ROBOTOK PROGRAMOZÁSA

1. ROBOT KINEMATIKAI ALAPOK

A robot felfogható egy irányított mechanizmusnak, amely előírható pályán mozoghat, és a pálya mentén előírható feladatokat láthat el. A robot szegmensekből áll, melyeket transzlációs vagy rotációs csuklók kapcsolnak össze, a robot mozgása pedig a fizika törvényeinek van alávetve. Működését az irányítás különböző szintjen különféle modellekkel (kinematikai, dinamikus) írhatjuk le. A robotok programozásakor ezek közül a legegyszerűbb kinematikai (geometriai) modell játszik szerepet. Ebben alapvető fontosságú a koordináta-rendszerek közötti kapcsolat leírása.

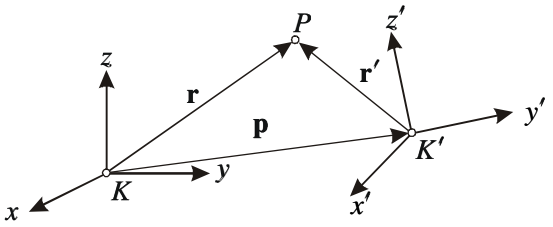
Alkalmazói szempontból ugyanis merev robotok esetén, amelyeknél a rugalmas alakváltozás elhanyagolhatóan kicsi a robot elvárt pontosságához képest, a robot által elvégzendő mozgás definiálásakor alapul választhatunk egy bázis koordináta-rendszert (szokás világ koordináta-rendszernek is nevezni), amelyhez előírjuk, hol legyen a robot munkavégző eszköze, az ún. végberendezés (end effector) a mozgás soronkövetkező fázisában a bázis koordináta-rendszerhez képest. Mivel azonban nem csak a végberendezés (megfogó, szerszám, technológiai eszköz: festékszóró pisztoly, pont- vagy ívhegesztő stb.) pozíciója, hanem annak orientációja is fontos (gondoljunk arra, hogy ha a megfogó egy vízzel teli pohart tart, akkor nem mindegy, hogy a pohár szája felfelé vagy lefelé néz), ezért kézenfekvő a végberendezéshez is egy koordináta-rendszert rögzíteni, és a két koordináta-rendszer közötti relatív pozíciót és orientációt előírni. Ez azt jelenti, hogy van két derékszögű koordináta-rendszer, a K_B bázis koordináta-rendszer és a K_E end effector koordináta-rendszer, és a mozgás definiálása során azt mondjuk meg, hol legyen K_E origója és hogyan álljanak a K_E egységvektorai a világ koordináta-rendszer origójához és egységvektoraihoz képest. Az előbbit a végberendezés pozíciójának (szerszám középpont, TCP, Tool Center Point), az utóbbit az orientációjának nevezzük.

Világos, hogy a robot szegmenseinek egymáshoz képesti relatív elhelyezkedése is hasonló elven alapulva írható le. Rögzíthetünk ugyanis a kinematikai láncban egy K_{i-1} koordináta-rendszert a megelőző szegmenshez, a kinematikai láncban soronkövetkező szegmenshez pedig egy K_i koordináta-rendszert, és mozgásukat leírhatjuk a két koordináta-rendszer relatív mozgása segítségével. Fizikailag a szegmenseket rendszerint motorok mozgatják fogaskerék vagy fogasléc áttételen keresztül, a motor elhelyezhető a megelőző szegmensen, az áttétel tengelye pedig tekinthető a következő szegmenst mozgató (R rotációs, vagy T transzlációs) csukló tengelyének. A K_{i-1} koordináta-rendszer z_{i-1} tengelye választható a soron következő

szegmenst mozgató motor áttételének tengelyeként, a K_i koordináta-rendszer x_i tengelye pedig merőlegesnek a megelőző (saját) és a következő csuklótengelyre. A K_i koordináta-rendszer z_i tengelye választható a következő csuklótengelynek, amelyen x_i kimetszi a K_i origóját. A részleteket a robot Denavit-Hartenberg paramétereinek tárgyalásakor fogjuk megismerni. Előbb azonban összefoglaljuk a fizikai rendszerek (robotok, repülőgépek, helikopterek stb.) geometriai/kinematikai modelljeiben szerepet játszó különféle orientáció jellemzéseket, valamint a pozíció és orientáció leírását homogén transzformációval.

1.1 Kinematikai alapfogalmak

Tekintsünk két (jobb sodrású) derékszögű koordináta-rendszert (röviden keretet, angolul frame), jelölje ezeket K és K', és legyenek a bázisvektoraik rendre i, j, k és i', j', k' (lásd *1.1. ábra*).



1.1. ábra. Keret (frame) koncepció

Határozzuk meg a koordináta-rendszerek vektorai közötti kapcsolatot:

$$\begin{aligned} r' &= \rho'_x i' + \rho'_y j' + \rho'_z k', \\ i' &= a_{11} i + a_{21} j + a_{31} k, \\ j' &= a_{12} i + a_{22} j + a_{32} k, \\ k' &= a_{13} i + a_{23} j + a_{33} k, \\ p &= p_x i + p_y j + p_z k, \end{aligned} \tag{1.1}$$

$$\begin{aligned} r = \rho_x i + \rho_y j + \rho_z k = r' + p = (a_{11}\rho'_x + a_{12}\rho'_y + a_{13}\rho'_z + p_x) i + \\ + (a_{21}\rho'_x + a_{22}\rho'_y + a_{23}\rho'_z + p_y) j + (a_{31}\rho'_x + a_{32}\rho'_y + a_{33}\rho'_z + p_z) k . \end{aligned}$$

Ezek az összefüggések mátrix alakban is felírhatók:

$$\begin{pmatrix}\rho_x\\ \rho_y\\ \rho_z\end{pmatrix}=\underbrace{\begin{pmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{pmatrix}}_{\substack{\uparrow i' \quad \uparrow j' \quad \uparrow k'\\ \text{felírva az}\\ i,j,k \text{ bázisban}}}\begin{pmatrix}\rho'_x\\ \rho'_y\\ \rho'_z\end{pmatrix}+\begin{pmatrix}p_x\\ p_y\\ p_z\end{pmatrix}, \tag{1.2}$$

$$r = Ar' + p \tag{1.3}$$

Hozzávéve ehhez az $1=1$ azonosságot, a K és K' keretek közötti A orientáció változást és a keretek origója közötti p pozíció vektort összevonhatjuk egy T homogén transzformációban:

$$\begin{pmatrix}\rho_x\\ \rho_y\\ \rho_z\\ 1\end{pmatrix}=\begin{pmatrix}a_{11} & a_{12} & a_{13} & p_x\\ a_{21} & a_{22} & a_{23} & p_y\\ a_{31} & a_{32} & a_{33} & p_z\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\rho'_x\\ \rho'_y\\ \rho'_z\\ 1\end{pmatrix}, \tag{1.4}$$

$$\begin{pmatrix}\rho\\ 1\end{pmatrix}=\left(\begin{array}{c|c}A & p\\ \hline 0^T & 1\end{array}\right)\begin{pmatrix}\rho'\\ 1\end{pmatrix}=:\begin{pmatrix}l & m & n & p\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\rho'\\ 1\end{pmatrix}=:T\begin{pmatrix}\rho'\\ 1\end{pmatrix}. \tag{1.5}$$

$$r = Tr'. \tag{1.6}$$

Itt $r\in R^4$ és $r'\in R^4$ jelölik $\rho\in R^3$ és $\rho'\in R^3$ homogén koordinátáit.

A T homogén transzformáció két fontos jelentéssel rendelkezik:

i) T megadja K' orientációját és pozícióját K-hoz képest.
ii) T lehetővé teszi a P pont koordinátáinak meghatározását K-ban, ha ismerjük a koordinátáit K'-ben.

Célszerű a T homogén transzformációt indexekkel ellátni annak kihangsúlyozására, hogy a K'-höz tartozó l,m,n,p komponenseit a K bázisában kell megadni: $T_{K,K'}$. Jól látható, hogy az indexek sorrendje az értelmezésben lényeges.

Homogén transzformációk egymásutánja a homogén transzformációk szorzatával írható le:

$$T_1T_2=\left[\begin{array}{c|c}A_1 & p_1\\ \hline 0^T & 1\end{array}\right]\left[\begin{array}{c|c}A_2 & p_2\\ \hline 0^T & 1\end{array}\right]=\left[\begin{array}{c|c}A_1A_2 & A_1p_2+p_1\\ \hline 0^T & 1\end{array}\right]. \tag{1.7}$$

1.1.1 Az orientáció jellemzése forgatásokkal

Tekintsük először csak az *orientáció* megváltozását. Tegyük fel, hogy eredetileg K és K' egybeestek, és K valamelyik tengelye körül K' el lett forgatva φ szöggel (a pozitív forgásirány a jobb csavar-szabály szerint értelmezett), és tekintsük az elforgatás utáni állapotot. Bevezetjük a következő konvenciót a jelölésekre:

$$C_\varphi := \cos\varphi\,,\ S_\varphi := \sin\varphi\,,\ C_{\alpha\beta} := \cos(\alpha+\beta)\,,\ C_{12} := \cos(q_1+q_2)\ \text{stb.} \tag{1.8}$$

Akkor (1.2) szerint az elemi forgatások (rotation) következők lesznek:

$$Rot(z,\varphi) = \underbrace{\left(i'_{\text{rot}}\ j'_{\text{rot}}\ k'_{\text{rot}}\right)}_{\text{kifejezve } i,j,k\text{-ban}} = \begin{bmatrix}\cos\varphi & -\sin\varphi & 0\\ \sin\varphi & \cos\varphi & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}C_\varphi & -S_\varphi & 0\\ S_\varphi & C_\varphi & 0\\ 0 & 0 & 1\end{bmatrix} \tag{1.9}$$

$$Rot(y,\varphi) = \left(i'_{\text{rot}}\ j'_{\text{rot}}\ k'_{\text{rot}}\right) = \begin{bmatrix}C_\varphi & 0 & S_\varphi\\ 0 & 1 & 0\\ -S_\varphi & 0 & C_\varphi\end{bmatrix} \tag{1.10}$$

$$Rot(x,\varphi) = \left(i'_{\text{rot}}\ j'_{\text{rot}}\ k'_{\text{rot}}\right) = \begin{bmatrix}1 & 0 & 0\\ 0 & C_\varphi & -S_\varphi\\ 0 & S_\varphi & C_\varphi\end{bmatrix} \tag{1.11}$$

Legyen t általános irányvektor, $\|t\| = 1$. Tegyük fel, hogy eredetileg K és K' egybeestek, és forgassuk el K'-t a t tengely körül φ szöggel. A forgatás egy $\mathsf{Rot}(t,\varphi)$ lineáris transzformáció, amelynek mátrixa $Rot(t,\varphi)$:

$$x^{\text{rot}} = \mathsf{Rot}(t,\varphi)x = x\cos\varphi + t{<}t,x{>}(1-\cos\varphi) + t\times x\sin\varphi \tag{1.12}$$

$$Rot(t,\varphi) = C_\varphi I + (1-C_\varphi)[t\circ t] + S_\varphi[t\times]\,. \tag{1.13}$$

Az összefüggés neve *Rodriguez-képlet*. A $[t\circ t]$ diadikus szorzat és a $[t\times]$ vektorszorzat mátrixának felhasználásával a Rodriguez-képlet mátrix alakban:

$$\begin{aligned} Rot(t,\varphi) &= \begin{pmatrix} l_x & m_x & n_x\\ l_y & m_y & n_y\\ l_z & m_z & n_z\end{pmatrix} = C_\varphi I + (1-C_\varphi)[t\circ t] + S_\varphi[t\times] = \\ &= \begin{bmatrix} C_\varphi + (1-C_\varphi)t_x t_x & (1-C_\varphi)t_x t_y - S_\varphi t_z & (1-C_\varphi)t_x t_z + S_\varphi t_y\\ (1-C_\varphi)t_x t_y + S_\varphi t_z & C_\varphi + (1-C_\varphi)t_y t_y & (1-C_\varphi)t_y t_z - S_\varphi t_x\\ (1-C_\varphi)t_x t_z - S_\varphi t_y & (1-C_\varphi)t_y t_z + S_\varphi t_x & C_\varphi + (1-C_\varphi)t_z t_z\end{bmatrix}. \end{aligned} \tag{1.14}$$

A forgatás inverze a visszaforgatás, tehát a forgatás t körül $-\varphi$ szöggel. Tudjuk, hogy az (1.14) egyenletben szereplő $[t\circ t]$ szimmetrikus, $[t\times]$ antiszimmetrikus, $\cos(-\varphi)=\cos(\varphi)$ és $\sin(-\varphi)=-\sin(\varphi)$, ezért

$$[Rot(t,\varphi)]^{-1}=[Rot(t,\varphi)]^{T}. \tag{1.15}$$

Ortonormált derékszögű koordináta-rendszerek esetén tehát az A orientációs mátrix inverze az A transzponáltja, és emiatt a T homogén transzformáció inverze is egyszerűen számítható:

$$A^{-1}=A^{T} \tag{1.16}$$

$$T=\left[\begin{array}{c|c} A & p \\ \hline 0^T & 1 \end{array}\right]^{-1}=\left[\begin{array}{c|c} A^T & -A^T p \\ \hline 0^T & 1 \end{array}\right]. \tag{1.17}$$

Az A orientációs rész a T homogén transzformációban $3\times 3=9$ elemet tartalmaz. Ezek azonban nem függetlenek, mert i',j',k' egy ortonormált bázis, tehát $\|i'\|=\|j'\|=\|k'\|=1$ és $<i',j'>=<i',k'>=<j',k'>=0$, ami 6 feltételt definiál a 9 elemre, és ezért az orientáció $9-6=3$ szabad paraméterrel jellemezhető. A Rodriguez-képlet esetén ez t és φ, ahol t egységvektor, és ezért csak 2 paramétere szabad.

A Rodriguez-képleten kívül az orientációnak más jellemzései is gyakoriak. Ha adott K és K', és az origók közösek, akkor vehetjük K-nak egy másik példányát, és ebből kiindulva elemi forgatások sorozatával eljuthatunk K'-be. A másodpéldány elforgatott tengelyeit jelölhetjük vesszővel és kétvesszővel, a forgatások szögeit pedig rendre φ,ϑ,ψ-vel. A *robotikában* megszokott terminológiát használva beszélhetünk *Euler-szögekről*, ha z,y',z'' a forgástengelyek, és *roll, pitch, yaw* (RPY) szögekről, ha a forgástengelyek z,y',x''. E szerint a konvenció szerint

$$\begin{aligned} A_{K,K'} &= Euler(\varphi,\vartheta,\psi)=Rot(z,\varphi)Rot(y',\vartheta)Rot(z'',\psi)= \\ &=\begin{bmatrix} C_\varphi & -S_\varphi & 0 \\ S_\varphi & C_\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C_\vartheta & 0 & S_\vartheta \\ 0 & 1 & 0 \\ -S_\vartheta & 0 & C_\vartheta \end{bmatrix}\begin{bmatrix} C_\psi & -S_\psi & 0 \\ S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}= \\ &=\begin{bmatrix} C_\varphi C_\vartheta C_\psi - S_\varphi S_\psi & -C_\varphi C_\vartheta S_\psi - S_\varphi C_\psi & C_\varphi S_\vartheta \\ S_\varphi C_\vartheta C_\psi + C_\varphi S_\psi & -S_\varphi C_\vartheta S_\psi + C_\varphi C_\psi & S_\varphi S_\vartheta \\ -S_\vartheta C_\psi & S_\vartheta S_\psi & C_\vartheta \end{bmatrix} \end{aligned} \tag{1.18}$$

$$A_{K,K'} = RPY(\varphi,\vartheta,\psi) = Rot(z,\varphi)Rot(y',\vartheta)Rot(x'',\psi) =$$

$$= \begin{bmatrix} C_\varphi & -S_\varphi & 0 \\ S_\varphi & C_\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_\vartheta & 0 & S_\vartheta \\ 0 & 1 & 0 \\ -S_\vartheta & 0 & C_\vartheta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\psi & -S_\psi \\ 0 & S_\psi & C_\psi \end{bmatrix} = \tag{1.19}$$

$$= \begin{bmatrix} C_\varphi C_\vartheta & C_\varphi S_\vartheta S_\psi - S_\varphi C_\psi & C_\varphi S_\vartheta C_\psi + S_\varphi S_\psi \\ S_\varphi C_\vartheta & S_\varphi S_\vartheta S_\psi + C_\varphi C_\psi & S_\varphi S_\vartheta C_\psi - C_\varphi S_\psi \\ -S_\vartheta & C_\vartheta S_\psi & C_\vartheta C_\psi \end{bmatrix}$$

