă, și se învață punctul `a`.

```
.DO SET old.tool = TOOL
.HERE a      ; învățare punct
```

- Se rotește gripper-ul cu 180 °, având grijă să se evite coliziunile. Se deplasează din nou robotul folosind MCP-ul, *doar în translație*, astfel încât punctul de rotație să coincidă din nou cu reperul ales. Se învață punctul `b`.

```
.DO DEPARTS 50
.DO MOVES HERE:RZ(180)
; deplasare robot folosind MCP, doar pe translație
.HERE b
.DO DEPARTS 50
```

Prin calculul "mediei" dintre cele două locații se obține punctul de referință `ref.loc`. În locul funcției `HERE` din metoda generală, se va folosi punctul `a`.

Punctul de referință `ref.loc` va avea orientarea și coordonata Z identice cu ale lui `a`. Astfel, transformarea Tool va fi o translație, având componente nenule doar pe axele X și Y.

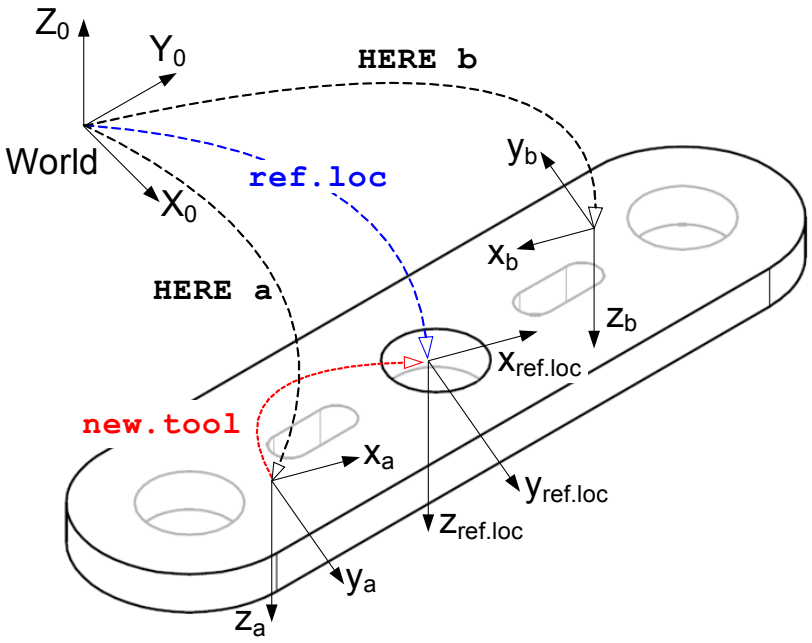

Figura 4.8: *Calculul transformării Tool pentru gripper excentric*

În continuare se excută următorii pași:

- Se calculează `ref.loc` și `new.tool`

```
.DO SET a.rot = SHIFT(a BY -DX(a), -DY(a), -DZ(a))
.DO SET ref.loc = TRANS((DX(a) + DX(b)) / 2,
                        (DY(a) + DY(b)) / 2,
                         DZ(a)):a.rot
.DO SET new.tool = old.tool:INVERSE(a):ref.loc
```

- Verificare: `new.tool` trebuie să difere, față de `old.tool`, doar pe X și Y:

```
; se compară old.tool și new.tool:
.LISTL new.tool, old.tool

; sau se listează "diferența" între new.tool și old.tool
.LISTL INVERSE(old.tool):new.tool
```

  Expresia `INVERSE(old.tool):new.tool` trebuie să aibă componente nenule doar pe X și Y.

- Se aplică transformarea `new.tool`:

  ```
  .TOOL new.tool
  ```

Se verifică transformarea Tool rotind gripper-ul de la MCP în jurul axei Z. Dacă rotația este efectuată în jurul punctului ales, calculele au fost corecte.

Întrebare: Dacă se alege punctul `b` pentru a obține orientarea și coordonata Z a punctului `ref.loc`, se va obține aceeași transformare Tool în urma calculelor, sau se va obține o transformare diferită?

Exercițiu

Se dorește deplasarea robotului în prima locație învățată (`loc.a`). După cum am spus mai devreme, `loc.a` nu mai este valabilă, deoarece am activat noua transformare Tool. Calculați `loc.a.new` în funcție de `loc.a` și `tool.trans` astfel încât robotul să poată fi dus în poziția dorită prin comanda `MOVE loc.a.new`.

Exercițiu

Extindeți metoda de calcul de mai sus pentru a funcționa și în cazul rotirii piesei cu un unghi RZ diferit de $180\,^\circ$.