1.1.2 Az orientáció jellemzése kvaterniókkal

Az orientáció jellemezhető *kvaterniókkal* is. A kvaterniók (Q) a háromdimenziós vektortér (R^3) kiterjesztését alkotják, amelyben szorzás $(*)$, összeadás, számmal szorzás, konjugált és norma van definiálva. Legyen $q \in Q$, $w \in R^3$, $s, \alpha \in R^1$, akkor definíció szerint

$$\begin{aligned} q &= (w, s) \\ \alpha q &= (\alpha w, \alpha s) \\ q_1 + q_2 &= (w_1, s_1) + (w_2, s_2) = (w_1 + w_2, s_1 + s_2) \\ q_1 * q_2 &= (w_1 \times w_2 + s_1 w_2 + s_2 w_1, s_1 s_2 - < w_1, w_2 >) \\ \tilde{q} &= (-w, s) \\ \|q\|^2 &= q * \tilde{q} = (0, \|w\|^2 + s^2) = \tilde{q} * q. \end{aligned} \tag{1.20}$$

Az alkalmazások szempontjából talán a legfontosabb tulajdonság, hogy

$$\|q_1 * q_2\| = \|q_1\| \cdot \|q_2\| . \tag{1.21}$$

Mivel $s \in R^1$ és $(0, s) \in Q$, továbbá $w \in R^3$ és $(w,0) \in Q$ azonosíthatók, ezért R^1 és R^3 be van ágyazva Q-ba. A kvaternió norma multiplikatív tulajdonsága alapján

$$\min_{\|q\|=1} \|q * r * \tilde{q}\| = \min_{\|q\|=1} (\|q * r\| \cdot \|q\|) = \min_{\|q\|=1} \|q * r\| \tag{1.22}$$

A kvaterniók és az A orientáció szoros kapcsolatban állnak. Minden A rotáció a Rodriguez-képlet szerint egy t egységvektor körüli φ szöggel való forgatás:

$$Ar = \cos\varphi \; r + (1 - \cos\varphi) < t, r > t + \sin\varphi \; t \times r . \tag{1.23}$$

Másrészt viszont bevezethetjük a

$$q := (\sin\frac{\varphi}{2}\ t, \cos\frac{\varphi}{2}) \tag{1.24}$$

kvaterniót ($\|q\| = \|t\| = 1$), ahonnan a Rodriguez-képlet felhasználásával kapjuk, hogy

$$q * (r,0) * \tilde{q} = (Ar,0), \tag{1.25}$$

ezért $q * r * \tilde{q}$ és Ar azonosíthatók. Világos, hogy ha az orientációt a $q = (w, s)$ kvaternióval jellemezzük, akkor ennek ismeretében A meghatározható, mert

$$\varphi = 2\arccos(s),\ t = w/\sin(\frac{\varphi}{2}) \Rightarrow A = C_\varphi I + (1 - C_\varphi)[t \circ t] + S_\varphi[t\times], \tag{1.26}$$

ahol

$$[t \circ t] = \begin{bmatrix} t_x t_x & t_x t_y & t_x t_z \\ t_y t_x & t_y t_y & t_y t_z \\ t_z t_x & t_z t_y & t_z t_z \end{bmatrix} \tag{1.27}$$

$$[t\times] = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}. \tag{1.28}$$

1.1.3 Az inverz orientációs feladat megoldása

Ha szükség van az A orientáció

$$A = \begin{bmatrix} l_x & m_x & n_x \\ l_y & m_y & n_y \\ l_z & m_z & n_z \end{bmatrix} \tag{1.29}$$

numerikus ismeretében az orientáció forgatásokkal való jellemzésének meghatározására (inverz orientációs feladat), akkor a három szokásos jellemzés esetén a következőképp járhatunk el.

Az inverz Rodriguez-feladat megoldása:

$$\begin{aligned} C_\varphi &= \frac{l_x + m_y + n_z - 1}{2} \\ S_\varphi &= \frac{+\sqrt{(m_z - n_y)^2 + (n_x - l_z)^2 + (l_y - m_x)^2}}{2} \\ \varphi &= \operatorname{atan2}\ (S_\varphi, C_\varphi) \in (-\pi, \pi] \end{aligned} \tag{1.30a}$$

$$t_x = \sqrt{\frac{l_x - C_\varphi}{1 - C_\varphi}}\,\text{sign}\,(m_z - n_y)$$
$$t_y = \sqrt{\frac{m_y - C_\varphi}{1 - C_\varphi}}\,\text{sign}\,(n_x - l_z) \qquad (1.30b)$$
$$t_z = \sqrt{\frac{n_z - C_\varphi}{1 - C_\varphi}}\,\text{sign}\,(l_y - m_x)$$

A feladatnak végtelen sok megoldása van, ha $\varphi = 0$ $(C_\varphi = 1)$. Valóban, nulla forgásszög esetén bármilyen t tengely megfelel.

Az inverz Euler-feladat megoldása:

i) $n_x^2 + n_y^2 \neq 0 \Rightarrow S_\vartheta \neq 0$

$$\varphi = \arctan\left(\frac{n_y}{n_x}\right) + \begin{Bmatrix} 0 \\ \pi \\ -\pi \end{Bmatrix} \in (-\pi, \pi], \qquad (1.31a)$$

$$\left.\begin{aligned} S_\vartheta &= C_\varphi n_x + S_\varphi n_y \\ C_\vartheta &= n_z \end{aligned}\right\} \quad \vartheta = \arctan\frac{C_\varphi n_x + S_\varphi n_y}{n_z}, \qquad (1.31b)$$

$$\left.\begin{aligned} S_\psi &= -S_\varphi l_x + C_\varphi l_y \\ C_\psi &= -S_\varphi m_x + C_\varphi m_y \end{aligned}\right\} \quad \psi = \arctan\frac{-S_\varphi l_x + C_\varphi l_y}{-S_\varphi m_x + C_\varphi m_y}. \qquad (1.31c)$$

ii) $n_x = n_y = 0 \Rightarrow S_\vartheta = 0$, *szinguláris konfiguráció*

a) $n_z = C_\vartheta = 1 \Rightarrow \vartheta = 0$ $(z = z')$

$$\left.\begin{aligned} C_{\varphi\psi} &= l_x \\ S_{\varphi\psi} &= l_y \end{aligned}\right\} \varphi + \psi = \arctan\left(\frac{l_y}{l_x}\right) \Rightarrow \text{ csak } \varphi + \psi \text{ határozható meg} \qquad (1.32)$$

b) $n_z = C_\vartheta = -1 \Rightarrow \vartheta = \pi$ $(z = -z')$

$$\left.\begin{aligned} S_{\psi-\varphi} &= m_x \\ C_{\psi-\varphi} &= m_y \end{aligned}\right\} \psi - \varphi = \arctan\left(\frac{m_x}{m_y}\right). \Rightarrow \text{ csak } \psi - \varphi \text{ határozható meg} \qquad (1.33)$$

Inverz RPY feladat megoldása:

i) $l_x^2 + l_y^2 \neq 0 \Rightarrow C_\vartheta \neq 0$

$$\varphi = \arctan\left(\frac{l_y}{l_x}\right) + \left\{\begin{array}{c} 0 \\ \pi \\ -\pi \end{array}\right\} \in (-\pi, \pi], \tag{1.34a}$$

$$\left.\begin{array}{l} S_\vartheta = -l_z \\ C_\vartheta = S_\varphi l_y + C_\varphi l_x \end{array}\right\} \vartheta = \arctan\frac{-l_z}{S_\varphi l_y + C_\varphi l_x}, \tag{1.34b}$$

$$\left.\begin{array}{l} S_\psi = S_\varphi n_x - C_\varphi n_y \\ C_\psi = -S_\varphi m_x + C_\varphi m_y \end{array}\right\} \psi = \arctan\frac{S_\varphi n_x - C_\varphi n_y}{-S_\varphi m_x + C_\varphi m_y}. \tag{1.34c}$$

ii) $l_x = l_y = 0 \Rightarrow C_\vartheta = 0$, *szinguláris konfiguráció*

a) $l_z = -S_\vartheta = 1 \Rightarrow \vartheta = -\pi/2 \;\; (x = x')$

$$\left.\begin{array}{l} S_{\varphi\psi} = -n_y \\ C_{\varphi\psi} = -n_x \end{array}\right\} \varphi + \psi = \arctan\left(\frac{n_y}{n_x}\right) \Rightarrow \text{ csak } \varphi + \psi \text{ határozható meg.} \tag{1.35}$$

b) $l_z = -S_\vartheta = -1 \Rightarrow \vartheta = \pi/2 \;\; (x = -x')$

$$\left.\begin{array}{l} S_{\psi-\varphi} = m_x \\ C_{\psi-\varphi} = m_y \end{array}\right\} \psi - \varphi = \arctan\left(\frac{m_x}{m_y}\right) \Rightarrow \text{ csak } \psi - \varphi \text{ határozható meg.} \tag{1.36}$$

1.2 Robotok kinematikai modellje

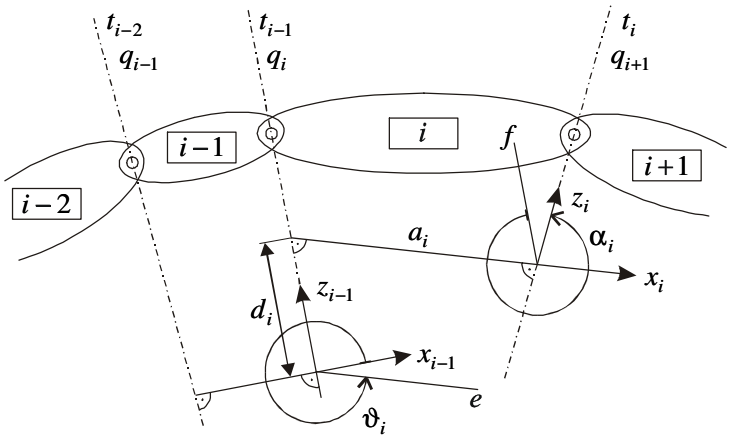
Számos, a gyakorlatban elterjedt ipari robot ún. nyíltláncú, elágazás nélküli merev robot, amelynek geometriai modelljét legtöbbször a Denavit–Hartenberg-alakkal írjuk le. A "merev robot" elnevezés idealizálás, és arra utal, hogy a robot garantált pontossági tartományában a rugalmas alakváltozások hatása a szegmensekben vagy a csuklókban elhanyagolható.

1.2.1 Csuklókkal összekapcsolt nyíltláncú merev testek geometriai modellje

Denavit–Hartenberg-alak (lásd *1.2. ábra*)

A robot szegmensekből (link) áll, amelyeket 1-szabadságfokú csuklók (joint) kötnek össze. A Denavit–Hartenberg-alak esetén a robot tengelyei z-tengely irányúak. A

csuklóváltozó (joint variable) d (transzlációs csukló esetén) vagy ϑ (rotációs csukló esetén). A Denavit–Hartenberg-paraméterek d, a (távolság) és ϑ, α (szög).



1.2. ábra. A Denavit–Hartenberg-alak

Tegyük fel, hogy a K_{i-1} keretet már felvettük. Az *1.2. ábrába* berajzoltuk a z_{i-1} és x_{i-1} tengelyeket, ahol z_{i-1} iránya azonos az $\boxed{i-1}$ és az $\boxed{i}$ szegmenst összekötő t_{i-1} csuklótengely (joint axis) irányával. Legyen t_i az i-edik és az $i+1$-edik szegmenst összekötő csuklótengely iránya. Legyen a K_i keret x_i tengelye merőleges a t_{i-1} és t_i tengelyekre. Legyen K_i origója az a pont, ahol x_i és t_i metszi egymást. Legyen e párhuzamos x_i-vel és menjen át K_{i-1} origóján, és legyen f párhuzamos z_{i-1}-gyel és menjen át K_i origóján. Akkor kitérő t_{i-1} és t_i tengelyek esetén

i) ϑ_i a z_{i-1} tengely körüli forgatás szöge, amely az x_{i-1} tengelyt e-be forgatja ($e \parallel x_i$);
ii) d_i az eltolás a z_{i-1} tengely mentén, amely az e egyenest x_i-be mozgatja;
iii) a_i az eltolás az x_i tengely mentén, amely z_{i-1} és x_i metszéspontját K_i origójába mozgatja;
iv) α_i az x_i tengely körüli forgatás, amely az f egyenest z_i-be forgatja.

A távolságok előjele pozitív, ha az eltolás $+z_{i-1}$ vagy $+x_i$ irányban történik. Hasonlóan a forgásszög pozitív, ha a forgatás z_{i-1} vagy x_i körül a jobb csavar-szabály szerint pozitív. A K_{i-1} és K_i közötti homogén transzformáció

$$T_{i-1,i} = Rot(z,\vartheta_i)\cdot Trans(z,d_i)\cdot Trans(x,a_i)\cdot Rot(x,\alpha_i) =$$

$$= \begin{bmatrix} C_{\vartheta_i} & -S_{\vartheta_i} & 0 & 0 \\ S_{\vartheta_i} & C_{\vartheta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_{\alpha_i} & -S_{\alpha_i} & 0 \\ 0 & S_{\alpha_i} & C_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} C_{\vartheta_i} & -S_{\vartheta_i}C_{\alpha_i} & S_{\vartheta_i}S_{\alpha_i} & a_iC_{\vartheta_i} \\ S_{\vartheta_i} & C_{\vartheta_i}C_{\alpha_i} & -C_{\vartheta_i}S_{\alpha_i} & a_iS_{\vartheta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (1.37)$$

Ha t_{i-1} és t_i párhuzamosak, akkor végtelen sok közös normálisuk van. Ekkor rotációs csukló esetén $d_i = 0$ választható, transzlációs csukló esetén pedig K_i origója azonosra választható a K_{i+1} keret origójával.

Ha t_{i-1} és t_i metszik egymást, akkor választható $a_i = 0$, és x_i párhuzamos $\mathrm{k}_{i-1} \times \mathrm{k}_i$ irányával.

A $\boxed{0}$ szegmens esetén z_0 egybeesik a t_0 tengellyel és x_0 merőleges z_0-ra.

Az utolsó, $\boxed{m}$ szegmens esetén z_m iránya tetszőleges és x_m merőleges a z_{m-1} és z_m irányokra.

Néhány speciális esetben (Descartes-féle TTT pozícionálás első x tengelye, vagy RPY orientáció utolsó x tengelye esetén) előnyösebb lehet, ha nem a z, hanem az x tengelyt választjuk azanosnak a csuklótengellyel. A $T_{i-1,i}$ homogén transzformáció továbbra is érvényben marad, de a csuklóváltozó a_i vagy α_i lesz.

A *1.1 és 1.2 táblázatok* a bemutatják a SCARA és PUMA robotok Denavit–Hartenberg-paramétereit.

1.1 táblázat. A SCARA RRTR robot Denavit–Hartenberg-paraméterei

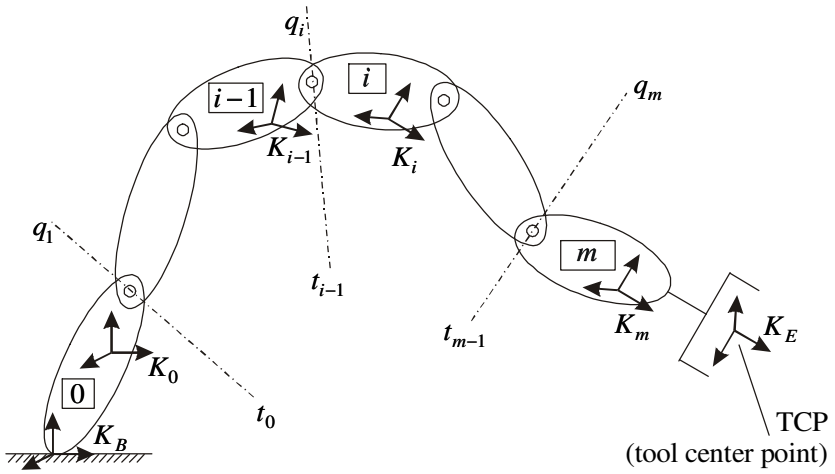
i	q_i	ϑ_i	d_i	a_i	α_i
1	ϑ_1	ϑ_1	d_1	a_1	0°
2	ϑ_2	ϑ_2	d_2	a_2	0°
3	d_3	0°	d_3	0	0°
4	ϑ_4	ϑ_4	0	0	0°

1.2 táblázat. A PUMA RRRRRR robot Denavit–Hartenberg-paraméterei

i	q_i	ϑ_i	d_i	a_i	α_i
1	ϑ_1	ϑ_1	0	0	–90°
2	ϑ_2	ϑ_2	0	a_2	0°
3	ϑ_3	ϑ_3	d_3	a_3	90°
4	ϑ_4	ϑ_4	d_4	0	–90°
5	ϑ_5	ϑ_5	0	0	90°
6	ϑ_6	ϑ_6	d_6	0	0°

1.2.2 Direkt geometriai feladat

Egy nyíltláncú, elágazás nélküli merev robot álljon a $\boxed{0}$, $\boxed{1}$, ..., $\boxed{m}$ szegmensekből (links) és a végberendezésből (end effector), amely lehet megfogó, szerszám stb. (gripper, tool). A szomszédos szegmenseket transzlációs vagy rotációs csuklók (prismatic or revolute joints) kötik össze. A transzlációs csuklók a szegmensnek a csuklótengely mentén történő elmozdulását, a rotációs csuklók a tengely körüli elfordulását teszik lehetővé. Legyen d vagy a a transzlációs csukló (T) mentén az elmozdulás. Hasonlóan jelölje ϑ vagy α a rotációs csukló (R) körüli forgásszöget. Közös nevük a q csuklóváltozó (joint variable).



1.3. ábra. Nyíltláncú, elágazás nélküli robot elvi sémája

A "nyíltláncú, elágazás nélküli robot" elnevezés arra utal, hogy a szegmensek megszámozhatók úgy, hogy minden szegmenst pontosan egy szegmens követ a szá-

mozás sorrendjében (az $\boxed{m}$ szegmenst a végberendezés követi), és az $\boxed{i}$ szegmens pozíciója és orientációja csak a megelőző szegmens pozíciójától és orientációjától, valamint az $\boxed{i-1}$ és $\boxed{i}$ szegmens közötti q_i csuklóváltozótól függ. A megelőző szegmensek definiálják az $\boxed{i-1}$ szegmenshez rögzített K_{i-1} keret pozícióját és orientációját, míg q_i definiálja az $\boxed{i}$ szegmenshez rögzített K_i keret relatív pozícióját és orientációját K_{i-1}-hez képest, amely a $T_{K_{i-1},K_i} := T_{i-1,i}$ homogén transzformációval jellemezhető. Az $\boxed{i-1}$ és $\boxed{i}$ szegmensek közötti csuklótengelyt t_{i-1} jelöli (lásd *1.3. ábra*).

A K_B keret a robot alapzatához van rögzítve. A K_0 keret a robot referencia kerete, amely azonos is lehetne K_B-vel, de rendszerint mégis eltér attól annak érdekében, hogy a csuklóvátozóktól való függés egyszerűbb legyen. Az irányítással befolyásolható csuklóváltozók a K_0 és K_m keretek között vannak. A K_B és K_0, valamint a K_m és K_E közötti kapcsolat nem függ a csuklóváltozóktól. A csuklók mind *egyszabadságfokúak* (one degree of freedom, 1–DOF) és a teljes robot *m*–DOF. A robot egyik lényeges jellemzője a *csuklóképlet* (joint formula). Például RRTRRR egy 6–DOF robotot jelöl, ahol minden csukló rotációs, kivéve a harmadikat, amelyik transzlációs. Annak érdekében, hogy a végberendezésnek mind a pozícióját, mind pedig az orientációját is megszorítás nélkül irányítani tudjuk, legalább 6 csukló szükséges $(m \geq 6)$. A robot még ebben az esetben is csak egy *korlátos munkatérben* (limited workspace) pozícionálható a szegmensek véges mérete és a csuklóváltozók limitált értéktartománya miatt.

Humán analógiák alapján beszélhetünk törzsről ($\boxed{0}$ trunk), karról ($\boxed{1}$, $\boxed{2}$, $\boxed{3}$ arm), kézcsuklóról ($\boxed{4}$, $\boxed{5}$, $\boxed{6}$ wrist) és kézről (hand, gripper). A végberendezés rendszerint cserélhető és az $\boxed{m}$ szegmenshez rögzíthető, kitüntetett pontja pedig a *szerszámközéppont* (tool center point, TCP).

Az $\boxed{i-1}$ szegmenshez rögzített K_{i-1} keret bárhol lehetne. A gyakorlatban gyakran használjuk a Denavit–Hartenberg konvenciót, ahol $\vartheta_i, d_i, a_i, \alpha_i$ határozza meg a K_{i-1} és K_i közötti $T_{i-1,i}$ homogén transzformációt. Ha kiválasztjuk a $\boxed{j}$ szegmens egy P_j pontját, amelynek x_j, y_j, z_j a koordinátái a K_j keretben, akkor ugyanennek a pontnak a koordinátái a K_{j-1} keretben x_{j-1}, y_{j-1}, z_{j-1}, amelyekre teljesül

$$\begin{pmatrix} x_{j-1} \\ y_{j-1} \\ z_{j-1} \\ 1 \end{pmatrix} = T_{j-1,j} \begin{pmatrix} x_j \\ y_j \\ z_j \\ 1 \end{pmatrix}. \tag{1.38}$$

Lépésről lépésre alkalmazván ezt az összefüggést kapjuk, hogy ugyanennek a pontnak az x_i, y_i, z_i koordinátái a K_i keretben

$$\begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = T_{i,i+1} \cdot T_{i+1,i+2} \cdot \ldots \cdot T_{j-1,j} \begin{pmatrix} x_j \\ y_j \\ z_j \\ 1 \end{pmatrix} = T_{i,j} \begin{pmatrix} x_j \\ y_j \\ z_j \\ 1 \end{pmatrix}, \tag{1.39}$$

ahol

$$T_{i,\,j} = T_{i,\,i+1} \cdot T_{i+1,\,i+2} \cdot \ldots \cdot T_{j-1,\,j}. \tag{1.40}$$

Itt $T_{i,j}$ írja le a koordináta-transzformációt, továbbá K_j relatív pozícióját és orientációját K_i-hez képest. Speciálisan

$$T_{0,m} = T_{0,1} \cdot T_{1,2} \cdot \ldots \cdot T_{m-1,m} = \left(\begin{array}{c|c} A_{0,m} & p_{0,m} \\ \hline 0^T & 1 \end{array} \right), \tag{1.41}$$

$$T_{B,E} = T_{B,0} \cdot T_{0,m} \cdot T_{m,E}. \tag{1.42}$$

Vegyük észre, hogy

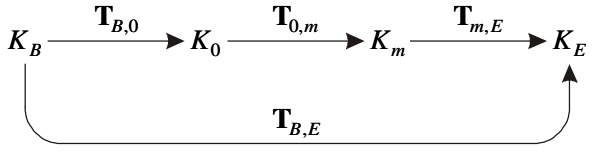
$$A_{0,m} = A_{0,1} \cdot A_{1,2} \cdot \ldots \cdot A_{m-1,m}, \tag{1.43}$$

$$A_{0,m}^{-1} = A_{m-1,m}^{T} \cdot \ldots \cdot A_{1,2}^{T} \cdot A_{0,1}^{T}, \tag{1.44}$$

$$T_{j,i} := T_{i,j}^{-1} = T_{j-1,j}^{-1} \cdot \ldots \cdot T_{i+1,i+2}^{-1} \cdot T_{i,i+1}^{-1}, \tag{1.45}$$

$$T_{0,m} = T_{B,0}^{-1} \cdot T_{B,E} \cdot T_{m,E}^{-1}. \tag{1.46}$$

Az utolsó egyenlet azt sugallja, hogy a robot egy ún. *robot gráffal* jellemezhető, lásd *1.4. ábra.*



1.4. ábra. Robot gráf

A robot gráf egy cimkézett irányított gráf. A gráf csúcsai a keretek, a kereteket élek kötik össze, az élek cimkéi a homogén transzformációk, az élek irányítása balról jobbra megfelel az indexelésnek.

K_0-ból K_m-be két különböző úton juthatunk el:

i) előrefelé haladva K_0, K_m sorrendben,

ii) hátrafelé haladva K_0, K_B, K_E, K_m sorrendben.

Ha a nyíllal szembe haladást invertálásként interpretáljuk, akkor visszakapjuk az (1.46) egyenletet, feltéve, hogy rendre vesszük a homogén transzformációt vagy annak inverzét az élek mentén és összeszorozzuk azokat. A robot gráf általánosítható összetettebb esetekre is (robot, szállítószalag, tárgy, kamera).

Szokás direkt geometriai feladatról beszélni, ha arra keresünk választ, hogyan függ $T_{B,E}$ vagy $T_{0,m}$ a $q = (q_1, \ldots, q_m)^T$ csuklóváltozó vektortól. Rendszerint az összefüggést szimbolikus (képletszerű) alakban keressük a robot Denavit–Hartenberg-paraméterei és (1.41) felhasználásával:

$$\text{direkt geometriai feladat: } q \mapsto T_{0,m}. \tag{1.47}$$

A szimbolikus alak azért szükséges, hogy számítási módszert (lehetőleg analitikus megoldást) tudjunk adni a később vizsgálandó inverz geometriai feladat megoldására.

1.2.3 Inverz geometriai feladat

Az inverz geometriai feladat célja a csuklóváltozók meghatározása, ha adott a végberendezés pozíciója és orientációja. A csuklóváltozók felhasználhatók referencia jelként (időfüggő alapjelként) a tengelyek szervohajtásai számára. A problémára mint

$$q = \operatorname{solve} T_{0,m} \tag{1.48}$$

fogunk hivatkozni. A feladat transzcendens egyenletrendszer megoldását igényli, amelyet általában nem tudunk képletszerű (szimbolikus, analitikus) alakban megadni. A gyakorlat számára fontos egyik esetben azonban a feladat analitikusan is megoldható.

1.1 Tétel (inverz geometriai feladat megoldása dekompozícióval): Ha az utolsó három tengely rotációs és a tengelyek egy közös pontban metszik egymást, akkor az inverz geometriai feladat felbontható egy pozícionáló és egy orientáló részproblémára, amelyek szeparáltan megoldhatók.

Bizonyítás: Legyen például $m=6$ és a robot Denavit–Hartenberg-alakjában $a_4=a_5=a_6=0$ és $\alpha_6=0°$, akkor

$$
\begin{aligned}
T_{0,6} &= \begin{bmatrix} l_{0,6} & m_{0,6} & n_{0,6} & p_{0,6} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\
&= T_{0,3} \left[\begin{array}{ccc|c} & & & 0 \\ & A_{3,4} & & 0 \\ & & & d_4 \\ \hline 0^T & & & 1 \end{array}\right] \left[\begin{array}{ccc|c} & & & 0 \\ & A_{4,5} & & 0 \\ \hline 0^T & & & 1 \end{array}\right] \begin{bmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\
&= \begin{bmatrix} l_{0,3} & m_{0,3} & n_{0,3} & p_{0,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} & & & 0 \\ & A_{3,6} & & 0 \\ \hline 0^T & & & 1 \end{array}\right] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\
&= \left[\begin{array}{ccc|c} & & & p_{0,3}+d_4 n_{0,3}+d_6 n_{0,6} \\ & A_{0,6} & & \\ \hline 0^T & & & 1 \end{array}\right]
\end{aligned}
\tag{1.49}
$$

és ezért

$$p_{0,3}(q_1,q_2,q_3) + d_4 n_{0,3}(q_1,q_2,q_3) = p_{0,6} - d_6 n_{0,6}, \tag{1.50}$$

$$A_{3,6}(q_1,q_2,q_3) = A_{0,3}^T(q_1,q_2,q_3)\left[l_{0,6}\ m_{0,6}\ n_{0,6}\right]. \tag{1.51}$$

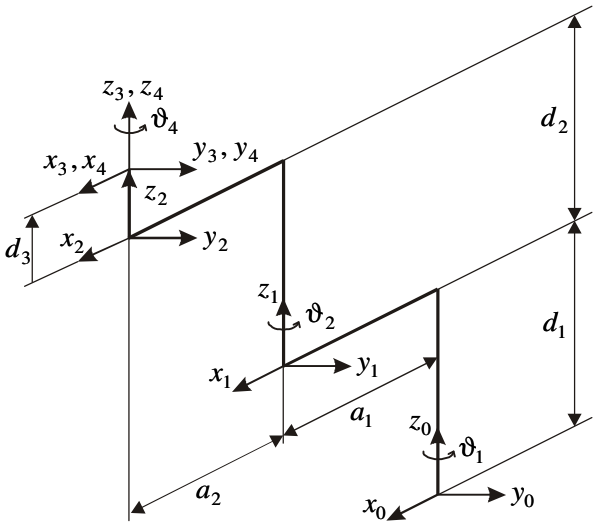
Az eredeti inverz problémát két 3-DOF részfeladatra bontottuk. Az első feladat (1.50), amely egy tiszta pozícionáló részfeladat, amelynek megoldása legyen q_1,q_2,q_3. Miután megoldottuk az első feladatot, kiszámíthatjuk (1.51) jobb oldalát és kereshetjük a második részfeladat megoldását a q_4, q_5, q_6 változókban, amely egy tiszta orientáló részfeladat. A második részfeladat elterjedt ipari robotoknál gyakran vezet az inverz Euler vagy az inverz RPY feladatra, amelyek megoldását korábban már tárgyaltuk.

1.2.4 A SCARA robot geometriai modellje

A direkt és inverz geometriai feladat megoldását a 4-DOF RRTR SCARA robot esetén mutatjuk be. A direkt feladat célja, hogy felderítsük a megfogó pozíciójának és orientációjának függését a csuklóváltozóktól. A rotációs csuklók miatt az összefüggés nemlineáris lesz. Másrészt az irányítások számára az szükséges, hogy ezt a nemlineáris összefüggést megfordítsuk, mivel az irányítási cél a csuklók olyan összehangolt mozgatása, amelynek eredményeképpen a megfogó felveszi a kívánt pozíciót és orientációt. A nemlineáris összefüggések miatt az inverz feladat egy nemlineá-

ris egyenletrendszer megoldása, ami tehát messze nem egy lineáris algebrai (mátrix inverziós) feladat. A problémát nehezíti, hogy az inverz feladat megoldása a valós időben realizálandó pályatervezés számára bemeneti adat, az inverz feladatot ezért lehetőleg iterációk nélkül és robusztus numerikus számítások bevonásával kell megoldani.

A SCARA robot Denavit–Hartenberg-paramétereit (lásd *1.1. táblázat*) az *1.5. ábra* koordináta-rendszerei alapján értelmeztük.



1.5. ábra. A SCARA robotnál használt keretek

Az egyes szegmensek közötti homogén transzformációk a következők:

$$T_{0,1} = \begin{bmatrix} C_1 & -S_1 & 0 & a_1 C_1 \\ S_1 & C_1 & 0 & a_1 S_1 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; T_{1,2} = \begin{bmatrix} C_2 & -S_2 & 0 & a_2 C_2 \\ S_2 & C_2 & 0 & a_2 S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.52}$$

$$T_{2,3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; T_{3,4} = \begin{bmatrix} C_4 & -S_4 & 0 & 0 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.53}$$

A homogén transzformációkat fokozatosan összeszorozzuk az eredő homogén transzformáció meghatározásához, ügyelve arra, hogy a mátrixszorzás nem kommutatív:

$$T_{2,4}=\begin{bmatrix} C_4 & -S_4 & 0 & 0 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ T_{1,4}=\begin{bmatrix} C_{24} & -S_{24} & 0 & a_2 C_2 \\ S_{24} & C_{24} & 0 & a_2 S_2 \\ 0 & 0 & 1 & d_2+d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.54}$$

$$T_{0,4}=\begin{bmatrix} C_{124} & -S_{124} & 0 & a_1 C_1 + a_2 C_{12} \\ S_{124} & C_{124} & 0 & a_1 S_1 + a_2 S_{12} \\ 0 & 0 & 1 & d_1+d_2+d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.55}$$

Itt $q \mapsto T_{0,4}$ a direkt geometriai feladat megoldása.