Exercițiu

Cunoaștem două posibilități de a determina punctul `ref.loc`, în funcție de efectul pe care dorim să îl obținem:

- pentru reorientarea transformării Tool, fără a altera originea;
- pentru mutarea axei Z a transformării Tool în centrul unui reper circular aflat pe piesa din gripper, fără a altera orientarea;

Propuneți o metodă pentru determinarea lui `ref.loc` astfel încât să se obțină simultan cele două rezultate de mai sus. Astfel, transformarea `new.tool` calculată cu `ref.loc` va trebui să mute axa Z într-una din găurile piesei din gripper și în același timp să alinieze axele sistemului Tool cu vectorii $\vec{n}$, $\vec{s}$ și $\vec{a}$ din Fig. 4.5 (a).

Aplicații pentru laborator

Paletizare pe plan înclinat

Determinați transformarea Base (`bs`) pentru o paletă aflată pe un plan înclinat și rulați programul `pal.inc`.

Spirala

Se consideră problema *Spirala* din laboratorul 3.

Învățați o transformare Tool care să poziționeze axa Z_{tool} în centrul unei piese de tip I. Cu această transformare, învățați din nou punctele de prindere `a` și `b`, apoi rulați programul. Poziția fizică de prindere în gripper poate să fie excentrică.

Căsuța

Se consideră problema *Căsuța* din laboratorul 3.

Învățați o transformare Tool care să poziționeze axa Z_{tool} în centrul unei piese de tip I, și de asemenea să alinieze axa X_{tool} cu axa principală (de inerție minimă) a piesei. Cu această transformare, învățați din nou punctele de prindere `a` și `b`, setați orientarea punctului `c` la (0, 180, 0) și rulați programul.

Rotiți punctul `c` cu un unghi oarecare în jurul axei Z și modificați programul astfel încât să funcționeze corect, construind o "căsuță" rotită, de ex. la 45 °.

Rulați programul "căsuța rotită" cu axuri montate pe paleta destinație, în cele 4 colțuri. Pentru aceasta se va reînvăța punctul `c` pe paletă, montat în 2 axuri. Paleta este rotită în jurul axei Z_{world} cu un unghi diferit de 0.

Probleme propuse

Paletizare 2D cu transformarea Base

Se consideră problema de paletizare din laboratorul 3, cazul în care paleta este rotită în jurul axei Z cu un unghi necunoscut. Definiți un sistem de coordonate local pe paletă (`bs`) folosind funcția `FRAME`. Este necesară învățarea altor puncte robot în afară de `pal` și `pal.x`? Dacă da, care sunt acestea? Dacă nu, este posibilă calcularea acestora doar în funcție de `pal` și `pal.x`? Cum?

Comparați soluția obținută prin metoda transformării Base cu rezolvarea din laboratorul 3, programul `paletizare`.

Paletizare cu piese de tip `r` rotite

Se dorește așezarea a `n = 12` piese de tip `r` într-un depozit cu 3D, $2 \times 2 \times 3$ ca în Fig. 4.9. Piesele se află în stiva verticală `st`.

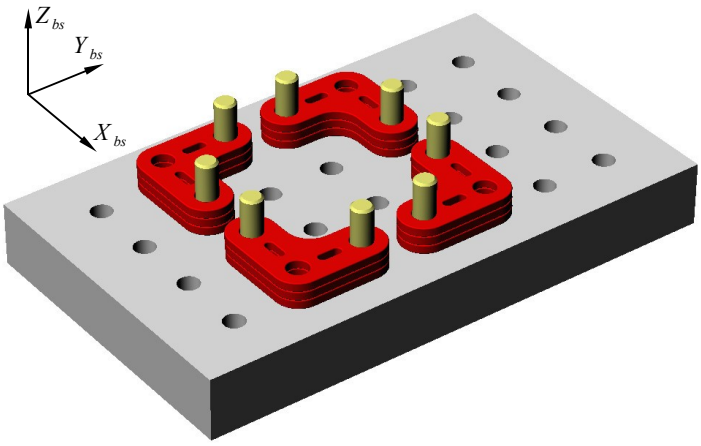

Figura 4.9: *Paletizare cu piese de tip `r` rotite*

Se cunosc:

- `st`: baza stivei verticale, învățată în World;
- `bs`: sistemul de coordonate local al paletei;
- `dz = 4.25 mm`: înălțimea unei piese;
- `dx = dy = 31.75 mm`: distanța între 2 găuri consecutive pe paletă;
- `#safe`;
- Configurația de lucru a robotului: `LEFTY/ABOVE/NOFLIP`.

Punctul `pal` nu este învățat.

Cerințe:

- Alegeți originea paletei (prima poziție), direcțiile pentru linii și coloane (i și j) și ordinea de completare;
- Alegeți o transformare Tool convenabilă;
- Specificați (prin desen) ce puncte robot vor fi învățate pentru a obține transformarea Tool aleasă mai sus, și unde va fi învățat punctul `pal`;
- Modificați programul de paletizare, astfel încât să se realizeze montajul din Fig. 4.9.

Observație: Pe paletă se va învăța un singur punct de prindere.