A $T_{0,4} \mapsto q$ inverz geometriai feladat megfogalmazásakor figyelembe kell venni, hogy a SCARA RRTR robot csak 4-szabadságfokú, ezért csak az eredő pozíciót és a megfogó x tengelyének irányát a vízszintes síkban írjuk elő:

$$a_1 C_1 + a_2 C_{12} = p_x, \tag{1.56}$$

$$a_1 S_1 + a_2 S_{12} = p_y, \tag{1.57}$$

$$d_1 + d_2 + d_3 = p_z, \tag{1.58}$$

$$C_{124} = l_x, \tag{1.59}$$

$$S_{124} = l_y. \tag{1.60}$$

A q_1 csuklóváltozó meghatározásához végezzük el a következő átalakításokat:

$$a_2 C_{12} = p_x - a_1 C_1,$$
$$a_2 S_{12} = p_y - a_1 S_1,$$

$$a_2^2 C_{12}^2 + a_2^2 S_{12}^2 = p_x^2 - 2a_1 p_x C_1 + a_1^2 C_1^2 + p_y^2 - 2a_1 p_y S_1 + a_1^2 S_1^2,$$
$$2a_1 p_x C_1 + 2a_1 p_y S_1 = p_x^2 + p_y^2 + a_1^2 - a_2^2,$$

ahol az utolsó egyenlet alakja $AC_\alpha + BS_\alpha = D$, amelynek megoldása

$$S_\alpha = \frac{DB + \delta_\alpha A\sqrt{A^2+B^2-D^2}}{A^2+B^2}, \tag{1.61}$$

$$C_\alpha = \frac{DA + \gamma_\alpha B\sqrt{A^2+B^2-D^2}}{A^2+B^2}. \tag{1.62}$$

Mivel teljesülnie kell, hogy $S_\alpha^2 + C_\alpha^2 = 1$, ezért csak két (S_α, C_α) megoldás van, amelyek a $(\delta_\alpha, \gamma_\alpha) = (1,-1)$ és $(\delta_\alpha, \gamma_\alpha) = (-1,1)$ értékpárokhoz tartoznak. Mivel (S_α, C_α) meghatározza a térnegyedet a síkon, ezért mindkét (S_α, C_α) párhoz egy-egy α megoldás van, amely

$$\alpha = \arctan2(S_\alpha, C_\alpha), \tag{1.63}$$

feltéve hogy a

$$A^2 + B^2 - D^2 \geq 0 \tag{1.64}$$

munkatér feltétel teljesül.Választhatjuk azt a q_1 megoldást, amely a megelőző q_1-hez legközelebb van.

Ha már q_1-et meghatároztuk, akkor

$$S_{12} = \frac{p_y - a_1 S_1}{a_2}, \tag{1.65}$$

$$C_{12} = \frac{p_x - a_1 C_1}{a_2}, \tag{1.66}$$

és innen

$$q_1 + q_2 = \arctan2(S_{12}, C_{12}). \tag{1.67}$$

Az utolsó lépések nyilvánvalóak:

$$q_3 = p_z - d_1 - d_2, \tag{1.68}$$

$$q_1 + q_2 + q_4 = \arctan2(l_y, l_x). \tag{1.69}$$

1.2.5 A PUMA 560 robot geometriai modellje

Az RRRRRR csuklóképletű PUMA robot 6-szabadságfokú, kielégíti a dekompozíciós feltételt, mivel utolsó három csuklójának tengelyei egy közös pontban metszik egymást. A robot Denavit–Hartenberg-paramétereit az *1.2. táblázat* tartalmazza, amelyből a mértékadó koordináta-rendszerek rekonstruálhatók.

A direkt geometriai feladat megoldásakor a SCARA robotnál bemutatott utat lehet követni, azaz a Denavit–Hartenberg-paraméterek és az azokra épülő $T_{i-1,i}$ képlet figyelembevételével meghatározzuk a $T_{i-1,i}$, $i = 1,2,\ldots,6$ homogén transzformációkat, majd ezekből képezzük a $T_{0,3} = T_{0,1}T_{1,2}T_{2,3}$ és a $T_{3,6} = T_{3,4}T_{4,5}T_{5,6}$ homogén transzformációkat és ezek összeszorzásával a direkt geometriai fealadat $T_{0,6}(q) = T_{0,3}(q_1, q_2, q_3)T_{3,6}(q_4, q_5, q_6)$ megoldását. Ennek során kihasználható, hogy a K_3 keret szempontjából az utolsó három csuklóváltozó rendre $q_4 = \varphi$,

$q_5 = \vartheta$, $q_6 = \psi$ Euler-forgatásokat realizál, ezért (bár a komponens transzformációk nem azonosak az Euler-szögeknél megismertekkel, mert a Denavit–Hartenberg-konvenció szerint a csuklótengely mindig z irányú) a K_3 keretet követően az három forgatás eredő hatása az orientációra:

$$A_{3,6} = Euler(q_4, q_5, q_6) = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & C_4S_5 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 \\ -S_5C_6 & S_5S_6 & C_5 \end{bmatrix} \tag{1.70}$$

A $T_{0,3}$ homogén transzformáció az első három homogén transzformáció összeszorzása után a következő alakú lesz:

$$T_{0,3}(q_1, q_2q_3) = \begin{bmatrix} C_1C_{23} & -S & C_1S_{23} & a_2C_1C_2 + a_3C_1C_{23} - d_3S_1 \\ S_1C_{23} & C_1 & S_1S_{23} & a_2S_1C_2 + a_3S_1C_{23} + d_3C_1 \\ -S_{23} & 0 & C_{23} & -a_2S_2 - a_3S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.71}$$

Az inverz geometriai feladat dekompozíciós algoritmusa szerint egy első inverz pozícionáló, majd egy második inverz orientáló feladatot kell megoldani, ahol a második egy már kidolgozott inverz Euler-probléma. Az inverz feladat bemeneti adata a specifikált (vagy a robot gráf segítségével más specifikált adatokból számított) $T_{0,6}$ eredő homogén transzformáció, amelynek numerikusan adott elemeit jelölje

$$T_{0,6} = \begin{bmatrix} l_x & m_x & n_x & p_x \\ l_y & m_y & n_y & p_y \\ l_z & m_z & n_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.72}$$

A dekompozíciós algoritmus alapján célszerű bevezetni a következő jelöléseket:

$$p_x^* = p_x - d_6n_x, \quad p_y^* = p_y - d_6n_y, \quad p_z^* = p_z - d_6n_z \tag{1.73}$$

Ekkor az inverz pozícionáló feladathoz a következő nemlineáris egyenletrendszert kell megoldani a trigonometrikus alapösszefüggések felhasználásával:

$$\begin{aligned} a_2C_1C_2 + a_3C_1C_{23} - d_3S_1 + d_4C_1S_{23} &= p_x^* \\ a_2S_1C_2 + a_3S_1C_{23} + d_3C_1 + d_4S_1S_{23} &= p_y^* \\ -a_2S_2 - a_3S_{23} + d_4C_{23} &= p_z^* \end{aligned} \tag{1.74}$$

Az első egyenletet $(-S_1)$-gyel a másodikat pedig C_1-gyel megszorozva és a két egyenletet összeadva egy $AC_\alpha + BS_\alpha = D$ alakú egyenlethez jutunk, amelynek megoldását a SCARA robotnál már bemutattuk, ezért q_1 meghatározható:

$$-S_1p_x^* + C_1p_y^* = d_3 \rightarrow q_1 \tag{1.75}$$

Két q_1 megoldást kapunk annak megfelelően, hogy a PUMA robot az emberi kezet (kart és kézcsuklót) közelíti, és adott pozíció elérhető jobb kézzel, vagy bal kézzel. A megoldáshoz ebből egyet, pl. a megelőző konfigurációhoz legközelebbit kiválaszthatjuk.

Ezután q_1 már ismert, ezért elemi algebrai átalakításokkal képezhető

$$\begin{aligned} e &:= C_1p_x^* + S_1p_y^* = a_2C_2 + a_3C_{23} + d_4S_{23} \\ S_2 &= \frac{-p_z^* - a_3S_{23} + d_4C_{23}}{a_2} \\ C_2 &= \frac{e - a_3C_{23} - d_4S_{23}}{a_2} \end{aligned} \tag{1.76}$$

Mivel azonban S_{23}, C_{23} még ismeretlen S_2, C_2 jobb oldalán, ezért az $S_\alpha^2 + C_\alpha^2 = 1$ azonosság többszöri felhasználásával, valamint négyzetre emeléssel és összeadással próbálkozunk:

$$\begin{aligned} a_2^2C_2^2 + a_2^2S_2^2 &= e^2 + a_3^2C_{23}^2 + d_4^2S_{23}^2 - 2ea_3C_{23} - 2ed_4S_{23} + 2a_3d_4C_{23}S_{23} \\ &\quad + p_z^{*2} + a_3^2S_{23}^2 + d_4^2C_{23}^2 + 2p_z^*a_3S_{23} - 2p_z^*d_4C_{23} - 2a_3d_4S_{23}C_{23} \\ a_2^2 &= e^2 + p_z^{*2} + a_3^2 + d_4^2 - 2(ea_3 + p_z^*d_4)C_{23} - 2(ed_4 - p_z^*a_3)S_{23} \end{aligned}$$

Bevezetve az

$$A = 2(ea_3 + p_z^* d_4), \quad B = 2(ed_4 - p_z^* a_3), \quad D = e^2 + p_z^{*2} + a_3^2 + d_4^2 - a_2^2 \qquad (1.77)$$

jelöléseket, ismét egy már ismerős alakú egyenletre jutunk, melyet megoldva megkapjuk a második és harmadik csuklóváltozó összegét:

$$AC_{23} + BS_{23} = D \rightarrow q_2 + q_3 \qquad (1.78)$$

Ismét két megoldást kapunk, amely közül választhatjuk például a megelőzőhöz legközelebbi megoldást.

Ezután S_2, C_2 jobb oldala már ismert, ahonnan q_2 meghatározható:

$$(S_2, C_2) \xrightarrow{\arctan 2} q_2 \qquad (1.79)$$

Világos, hogy q_2 és $q_2 + q_3$ ismeretében képezhető q_3 is. Ezzel az inverz pozícionáló feladatot megoldottuk, a megoldások közül választhattunk a megelőző megoldás közelében vagy bizonyos konfigurációkat preferálva (jobb/bal kéz, alsó/felső könyök helyzet).

Természetesen nincs akadálya annak sem, hogy az (1.76) egyenletből előbb S_{23}, C_{23}-at fejezzük ki analitikusan megoldván egy két ismeretlenes lineáris egyenletrendszert, emeljük négyzetre S_{23}, C_{23} így kapott kifejezését, adjuk össze a két egyenletet, amikor is egy újabb $AC_2 + BS_2 = D$ alakú egyenletre jutunk alkalmas A, B, D együtthatókkal, amelynek q_2 megoldását meghatározva már számítható S_{23}, C_{23} és abból $(S_{23}, C_{23}) \xrightarrow{\arctan 2} q_2 + q_3$, majd q_2 ismeretében q_3 is.

Az egyenletrendszer megoldásánál kihasználható, hogy

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} e \\ f \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{pmatrix} e \\ f \end{pmatrix} = \frac{1}{ad - bc} \begin{pmatrix} de - bf \\ -ce + af \end{pmatrix},$$

ahonnan (1.76) alapján kapjuk, hogy

$$\begin{bmatrix} a_3 & -d_4 \\ d_4 & a_3 \end{bmatrix} \begin{pmatrix} S_{23} \\ C_{23} \end{pmatrix} = \begin{pmatrix} -a_2 S_2 - p_z^* \\ -a_2 C_2 + e \end{pmatrix},$$

amelynek megoldása:

$$S_{23} = \frac{1}{a_3^2 + d_4^2}[-a_2 a_3 S_2 - a_2 d_4 C_2 + (-a_3 p_z^* + d_4 e)]$$
$$C_{23} = \frac{1}{a_3^2 + d_4^2}[a_2 d_4 S_2 - a_2 a_3 C_2 + (d_4 p_z^* + a_3 e)] \tag{1.80}$$

$$A = 2a_2[d_4(-a_3 p_z^* + d_4 e) + a_3(d_4 p_z^* + a_3 e)]$$
$$B = 2a_2[a_3(-a_3 p_z^* + d_4 e) - d_4(d_4 p_z^* + a_3 e)] \tag{1.81}$$
$$D = a_2^2(a_3^2 + d_4^2) + (-a_3 p_z^* + d_4 e)^2 + (d_4 p_z^* + a_3 e)^2 - (a_3^2 + d_4^2)^2$$

Miután meghatároztuk az első három csuklóváltozó értékét, behelyettesíthetjük étéküket az $A_{0,3}$ orientációs mátrixba és meghatározhatjuk annak numerikus értékét:

$$A_{0,3}(q_1, q_2, q_3) = \begin{bmatrix} C_1 C_{23} & -S_1 & C_1 S_{23} \\ S_1 C_{23} & C_1 & S_1 S_{23} \\ -S_{23} & 0 & C_{23} \end{bmatrix} \tag{1.82}$$

Ezután kiszámítható $A_{3,6}$ numerikus értéke az

$$A_{0,6} = A_{0,3} A_{3,6} \Rightarrow A_{3,6} = A_{0,3}^{-1} A_{0,6} = A_{0,3}^T A_{0,6} =: \begin{bmatrix} L_x & M_x & N_x \\ L_y & M_y & N_y \\ L_z & M_z & N_z \end{bmatrix} \tag{1.83}$$

összefüggés felhasználásával.

Az eredő numerikus értékeket szándékosan $L_x, \ldots, N_z$ nagybetűk jelölik, utalva arra, hogy numerikusan eltérnek $A_{0,6}$ kis betűvel jelölt elemeitől, de ettől eltekintve az inverz orientációs feladat megoldásakor felhasználhatók az inverz Euler probléma kidolgozott egyenletei (a kisbetűs elemek helyett a nagybetűvel jelölteket használva, pl. $\arctan 2(N_y, N_x) = \varphi = q_4$ stb.):

$$A_{3,6} \xrightarrow{\textit{inverz Euler}} \varphi = q_4, \vartheta = q_5, \psi = q_6 \tag{1.84}$$

2. A PÁLYATERVEZÉS ALAPJAI

Az irányítások számára meg kell fogalmazni, mi legyen a szabályozott jellemző előírt értéke (alapjel), vagy követendő időfüggvénye (követő jel), vagyis az ún. referencia jel. A szabályozó ehhez képest képzi a hibajelet (a referencia jel és a szabályozott jellemző aktuális értéke közötti eltérést), és a szabályozási algoritmus célja a hibajel hatásának mérséklése a tranziensek alatt és állandósult vagy kvázi-stacioner állapotban a hiba lehetőség szerint teljes megszüntetése. További cél még a zavaró jelek hatásának csökkentése és a szabályozási rendszer robusztussá tétele a szabályozott szakasz (folyamat stb.) paramétereinek változása ellenére.

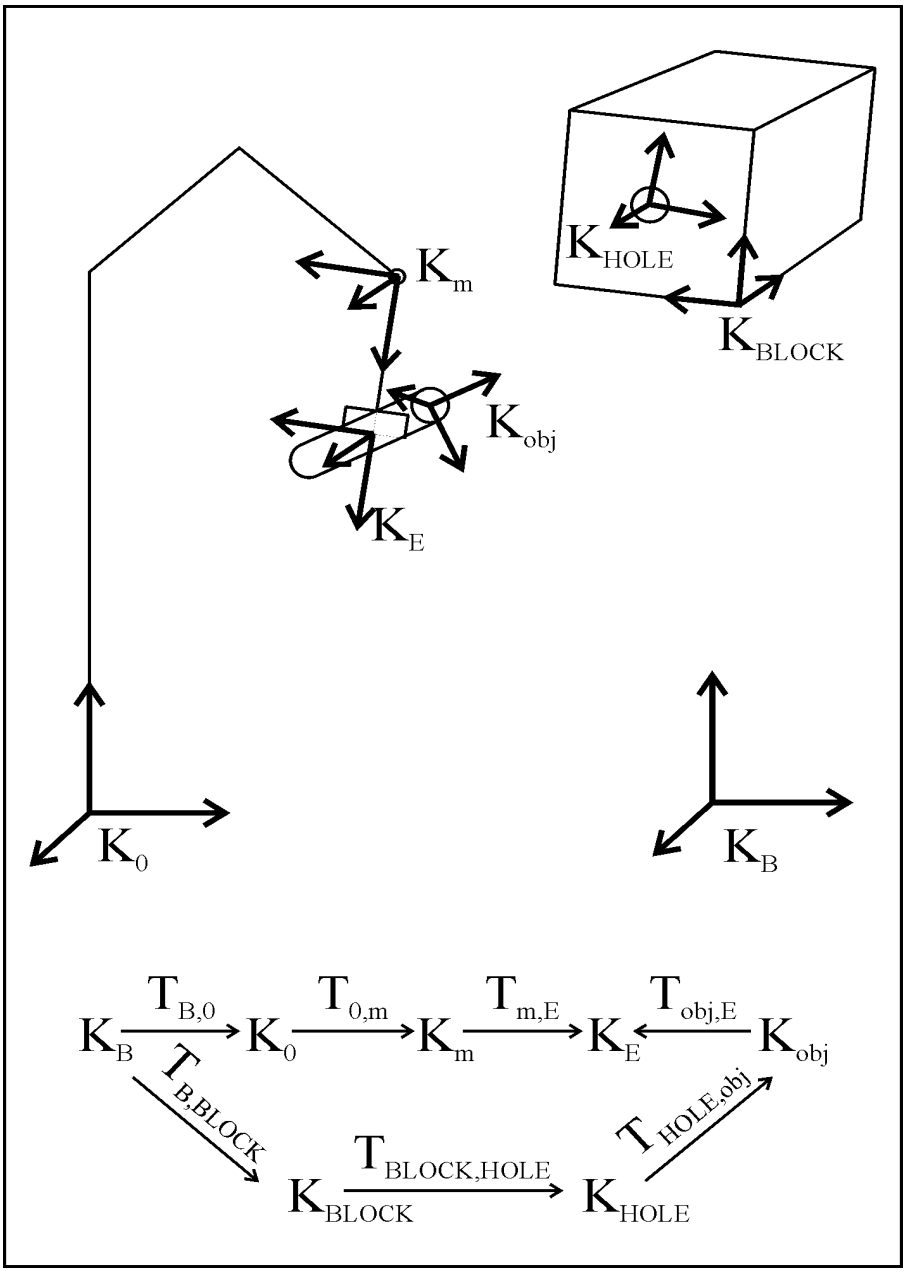
Mozgó rendszerek (robotok, járművek stb.) esetén a referencia jelnek a rendszer mozgása számára előírt pálya felel meg, amely időben változó, megvalósítására hatékony irányítási algoritmusok szükségesek. A pályatervezés tehát a robotirányítások számára elengedhetetlen, ezért néhány alapvető eljárást mutatunk be az ipari robotok körében elterjedt pályatervezési módszerek közül.

2.1 A pályatervezés adtstruktúrája

Tekintsük először a *2.1. ábrában* bemutatott esetet, amikor a robot megfogójával korábban már megfogott egy tárgyat, és a feladat a hengerszerű tárgy elhelyezése egy szerelvény blokk furatába. A problémát csap-furat problémának is szokás nevezni (peg-into-hole problem).

A célunk annak bemutatása, hogy az alkalmazói oldalon megfogalmazott feladat egy általános $T_{0,m} = COORD * POS * TOOL^{-1}$ adatstruktúrává konvertálható. Ennek jobb oldalát kiszámítva megkapjuk az m-szabadságfokú robot erdő homogén transzformációját, majd erre megoldva az inverz geometriai feladatot a kapott q vektor tekinthető a referencia jel aktuális értékének a csuklóváltozó vektor értékében kifejezve. Ennek komponensei elküdhetők egy csukló-filozófiájú decentralizált irányító rendszer alrendszereinek alapjel bemeneteire, és ha az alrendszerek szabályozásai ezt megvalósítják, akkor a robot a térben az előírt helyzetben (pozícióban és orientációban, röviden helyzetben, POSE) lesz.

Tekintsük tehát a *2.1. ábrában* specifikált feladat koordináta-rendszereit (röviden kereteit, frames). A feladatban K_B a bázis keret (world frame), K_0 a robot referencia kerete, elhelyezkedése a bázishoz képest az ismert $T_{B,0}$ homogén transzformációval adható meg. K_m a robot utolsó szegmensének kerete, az utolsó szegmenshez van rögzítve az end effector (megfogó), amelynek kerete K_E, és a végberendezés a mozgás egy korábbi fázisában már megfogott tárgyat tart, amelynek kerete K_{obj}. A korábbi mozgási fázis specifikációiból kinyerhető $T_{obj,E}$, amely megmondja numerikusan a végberendezés elhelyezkedését a tárgyhoz képest.



2.1. ábra. A csap-furat probléma. Koordináta-rendszerek és robot gráf

Az aktuális mozgási fázisban elő van írva, hogy helyezkedjel el a tárgy a szerelvény furatához képest, amelyet a $T_{HOLE,obj}$ homogén transzformációban adhat meg numerikusan a felhasználó. A bázis koordináta-rendszerhez képest a szerelvény például egy munka asztalon helyezkedhet el, a hozzá rögzített keret K_{BLOCK}, helyzetét numerikusan a $T_{B,BLOCK}$ ismert homogén transzformáció reprezentálja. A szerelvényhez képest a furat meghatározott pozícióban és orientációban van, a furathoz rögzített keret legyen K_{HOLE}, elhelyezkedése a szerelvénynek a gépészmérnöki gyakorlatban használt CAD-rendszerben készült szerelési rajzából kinyerhető és a $T_{BLOCK,HOLE}$ homogén transzformációvá konvertálható.

Jól látható, hogy a $T_{HOLE,obj}$ felhasználói információ helyett a robotirányítás pályatervezési feladatához a $T_{0,m}$ meghatározása szükséges, amely a robot gráfból a korábban megismert interpretációval és $T_{obj,E} * T_{m,E}^{-1} = (T_{m,E} * T_{obj,E}^{-1})^{-1}$ figyelembevételével a következő lesz:

$$T_{0,m} = \underbrace{T_{B,0}^{-1} * T_{B,BLOCK} * T_{BLOCK,HOLE}}_{COORD} * \underbrace{T_{HOLE,obj}}_{POS} * \underbrace{(T_{m,E} * T_{obj,E}^{-1})^{-1}}_{TOOL^{-1}} \tag{2.1}$$

Az itt vizsgált csap-furat probléma és számos más pályatervezési feladat a

$$T_{0,m} = COORD * POS * TOOL^{-1} \tag{2.2}$$

sémára vezet, ahol az adatstruktúrában szereplő homogén transzformációk közül $COORD$ számos koordináta-rendszertől függ, POS a felhasználói előírás és $TOOL$ a fizikai vagy logikai szerszám. Esetünkben $TOOL = T_{m,E} * T_{obj,E}^{-1} = T_{m,E} * T_{E,obj}$ a logikai szerszám, ahol a fizikai $T_{m,E}$ szerszámhoz hozzá lett véve a $T_{E,obj}$ tárgy, és a kettő együttesen alkotja a logikai szerszámot. Megjegyezzük, hogy magas szintű robotprogramozási nyelvekben vannak utasítások a fizikai szerszám lecserélésére logikai szerszámmá.

Általában a robot mozgása fázisokra bontható, például a csap-furat probléma felbontható a furat megközelítésére úgy, hogy a csap tengelye és a furat tengelye egybeessen, de a relatív pozíció ne legyen nulla, ezt követheti a mozgás a furat kezdetéhez, majd innen a furat tengelye mentén a mozgás a furat aljáig. Az első két fázis mozgásirányítással megvalósítható (szabad mozgás), a harmadik azonban a csap és a furat tengelye között mindig meglévő kisebb irányítási pontatlanságok és a szoros illesztések miatt általában csak ún. hibrid pozíció/erő irányítással valósítható meg (korlátozott mozgás). Az utóbbi során tengelyirányú erőt is ki kell fejteni a szabályozás keretében a mozgás mellett, és az erő/nyomaték érzékelő által mért hirtelen

megnövő erőből lehet következtetni arra, hogy elérte a csap a furat alját, és a csap betolását a furatba le kell állítani. A hibrid irányítás a csuklóváltozókat mérő belső érzékelőkön kívül (6-komponensű) erő/nyomaték érzékelő alkalmazását is igényli, ami jelentős költségnövelő tényező.

A továbbiakban megfogalmazunk egy pályatervezési sémát egyetlen skalárváltozóban, majd kihasználván, hogy a csuklóvektor és a térben ún. Descartes-koordinátákban a pozíció és az orientáció is skalár komponensekkel jellemezhető, a csuklóváltozókban és a Descartes-koordinátákban történő pályatervezést a séma többszöri alkalmazásával oldjuk meg.

2.2 Pályatervezés skalárváltozóban folytonos gyorsulással

A következőkben egy sémát (primitívet) fogunk kidolgozni egyetlen skalárváltozó (y) esetén, amely során feltesszük, hogy adottak a skalárváltozó által befutandó sarokpontok és az elérési ún. abszolút időértékek (később relatív időre térünk át), továbbá a gyorsulás abszolút értéke korlátozva van egy megengedett maximális érték révén:

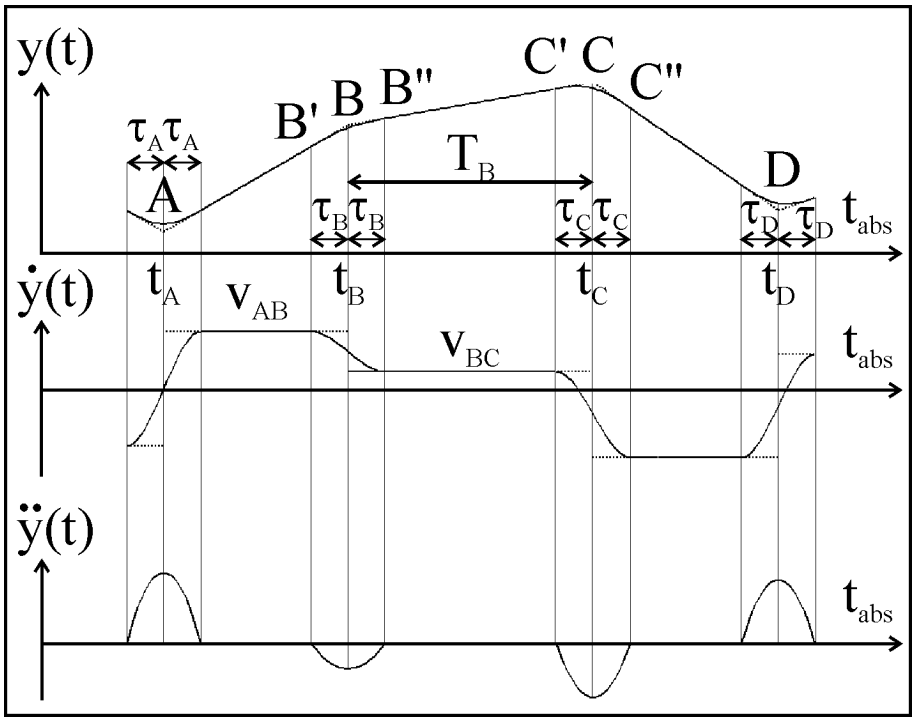
$$\{y_i\} = \{\ldots, A, B, C, D, \ldots\}, \qquad \left|\ddot{y}\right|_{\max} \text{ adott} \tag{2.3a}$$

$$\{t_i\} = \{\ldots, t_A, t_B, t_C, t_D, \ldots\} \text{ abszolút idő} \tag{2.3b}$$

Az induló hipotézis a sarokpontok összekötése lineáris szakaszokkal, azaz szakaszonként lineáris $y(t)$ függvény megcélzása. Ekkor az idő szerinti első derivált $\dot{y}(t)$ szakaszonként konstans lépcsős függvény lenne, amelynek deriváltja, azaz $y(t)$ -nek a második deriváltja azonosan nulla, kivéve az első derivált ugrás helyeit, ahol Dirac-deltákat tartalmazna, tehát nem lenne korlátos, szemben az elvárásokkal. Ennek felszámolására két lehetőség terjedt el a robotikában.

Az első szakaszonként lineáris első deriváltat céloz meg , amikor is a második derivált lépcsős függvény (trapéz alakú sebességprofil). A mozgástörvények alapján ekkor a gyorsulás lépcsős függvényének ugráshelyein jelentős erő/nyomaték változás szükséges, ami komoly feladatot ró a szabályozások számára (nagy hibajel és nagy erősítés esetén telítés a hajtásokban, ami az erősítés visszavételével kerülhető el, ami viszont nagyobb tranziens és maradó hibával jár).

A második folytonos gyorsulást céloz meg. Mivel azonban az ún. utazó szakaszon a sebesség konstans és a gyorsulás nulla, ezért a szakaszok határán a sebességváltozás megvalósításához a gyorsításnak nullából nem nullába kell mennie, tartani kell az előjelet (a gyorsításnak vagy lassításnak megfelelően), majd ismét nullába kell mennie. Ez a gyorsítás számára folytonos függvénnyel a legegyszerűbben másodfokú polinommal (parabolával) valósítható meg.



2.2. ábra. A pályatervezés sémája folytonos gyorsulással egy skaláváltozóban

A pályatervezés tehát egy $B' \to B \to B''$ gyorsító szakaszra és egy $B'' \to C$ utazó szakaszra bontható. A könnyebb kezelés érdekében t relatív időt vezetünk be, amelynek nulla helye a gyorsító szakasz közepe, a gyorsító szakasz teljes hossza pedig a τ_B érték kétszerese:

$$B' \to B \to C: \quad t = t_{abs} - t_B \in [-\tau_B, T_B] \text{ relatív idő}$$

$$\begin{aligned} B'B'': \quad \ddot{y}(t) &= a_0 t^2 + a_1 t + a_2 \\ \dot{y}(t) &= a_0 \frac{t^3}{3} + a_1 \frac{t^2}{2} + a_2 t + a_3 \\ y(t) &= a_0 \frac{t^4}{12} + a_1 \frac{t^3}{6} + a_2 \frac{t^2}{2} + a_3 t + a_4 \end{aligned} \tag{2.4}$$

Mivel az $y(t)$ negyedfokú polinomnak 5 együtthatója van, ezért meghatározásukhoz 5 feltétel szükséges, amelyek a következők:

$$\begin{aligned} \ddot{y}(-\tau_B) &= 0 \\ \ddot{y}(\tau_B) &= 0 \\ \dot{y}(-\tau_B) &= v_{AB} = \frac{B - B'}{\tau_B} \\ \dot{y}(\tau_B) &= v_{BC} = \frac{C - B}{T_B} \\ y(\tau_B) &= B + v_{BC} \cdot \tau_B \end{aligned} \tag{2.5}$$

Behelyettesítve a függvényekbe ezeket az értékeket egy lineáris egyenletrendszer keletkezik az ismeretlen együtthatókra, amelynek analitikus megoldása a következő:

$$\begin{aligned} a_0 &= -\frac{3}{4} \cdot \frac{v_{BC} - v_{AB}}{\tau_B^3} \\ a_1 &= 0 \\ a_2 &= \frac{3}{4} \cdot \frac{v_{BC} - v_{AB}}{\tau_B} \\ a_3 &= \frac{v_{BC} + v_{AB}}{2} \\ a_4 &= B + \frac{3}{16}(v_{BC} - v_{AB})\tau_B \end{aligned} \tag{2.6}$$

A paraméterek ismeretében egy skalárváltozóban a pálya a gyorsító szakaszon egy negyedfokú, az utazó szakaszon pedig egy elsőfokú polinom lesz:

$$y(t) = \begin{cases} a_0 \dfrac{t^4}{12} + a_1 \dfrac{t^3}{6} + a_2 \dfrac{t^2}{2} + a_3 t + a_4, & t \in [-\tau_B, \tau_B) \\ B + v_{BC} \cdot t, & t \in [\tau_B, T_B - \tau_C) \end{cases} \tag{2.7}$$

Az $y(t)$ függvényt $t \in [-\tau_B, T_B - \tau_C)$ között használhatjuk, $T_B - \tau_C$ elérésekor újra el kell végezni a pályatervezést az új $C' \to C \to D$ szakaszra, arra alkalmazván a $B' \to B \to C$ szakaszra kidolgozott pályatervezési primitívet.

Vegyük észre, hogy a pálya nem pontosan a t_B abszolút időpontban halad át az előírt B ponton, az eltérés ekkor

$$B - y(0) = B - a_4 = -\frac{3}{16}(v_{BC} - v_{AB})\tau_B \tag{2.8}$$

A hiba abszolút értékét τ_B csökkentésével lehetne csökkenteni, ennek azonban az előírások szerint határt szab, hogy

$$\left|\ddot{y}\right|_{\max} \geq \left|\ddot{y}(0)\right| = \left|a_2\right| = \frac{3}{4} \cdot \frac{\left|v_{BC} - v_{AB}\right|}{\tau_B} \Rightarrow \tau_B \geq \frac{3}{4} \cdot \frac{\left|v_{BC} - v_{AB}\right|}{\left|\ddot{y}\right|_{\max}} \tag{2.9}$$

Az előírt sarokpontokon való áthaladást a pályatervezési primitív megtartása mellett egyszerű eszközökkel lehet biztosítani, ehhez elegendő ugyanis a kritikusnak tekintett sarokpontot megduplázni. Ennek ára, hogy a pálya befutási ideje $2\tau_B$ értékkel megnő, ha B a kritikus sarokpont, továbbá a sebesség egy pillanatra nullává válik. Világos, hogy a start és end sarokpontot, mivel azt garantáltan be kell tartani, a sorozatban mindig meg kell duplázni: $\{S, S, \ldots, A, B, C, D, \ldots, E, E\}$. Egy speciális feltétel olvasható még le a *2.1. ábrából*, nevezetesen teljesülni kell a $T_B \geq \tau_B + \tau_C$ feltételnek is.

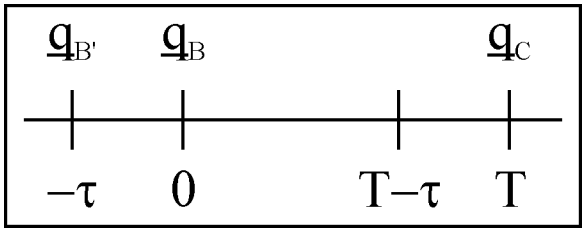
2.3 Pályatervezés csuklókoordinátákban

A csuklóvektor komponensei skalárok, ezért azokra komponensenként alkalmazható az egyetlen skalárváltozóra kidolgozott primitív. Ennek során azonban be kell tartani, hogy a csuklóhajtások motorjainak eltérő teljesítményviszonyai miatt az egyes csuklóváltozók deriváltjaira eltérő $\left|\dot{q}_i\right|_{\max}$, $\left|\ddot{q}_i\right|_{\max}$ maximális értékek vannak előírva. Ezen kívül az egyes csuklók $\left|q_{Ci} - q_{Bi}\right|$ megváltozásának értéke is eltérő lehet, ezért a $\left|\dot{q}_i\right|_{\max}$ maximális sebességgel való befutásukhoz csuklónként eltérő idő szükséges. Másrészt viszont csak azt a csuklót célszerű maximális sebességgel mozgatni, amelyre a saját befutási idő maximális. Nincs értelme ugyanis más csuklókat hamarabb a célhelyzetbe küldeni, mert hiába érkezik egy csukló korán a célhelyzetébe, ha a másik csukló még nincs a célhelyzetében, mivel a teljes robot még nem érte el célkonfigurációt, és nem lehet például egy tárgy megfogását stb. indítani.

Mivel a fölöslegesen gyorsan mozgó csuklók mozgásához fölöslegesen nagyobb kinetikus energia szükséges, ezért célszerű ún. koordinált mozgást megvalósítani, amikor is csak a legnagyobb időt igénylő csukló mozog maximális sebességgel, a

többiek pedig ugyanennyi idő alatt, de a saját maximális sebességüknél lassabban és kevesebb energiát fogyasztva mozognak (koordinált mozgás elve).

A pályatervezés sémáját csuklóváltozókban a *2.3. ábra* mutatja.



2.3. ábra. A pályatervezés sémája csuklóváltozókban

Az iterációk valós időben való elkerülésének érdekében minden τ értéket (2.9) alapján egységesen egyformának választottunk figyelembe véve, hogy a legnagyobb $\left|v_{BC,i} - v_{AB,i}\right|$ sebességváltozás értéke a $\left|\dot{q}_i\right|_{\max}$ sebességmaximum kétszerese is lehet:

$$\tau = \max_i \frac{3}{2} \frac{\left|\dot{q}_i\right|_{\max}}{\left|\ddot{q}_i\right|_{\max}} \tag{2.10}$$

Pályatervezési algoritmus csuklóváltozókban:

$q_{B'} := q(T - \tau)$

$q_B := q_C$

$q_C := solve\{COORD * POS * TOOL^{-1}\}_D$

$T_i := \dfrac{\left|q_{Ci} - q_{Bi}\right|}{\left|\dot{q}_i\right|_{\max}}, \quad \forall i$

$T := \max\{\max_i\{T_i\}, 2\tau\}$

$v_{AB} := \dfrac{q_B - q_{B'}}{\tau}$

$v_{BC} := \dfrac{q_C - q_B}{T}$

$a_0, \ldots, a_4, \quad t := -\tau$

repeat until $t < T - \tau$: $\{q(t); \quad t := t + \Delta\}$

A számítások itt vektorizálva értendők (lásd MATLAB), ciklusba szervezésük C-nyelvi környezetben triviális. A $q(t)$ függvény a gyorsító szakaszon negyedfokú, az utazó sebességű szakaszon elsőfokú polinom, Δ a mintavételi idő.

Célszerű kihangsúlyozni, hogy a sarokpontokat vagy közvetlenül csuklóváltozókban kell betanítani, vagy pedig az adatstruktúrában a $COORD, POS, TOOL$ homogén transzformációkkal indirekt definiálni. Az utóbbi esetben *solve* utal arra, hogy meg kell oldani az inverz geometriai feladatot. A sarokpontokat a pályatervezás sima függvényekkel köti össze.

A pályatervezés az utazó sebességű szakaszon a $T_{0,m}(q)$ argumentumába lineáris függvényt helyettesít. Mivel azonban a rotációs csuklók miatt $T_{0,m}(q)$ nemlineáris, ezért egy nemlineáris függvény argumentumába helyettesítve lineáris időfüggvényt a kiadódó robotmozgás nem lesz lineáris az időben, azaz a térben két sarokpont között a robot nem fog egyenes mentén mozogni. Az "imbolygó" mozgás ugyan csökkenthető lenne azáltal, ha sok egymáshoz közeli sarokpontot tanítanánk be (kis változásra a folytonos függvény jól közelíthető lineárissal), ez azonban a betanítást nagyon időigényessé tenné.

Ki kell tehát hangsúlyozni, hogy a pályatervezés csuklóváltozóban egyszerű, mert csak sarokpontonként egyszer igényli az inverz geometriai feladat megoldását, másrészt viszont csak akkor javasolható alkalmazása, ha a robot a sarokpontok között haladva távol marad az akadályoktól. Minden más esetben egyenes mentén haladó és az orientációt is simán változtató pályát kell tervezni az akadályok elkerülése érdekében. Erre a pályatervezés Descartes-koordinátákban ad lehetőséget.

A robotprogramozási nyelvek a választásra lehetőséget adnak, lásd később a GO, GONEAR utasításokat csuklóváltozóban, valamint a GOS, GOSNEAR utasításokat Descartes-koordinátákban történő pályatervezés esetén (S= stright line az egyenesre utal).

2.4 Pályatervezés Descartes-koordinátákban

A robot mindig térben mozog, ezért a csuklófilozófiájú irányítást a közvetlenül térbeli, tehát pozícióban és orientációban történő irányítástól a robotika területén azáltal különböztetjük meg, hogy az utóbbit Descartes-koordinátákban történő irányításnak nevezzük (Descartes latin neve után angolul Cartesian Control). Ilyen értelemben beszélünk pályaterezésről is közvetlenül Descartes-koordinátákban.

Amennyiben a pozíciót x, y, z jelöli, az orientációt pedig a φ, ϑ, ψ Euler-szögekkel jellemezzük, akkor a pályatervezési feladat felfogható egy fiktív TTTRRR csuklóképletű robot számára pályatervezésnek, amelyben a fiktív "q" csuklóvektor a pályatervezés idejére $q = (x, y, z, \varphi, \vartheta, \psi)^T$. Ennek ismét skalárváltozók a komponensei, ezért a csuklóváltozók terében korábban kidolgozott algoritmust alkalmazhatjuk a fiktív robotra. Arra kell azonban ügyelni, hogy ebben

az esetben az irányításban történő alkalmazáshoz minden mintavételi időpontban meg kell oldani az inverz geometriai feladatot, amely jelentős mértékben terheli az irányítást végző processzort vagy számítógépet.

Pályatervezési algoritmus Descartes-koordinátákban:

$$T_{0,m} = COORD * POS * TOOL^{-1}$$

$$\begin{matrix} B' & \rightarrow & B & \rightarrow & C \\ T_{0,m}^{B'} & & T_{0,m}^{B} & & T_{0,m}^{C} \\ \begin{pmatrix} p_x \\ p_y \\ p_z \\ \varphi \\ \vartheta \\ \psi \end{pmatrix}_{B'} & & \begin{pmatrix} p_x \\ p_y \\ p_z \\ \varphi \\ \vartheta \\ \psi \end{pmatrix}_{B} & & \begin{pmatrix} p_x \\ p_y \\ p_z \\ \varphi \\ \vartheta \\ \psi \end{pmatrix}_{C} \end{matrix}$$

$$TTTRRR \xrightarrow{q=(x,y,z,\varphi,\vartheta,\psi)^T} a_0,\ldots,a_4 \text{ for } p_x(t),\ldots,\psi(t), \quad t := -\tau$$

repeat until $t < T - \tau$:

{

$$p(t) = \begin{pmatrix} p_x(t) \\ p_y(t) \\ p_z(t) \end{pmatrix}, \quad A(t) = Euler(\varphi(t), \vartheta(t), \psi(t))$$

$$T_{0,m}(t) = \begin{bmatrix} A(t) & p(t) \\ 0^T & 1 \end{bmatrix} \xrightarrow{solve} q(t)$$

$$t := t + \Delta$$

}

Az algoritmusban *solve* az inverz geometriai feladat megoldását jelenti a valódi robot számára. Jól látható, hogy az inverz geometriai feladatot a mintavételi idő (Δ) ütemében kell megoldani. Ha erre a rendelkezésre álló idő az adott processzoron (processzorokon) valós időben kevés, akkor próbálkozhatunk az inverz feladat ritkább megoldásával, és a két $q(t_1), q(t_2)$ csuklóváltozó érték között lineáris interpolációval az idő függvényében a mintavételi idő ütemében. Robotok esetében a szabályozás ideális mintavételi ideje kb. 1 msec.

3. PUMA robot irányító és programozási rendszere

A PUMA 560 robot egy RRRRRR csuklóképletű robot, amelynek utolsó három csuklótengelye egy közös pontban metszi egymást, és az utolsó 3 csukló a harmadik szegmens koordináta-rendszeréhez képes z, y, z forgatásokat valósít meg. Ezért q_4, q_5, q_6 rendre felfogható a φ, ϑ, ψ Euler-szögeknek, amit az inverz geometriai feladat megoldásakor korábban már ki is használtunk.

Másrészt a Nokia Puma 560 robot ARPS robotprogramozási nyelve a BASE és TOOL koordinátarendszerek közötti $T_{BASE,TOOL}$ (korábbi jelöléssel $T_{B,E}$) homogén transzformáció orientációjának jellemzésére nyelvi szinten szintén Euler-szögeket használ, amelyet azonban erre a célra *o,a,t*-vel jelöl. A kettő nem keverendő össze.

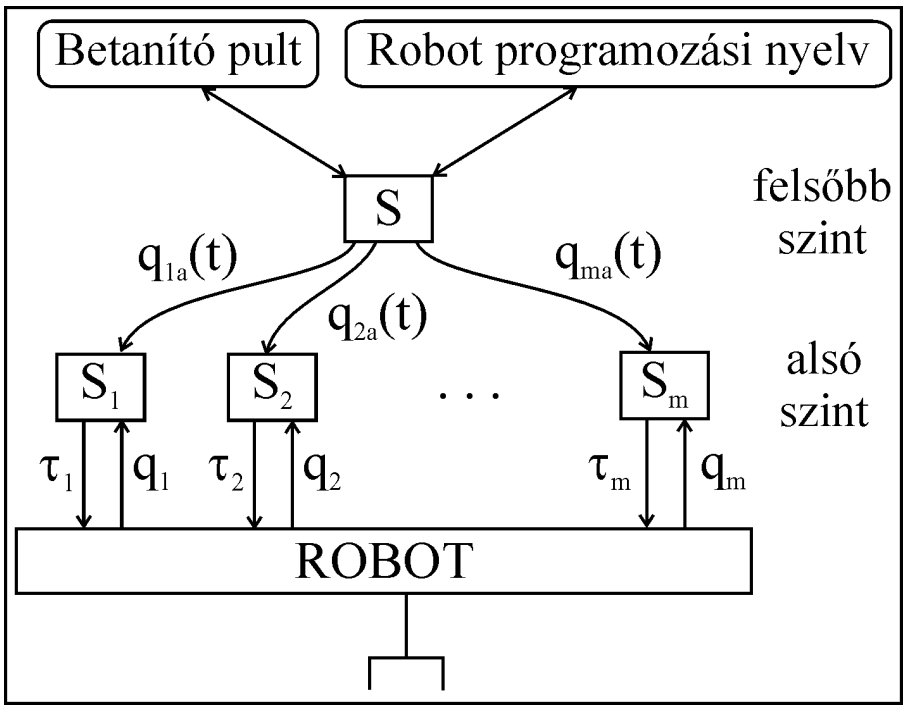
3.1 Robotirányítási alapfogalmak

Az ipari robotok irányító rendszere rendszerint egy többszintű irányító rendszer, amelynek felsőbb szintjén történik a robotprogramozási nyelv utasításainak értelmezése, a pályatervezés, továbbá az utasítások végrehajtásának megszervezése és a felügyelete, alsóbb szinten pedig a robotirányítási algoritmusok megvalósítása.

Az algoritmusok és az irányítási filozófia különböző lehet, kezdve az egyszerű decentralizált kaszkád irányításoktól (háromhurkos pozíció, sebesség és áramszabályozás) a robot nemlineáris dinamikus modelljére épülő különféle korszerű irányítási algoritmusokig (kiszámított nyomatékok módszere, csúszó szabályozás, nemlineáris szétcsatolás, hibrid pozíció/erő irányítás, önhangoló adaptív irányítás).

A *3.1. ábrán* egy egyszerű csuklóváltozó filozófiájú irányítás sémáját mutatjuk be. Ipari robotok esetén a pálya mértékadó sarokponjait rendszerint csak betanítás révén, a betanító személy vizuális felügyelete mellett lehet bevinni a rendszerbe, amire a betanító pult szolgál. Az így bevitt adatokon aztán már egyszerű adatmódosításokat (másolás, eltolás stb.) nyelvi eszközökkel is el lehet végezni. A felhasználó programját az editor szolgátatásaival viheti be a rendszerbe, és alkothatja meg az alkalmazás robotprogramját.

A betanító pult (Teach Pendant) és a robotprogramozási nyelv (Robot Programming Language) az irányító rendszer felső szintjén megjelenő szolgáltatás, amely vagy egy master célprocesszoron, vagy pedig egy host számítógépen (PC stb.) valósul meg.



3.1. ábra. Robotirányítási rendszer elvi sémája

A mozgást előíró utasítások végrehajtásakor a felső szinten megtörténik a pályatervezés, és a kiszámított pálya adatai eljutnak a decentralizált tengelyhajtások alapjel bemeneteire, amelyet azok értelmeznek és az alkalmazott algoritmustól és paraméterbeállítástól függően kisebb-nagyobb hibával megvalósítanak. A robot fizikai értelemben vett pontossága erősen függ a szabályozás pontosságától.

A szabályozási sémában S a master processzor, S_i, $i = 1, \ldots, m$ pedig a csuklóhajtások processzorai. A pályatervezés eredményeképpen előálló q_{ia} alapjel a master processzortól valamilyen (párhuzamos stb.) interfészen jut el a csuklóprocesszorokig.

Az S_i alrendszer méri a q_i csukóváltozó értékét, amely itt a szabályozott jellemző szerepét tölti be, képezi a szabályozás $q_{ir} = q_{ia} - q_i$ hibajelét, a választott szabályozási algoritmus és annak paraméterbeállítása alapján kiszámítja a beavatkozó jelet, amely tekinthető a beavatkozó szervként működő motor meghajtó nyomatékának (a jelölésben itt τ a nyomaték angol nevének, torque, rövidítése). A motor áttételt is tartalmaz, amelynek feladata a kisebb motoroldali nyomaték megnövelése a szegmens oldalon az áttétel arányában.

A nyomaték rendszerint áramszabályozás révén valósul meg, mivel DC motorok esetén a motor tengelyén kifejtett nyomaték egyenesen arányos a rotorárammal. A hajtás teljesítmény elektronikája rendszerint a nagyobb teljesítmény és a melegedés hatásának csökkentése érdekében impulzus szélesség modulációt (PWM) használ.

A csuklóváltozót egy nagyfelbontású kódtárcsa impulzusainak (inkremensek) számlálásával lehet mérni, azonban mivel ez csak növekménymérésre alkalmas, a kezdeti értéket a bekapcsolás után egy kalibrálási folyamat keretében kell meghatározni, amiben egy nagyfelbontású helipot analóg jele, valamint a kódtárcsa index jele és fázisban eltolt két impulzus jele is szerepet játszik. A kódtárcsa fázisban eltolt két impulzusjele lehetővé teszi a forgásirány figyelembevételét. A növekménymérésre manapság célprocesszorok is rendelkezésre állnak, amelyek egyszerre több tengely impulzusszámlálását is el tudják végezni.

A csuklóváltozókat (és deriváltjaikat) mérő érzékelőket belső érzékelőknek (internal sensors) nevezik, szemben az intelligens külső érzékelőkkel (external sensors), amelyek hatkomponensű erő/nyomaték mérésre, közvetlen pozíció/orientáció mérésre vagy taktilis érzékelésre képesek.

Pont-pont irányításról (PTP, point-to-point control) beszélünk, ha a hajtások tengelyei számára nincs definiálva a teljes $q_{ia}(t)$ pálya az idő függvényében, hanem csak a pálya q_{ia} végpontja. Ilyenkor a robot a szabályozási tranziensektől függően kiadódó $q_i(t)$ trajektória mentén közeledik a végponthoz, és mivel ez a szabályzás belső tulajdonsága, ezért jelentős az ütközésveszély a környezettel.

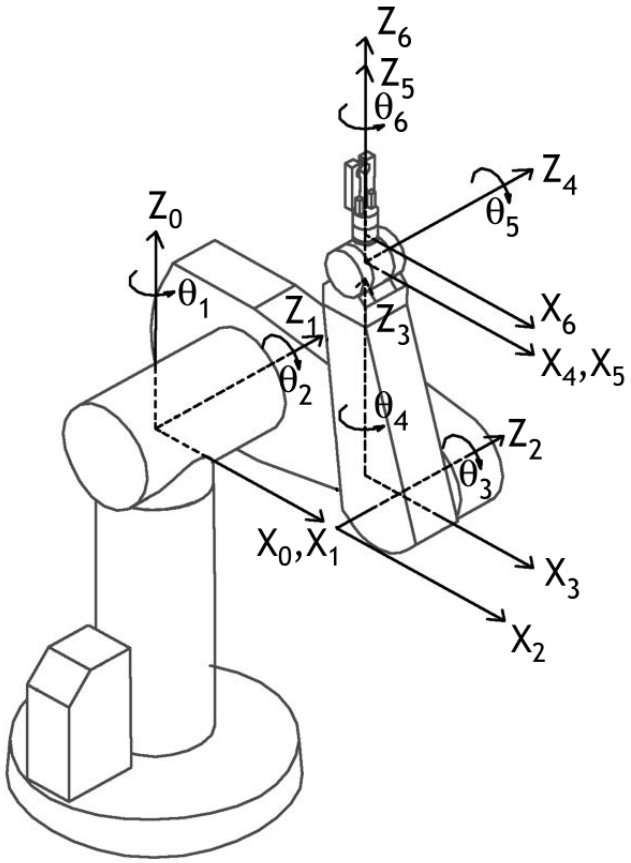
Folytonos pályairányítás (CT, continuous path control) esetén a hajtások tengelyei a befutandó pontok között megtervezett $q_{ia}(t)$ pályát kapnak az idő függvényében. A pályatervezés történhet csuklóváltozókban vagy közvetlenül Descartes-koordinátákban, esetleg interpoláció bevonásával. A pálya mentén az időparamétert el kell osztani a maximális sebességek és gyorsulások figyelembevételével. A robot megtervezett és a robotprogram tesztelésekor ellenőrzött pályán mozog, ezért kicsi az ütközésveszély a környezettel, különösen ha a pályatervezés Descartes-koordinátákban történt, amikor a robot megfogója a kitüntetett pontokat egyenessel köti össze. A pályatervezés koordinált mozgást valósít meg, ezét a csuklók egyszerre érik el a kitüntetett végpontokat és az energiafelhasználás gazdaságos.

3.2 A Nokia PUMA 560 robot HW/SW felépítése

A 6-szabadságfokú RRRRRR csuklóképletű Puma 560 robot Denavit–Hartenberg-paramétereit a *3.1. táblázat* tartalmazza. A robot vázlatos felépítését és a mértékadó koordináta-rendszereket a *3.2. ábra* mutatja be.

3.1. táblázat. A Nokia PUMA560 robot Denavit–Hartenberg-paraméterei

i	q_i	ϑ_i	d_i	a_i	α_i
1	ϑ_1	ϑ_1	0	0	–90°
2	ϑ_2	ϑ_2	0	a_2	0°
3	ϑ_3	ϑ_3	d_3	a_3	90°
4	ϑ_4	ϑ_4	d_4	0	–90°
5	ϑ_5	ϑ_5	0	0	90°
6	ϑ_6	ϑ_6	d_6	0	0°



3.2. ábra. A Nokia Puma 560 robot felépítése és koordináta-rendszerei

A koordináta-rendszerek a felhasználó szempontjából mint

$$K_B \xrightarrow{BASE} K_0 \xrightarrow{T_{06}} K_6 \xrightarrow{TOOL} K_E \tag{3.1}$$

jelennek meg, ahol BASE és TOOL a robot ARPS programozási nyelvének ilyen nevű utasításaival állíthatók be a kívánt értékre. Itt BASE a felhasználó által preferált világ koordináta-rendszert definiálja, TOOL pedig a fizikai megfogó elhelyezkedését K_6-hoz képest. A megfogó cserélhető, a cseréhez K_6-on a megfelelő furatok ki vannak képezve.

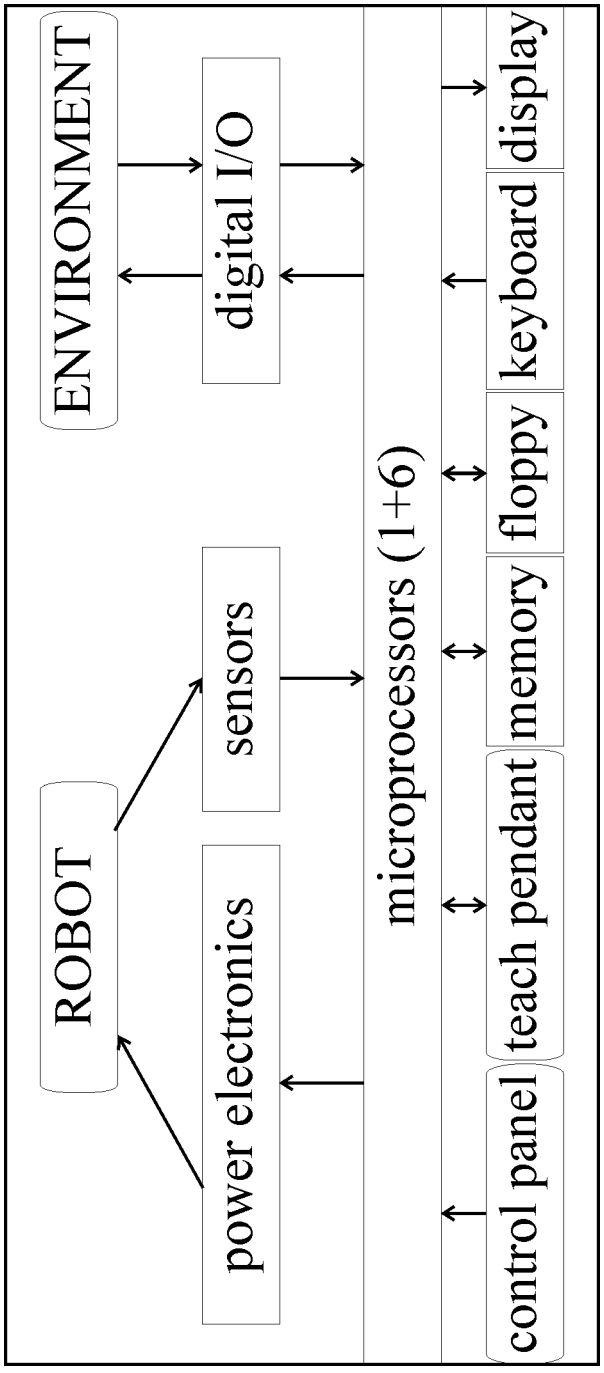
A robot fontosabb hardver egységeit a *3.3. ábra* mutatja be.

A robotirányítási feladatok ellátását 1 master és 6 slave processzor felügyeli és valósítja meg. A master processzoron fut az ARPS robotprogramozási nyelv, a pályatervezés és a szabályozások alapjel időfüggvényeinek előállítása, továbbá a master processzor felügyeli a betanítást, a bemeneti/kimeneti adatforgalomat, a rendszer státuszának figyelését és az ember/gép kapcsolatot.

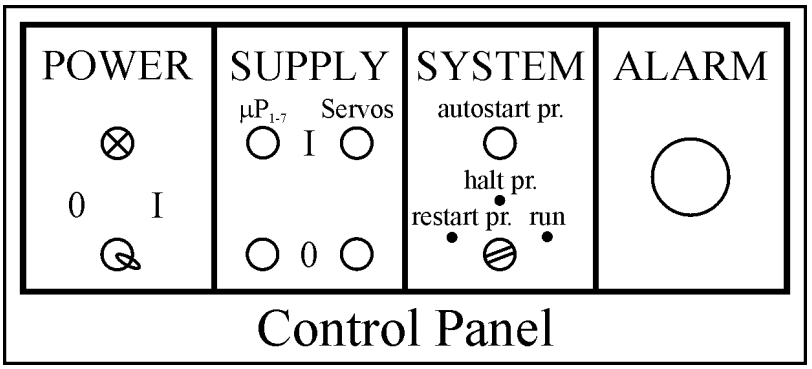
A slave processzorok valósítják meg a decentralizált szervohajtásokat, tehát tengelyenként egy-egy slave processzor végzi az illető csuklóváltozó alapjelének fogadását, a csuklóváltozó belső érzékelőjének lekérdezését, a hibajel képzését, a tengelyhajtás szabályozási algoritmusának és a szabályozó kimenőjelének képzést, a kimenőjel eljuttatását a teljesítmény elektronikához, a tengelyhajtás státuszfigyelését, valamint a szenzor és státusz információ eljuttatását a master processzorhoz.

A vezérlő pult a tápellátást, a programfutást és a vészleállást vezérli. A betanító pult RS 232 soros interfészen keresztül kapcsolódik az irányító rendszerhez.

A digitális I/O vonalak 16 vonalas egységekbe vannak szervezve. Tipikusan 2x16=32 bemeneti és 2x16=32 kimeneti vonal áll rendelkezésre a környezettel való kommunikációhoz. A bemeneti/kimeneti vonalak nem TTL szintűek, a robot környezetében lévő technológiai berendezések (szállítószalag, tároló pufferek stb.) vezérlése céljából a logikai szintek low=0V, high=24V. A digitális bemeneti/kimeneti egységeket a környezet berendezéseivel kábelekkel össze kell kötni. A lábkiosztásnak és ARPS szinten a logikai jelek címeinek összhangban kell lenniük.



3.3. ábra. A Nokia PUMA 560 robot fontosabb egységei



3.4. ábra. A Nokia PUMA 560 robot vezérlő pultja

A vezérlő pult (Control Panel) kezelőszerveit a *3.4. ábra* mutatja be. A tápfeszültség (POWER) bekapcsolása biztonsági megfontolásból csak kulccsal lehetséges. Ekkor már működik a tápellátás, de a mikroprocesszorok még nem kapják meg a tápfeszültséget. Hasonlóan nem működnek a szervohajtások, azaz a szabályozási körök nem zárulnak, hanem tápellátást nem igénylő mechanikus fékrendszer rögzíti a robotot az utolsó konfigurációban.

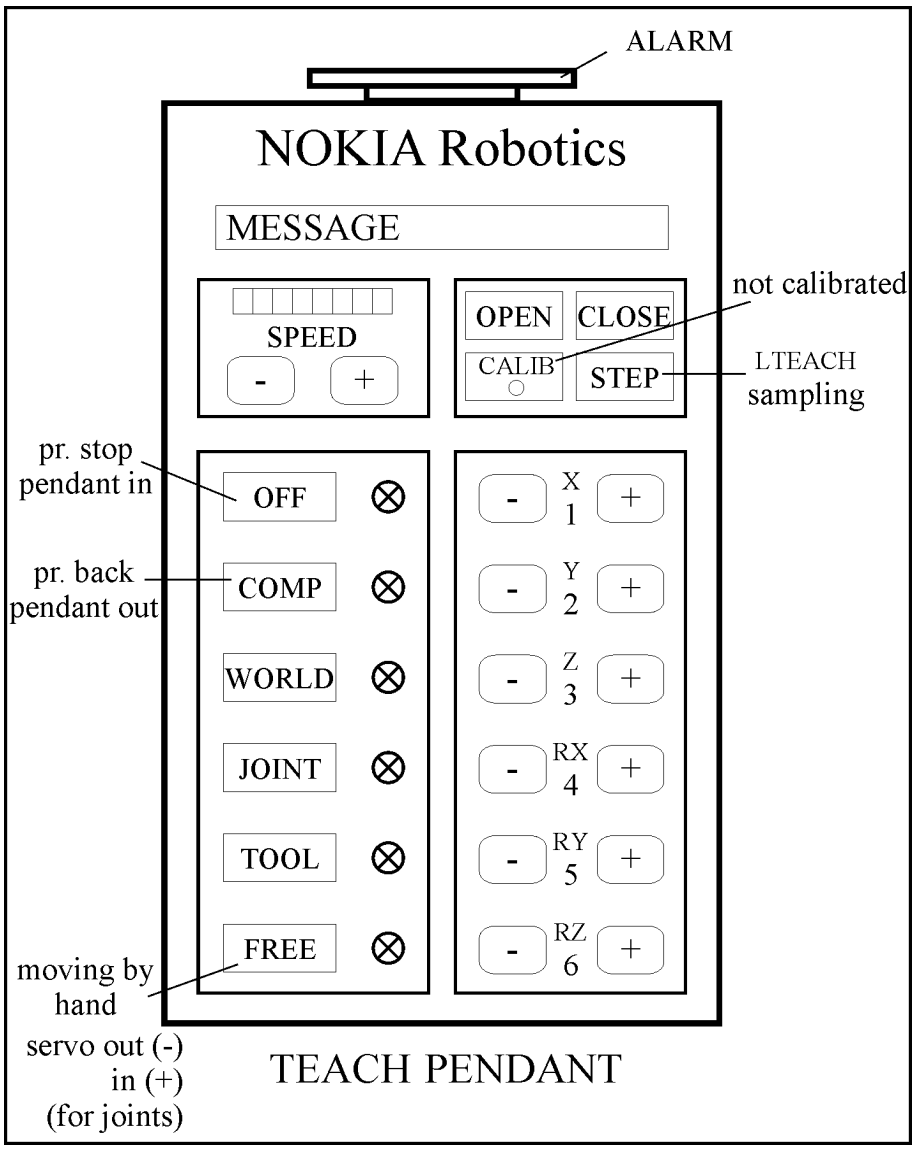
A μP_{1-7} kapcsolóval bekapcsolható a mikroprocesszorok tápfeszültsége. A `Servos` kapcsoló bekapcsolásakor a mechanikus fékek elengednek, és a szervohajtások szabályozási körei zárulnak. A robotot az aktuális konfigurációban most már a szabályozási körök tartják meg.

A SYSTEM kapcsoló `run` állásában elidítható a robotprogram, `halt pr.` állásában pedig az aktuális utasítás befejeztével átmenetileg felfüggesztődik a robotprogram utasításainak végrehejtása, ahonnan ismét `run` állásba kapcsolva a programvégrehajtás a soronkövetkező utasítással folytatható. A `restart pr.` kapcsolóval újraindítható egy megkülönböztetett program, aminek ciklikusan működtetendő kitüntetett robotprogram és újraidulás esetén van jelentősége.

Mivel a robot mozgó rendszer, és bár a robot csuklók el vannak látva a működési tartományaikat behatároló végálláskapcsolókkal, semmi sem véd az ellen, hogy a robot szegmensei egymásba mozduljanak. Ez ugyanis a számítógép grafikában valós időben észlelendő és előre jelzendő bonyolult test-test probléma lenne, amelyre jelenleg még nincs valós idejű és olcsó megoldás. Hasonlóan nem észlelhető előre, ha a robot a kezelő személy, a robot környezetében lévő más személy vagy egy akadály felé közeledik.

Ezért alapvető fontosságú a piros színű ALARM gomb a vezérlő pulton, amelyet meg kell nyomni, ha vészhelyzet vagy más probléma állna elő. Ekkor a robot szervorendszerének szabályozó körei felnyílnak, meghúznak a mechanikus fékek és a robot az utolsó konfigurációban arretál. Figyelem, az utolsó konfiguráció lehet ve-

szélyes konfiguráció is a robot vagy környezete számára! Ezért a mérés során maximális körültekintéssel és a mérésvezető utasításait maximálisan betartva kell viselkedni. A vészhelyzet felszámolása után a robot újraindítása mechanikai szempontból kritikus, ezért csak kiképzett szakember végezheti el az újraindítást.



3.5. ábra. A betanító pult felépítése

A betanító pult (Teach Pendant) kezelőszerveit a *3.5. ábra* mutatja be. Mivel betanításkor rendszerint eltávolodunk a vezérlő pulttól, ezért a betanító pulton is található egy piros ALARM gomb, amit vészhelyzetben a vezérlő pulthoz hasonlóan meg kell nyomni.

A betanító pult alaphelyzetben élesítetlen, élesítése a betanítás céljából az OFF gombbal lehetséges. Hatására leáll a program végrehajtás az éppen végrehajtás alatt álló utasítás befejeztével, és bekapcsol a betanító pult. Fordítva, a COMP gomb megnyomásakor passziválódik a betanító pult és folytatódhat a félbehagyott program végrehajtása. Bekapcsolt pult esetén ég a CALIB led.

A pult használata csak a pult bekapcsolt állapotában lehetséges. A betanító pulton 2x6 gomb található $X\,1,\ldots,RZ\,6$ felirattal, amelynek értelmezése a pult aktuális állapotától függ, az aktuális állapotot a JOINT, WORLD, TOOL gombokkal lehet beállítani.

JOINT állapotban a 2x6 gomb az egyes csuklókhoz van hozzárendelve, megnyomásukkal a megfelelő csukló továbbmozdul + vagy – irányban. Az egy megnyomás hatására történő elmozdulás mértéke 8 fokozatban a SPEED mező világító ledjeinek növelésével növelhető, világító ledjeinek csökkentésével pedig csökkenthető. A növelés/csökkentés mértéke a SPEED mező + és – gombjaival állítható. A betanítás során a betanítandó célhelyzettől távol általában nagyobb sebességgel végezzük a betanítást, a célhelyzet közelében azonban kis sebességre állunk át, hogy vizulás megfigyelésünkre alapozva nagy pontossággal állíthassuk be a betanítandó konfigurációt. Ha a kívánt robot konfigurációt elértük, a STEP gomb megnyomásának hatására a rendszer mintát vesz a csuklóváltozók értékéből, és hozzárendeli a csuklóvektor értékét a betanításkor megjelölt változóhoz. A csuklóváltozók neveit az ARPS a `#name` módon különbözteti meg az ún. koordináta-pontoktól, amelynek neve előtt nem áll # (hesh mark).

WORLD állásban koordináta-pontokat (coordinate points) lehet betanítani, amelyek (x, y, z, o, a, t) alakban tárolják a betanított pozíciót és az Euler-szögekkel kifejezett orientációt. Koordináta-pont neve előtt nem állhat #. WORLD állásban a 2x6 gomb értelmezése lineáris mozgás a bázis koordináta-rendszer x_B, y_B, z_B tengelyeinek irányában (első három gombpár), vagy forgatás a bázis koordináta-rendszer x_B, y_B, z_B tengelyei kürül (utolsó három gombpár). A STEP gomb megnyomásának hatására az aktuális konfigurációhoz tartozó $T_{B,E}$ információ (x, y, z, o, a, t) alakban hozzárendelődik a betanításkor megjelölt `name` koordináta-ponthoz.

TOOL állásban az eljárás analóg a WORLD állásban ismertetetthez, de az irányok a K_E végberendezés tengelyeinek irányában értendők. Az első három gombpár hatására az x_E, y_E, z_E tengelyek irányában, tehát saját magához képest relatíve végez lineáris mozgást a végberendezés, illetve az utolsó három gombpár ha-

tására a saját tengelyei körül végez forgást a végberendezés. Minden lépés után megváltozik K_E helyzete. A STEP gomb megnyomásának hatására most is az aktuális konfigurációhoz tartozó $T_{B,E}$ információ rendelődik hozzá (x, y, z, o, a, t) alakban a betanításkor megjelölt `name` koordináta-ponthoz.

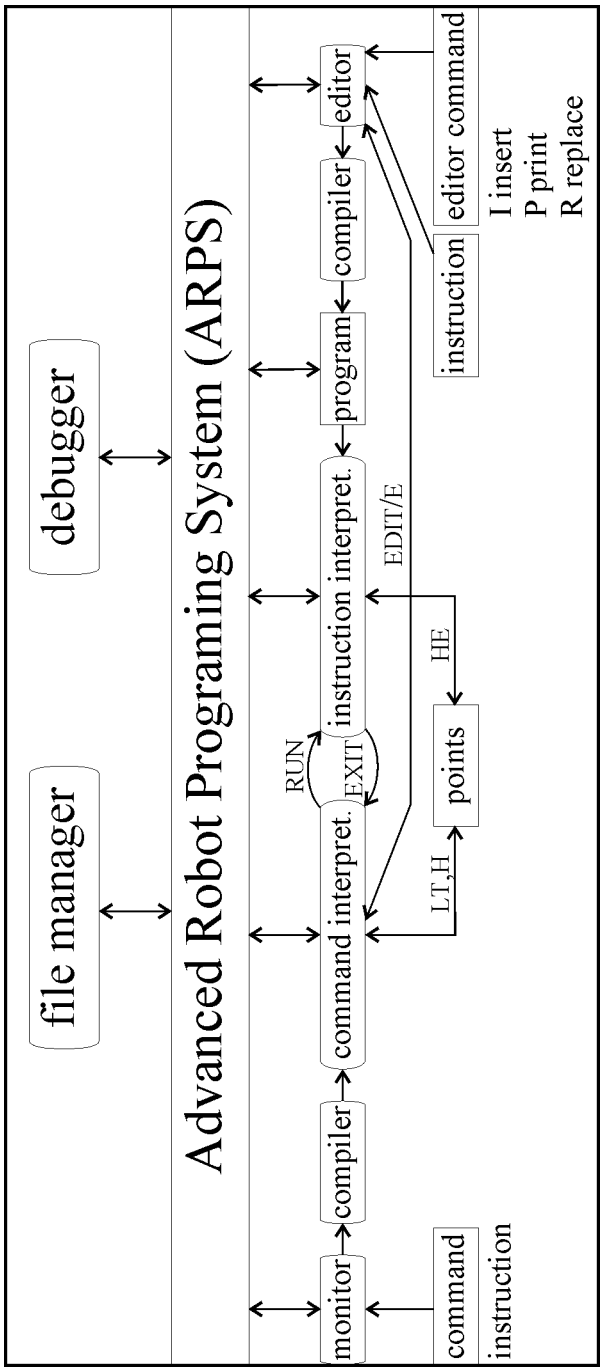
Van még egy FREE állapot is, amelyet különös figyelemmel szabad csak használni. Ez az állapot arra szolgál, hogy kézzel mozgassuk a robotot a betanítandó célhelyzetbe. Mivel azonban a viszonylag nagy (kb. 100 nagyságrendű) áttételekkel szemben nehéz mozgatni a robotot, ezért a FREE állapotban felszakíthatjuk a kiválasztott csuklóhajtás szabályozási körét (esetleg egynél többet is) a 2x6 gombpár megfelelő + gombjával, de ekkor sem a hajtás, sem pedig a fék nem aktív, ezért az összecsuklás ellen a robotot meg kell tartani a felszabadított csuklók mozgatása közben. STEP gomb hatására a csuklóváltozók értéke most is kimenthető a `#name` változóba. Ez a folyamat a robot mozgatása és a betanító pult kezelése miatt legalább két személy közreműködését igényli. A mozgatás végén a csukló – gombjának megnyomására a felnyitott hajtásszabályozási kör ismét zárul, és a hajtás megtartja a csukló szegmensét. Ha az összes felszabadított csukló szabályozási körét ismét zártuk, elhagyhatjuk a FREE állapotot.

A betanító pult rendelkezik még egy MESSAGE kijelző mezővel is, amelyben hiba esetén a rendszerint fatális hibára jellemező üzenetek jelennek meg, melyet a rendszer generál. Fatális hiba esetén kiképzett szakember tanácsát kell kikérni a folytatáshoz. Hiba keletkezik akkor is, ha a robot elhagyná a csuklóváltozók megengedett tartományát, vagyis a határoló végálláskapcsolók megszólalnak. Az első 5 csukló működési tartománya kisebb, mint a $[0^\circ, 360^\circ]$ tartomány, a hatodik csukló esetén azonban körbefordulás is lehetséges.

3.3 Az ARPS robotprogramozási rendszer

Az ARPS (Advanced Robot Programming System) robotprogramozási rendszer felépítését a *3.6. ábra* mutatja be. A file manager a fájlok tárolását/lekérdezését felügyeli a floppy lemezes háttértárolón. A debugger szervíz célokat szolgál (hibakeresés és elhárítás), a felhasználó számára nem érhető el.

Az ARPS rendszernek belső integráns része egy real time kernel, amely a feladatok ütemezését, az időzítéseket és az eseményeket kezeli. Az ARPS rendszer, eltekintve a kerneltől, egy kétszintű programrendszer. Alacsony szinten fut a robotprogram vagy az editor egymást kizárva, az egyszerűség kedvéért nevezzük ezt background-nak. A robotprogram utasításai (instruction) az editor segítségével vihetők be a rendszerbe. Az editor sor-orientált, minden nem comment sor egy utasítást definiál. Magasabb szinten fut az ún. monitor, amelynek az operátor kommunikáció keretében megadható parancsaival (command) a robotprogram futása közben is beavatkozhatunk a rendszer működésébe. Nevezzük ezt a magasabb szintet az egyszerűség kedvéért foreground-nak.



3.6. ábra. Az ARPS robotprogramozási rendszer felépítése

Az ARPS robotprogramozási nyelv egy egyszerű nyelv, valahol a BASIC és a FORTRAN határán. Az egyszerűséget a gyakorlat elvárásai definiálják: A felhasználó közepesen kvalifikált, főiskolai/egyetemi végzettség nem igény. Az utasítások cimke, mnemonik, operandus struktúrájúak. A cimke decimális szám, az operandusok integer típusúak, az operációk aritmetikai és logikai utasítások lehetnek, a logikai feltételek alapján elágazó utasítások lehetségesek.

Az utasításokban megengedett operátorok: $+ \mid - \mid * \mid / \mid MOD \mid AND \mid OR$

Relációjelek: $= \mid <> \mid < \mid > \mid \leq \mid \geq$

Bár az utasítások egyszerűek, az adatstruktúrák kielégítik a magasabb robotikai elvárásokat. A változók között robotikai célra megengedett a q csuklóvektor (precise point, #name), az x, y, z pozíció vektorral és az o, a, t Euler-szögekkel jellemzett koordinata pont (coordinate point, name), valamint relatív pont (combined point), inverz pont és tükörpont (mirror point). A változók másolhatók (locate), és rajtuk egyszerű adat transzformációk (shift stb.) írhatók elő.

Magas szintű adatstruktúra például a frame, amely koordináta-rendszert definiál. A definíció praktikus, nevezetesen előírható az origó, az x-tengely egy pontja, valamint az xy síknak egy $+y$ oldali pontja. A praktikus alatt az értendő, hogy be lehet tanítani az origót és az x-tengely egy pontját, viszont nehéz elvárni, hogy az x-tengelyre merőleges y-tengely egy pontját is be tudjuk tanítani. Ezért a rendszer csak az xy sík egy pontjának betanítását igényli (például a munkaasztal, work table egy pontjáét), amely könnyen betanítható, és a rendszerre bízza ezekből az információkból a jobb sodrású koordináta-rendszer megkonstruálását.

Alkalmazási oldalról különösen fontos a futó robotprogram számára, hogy egy utasítás megkezdése egy esemény bekövetkezéséhez, nevezetesen egy digitális bemenőjel előírható szintjének megjelenéséhez köthető. A robotprogram futása ilyenkor felfüggesztődik, az eseményre való várakozás igénye a real time kernel eseménytáblázatába beíródik, amelyet a kernel ciklusidejétől függő gyakorisággal ellenőriz, és ha a jelszint bekövetkezését tapasztalja, akkor a felfüggesztés alatt álló programot tovább folytatja.

Az egyszerű utasítások ellenére a rendszer intelligenciájára utal, hogy a digitális rendszer jeleinek éleire megszakítási szubrutint lehet felkötni, amely a robotprogram futása közben is lefut, ha a digitális jel élén az előírt változást tapasztalja a real time kernel. A megszakítási szubrutin szintén egy program, amelynek utasításai az editor segítségével definiálhatók. Természetesen a környezetet kiszolgáló megszakítási szubrutinnak rövid lefutásúnak kell lennie, hiszen beékelődhet a robotprogram mozgásutasításai közé, amelyek időkritikusak lehetnek.

A gyakorlat számára fontos további specialitás, hogy lehetséges mozgást előírni egy egyenes irányában, és az egyenes körül szinuszos oszcillációs mozgást kérni,

amelynek amplitúdója előírható, és megadható egy idő is, ameddig a csúcsértéket tartani kell. Ez a WEAVE utasítás, amely jól használható például kémiai alkalmazásoknál, ahol egy egyenes mentén valamilyen port/vegyületet kell szórni.

Az ARPS nyelv a digitális jelekre a címükkel (pozitív integer a bemenő vagy kimenő jelek egységeiben, összhangban a kábelekben használt lábkiosztással), valamint a cím előtt álló + vagy – előjellel hivatkozunk, ahol `+address` a magas szintű jelet, `-address` az alacsony szintű jelet jelöli, a default érték +, amely el is hagyható.

Tekintsük most részletesebben a *3.6 ábrát*. Látható, hogy az ARPS rendszert a monitor felügyeli a command-jaival. A monitor parancsok sikeres ellenőrzés után (compiler) eljutnak a command interpreterhez, és elkezdődik végrehajtásuk.

Programíráskor elsődleges a robotprogram utasításainak megadása a program számára. A program editálás az EDIT monitor paranccsal indítható el. Az editor sororientált, a C kezdetű comment sorok kivételével minden program utasítás (instruction) egy forrássorban adandó meg. Az utasítás sor cimkéből, mnemonikból és operandusokból áll, a cimke decimális szám. A sor vége hatására megtörténik a program utasítás ellenőrzése (compiler), és a helyesnek talált utasítás bekerül a program következő utasításának helyére a memóriában. Hiba esetén üzenet keletkezik és a hiba kijavítható. Az editor is rendelkezik egybetűs parancsokkal, ezek közül a fontosabbak az I (insert), D (delete), R (replace), valamint az editálás végét jelző E (exit editor) parancs.

A T GO (teach go) vagy a T GOS (teach go stright line) speciális editor utasítás hatására utasítássorozatot (sarokpontokat) lehet definiálni a betanító pult bevonásával a STEP gomb megnyomására, amely a megfogó nyitott (open) vagy zárt (close) állapotát is megjegyzi:

$$GO[S]\,\&\left\{\begin{matrix} OPEN \\ CLOSE \end{matrix}\right\}[\#]namei$$

A nevet kiegészítő i érték nulláról indulva miden STEP után eggyel nő.

A robotprogram elindítása a RUN monitor parancs hatására történik meg. Az instruction interpreter értelmezi a soron következő utasítást és végrehajtja azt. A mozgásutasítások hatására megtörténik a pályatervezés és a pálya megvalósítása az irányító rendszer bevonásával. A mozgás sikeres befejeztével a rendszer visszatér az instruction interpreterhez, amely megkezdi a soron következő utasítás végrehejtását. HE (here) utasítás hatására a rendszer kiolvassa az érzékelők aktuális értékét, amely az aktuális robot konfigurációra jellemző, és az értéket csuklóvektor vagy pozíció és orientáció alakjában az utasításban megadott változóhoz rendeli. STOP utasítás hatására a program leáll, de a monitor megfelelő parancsának hatására folytatható, HALT utasítás hatására nem folytatható. EXIT utasítás hatására a program visszatér a monitorhoz.

A monitor H (here) parancsa hatására betanítás közben is változóhoz rendelhető az érézkelőknek az aktuális robot konfigurációhoz tartozó értéke csuklóvektor vagy pozíció és orientáció alakjában. A monitor LT (location teach) parancsának hatására sarokpontok sorozata tanítható be a betanító pult bevonásával. A betanító pult STEP gombjának minden egyes megnyomásakor az információ hozzárendelődik a monitor parancsban megadott nevű változóhoz, amelynek neve rendre kiegészül a $0,1,2,\ldots$ kiegészítő karakterrel.

A program végrehajtásának vezérlését program kapcsolók (program switches) felügyelik, melyeket a monitor parancsaival lehet beállítani, lásd *3.2. táblázat*.

3.2. táblázat. Program kapcsolók az ARPS rendszerben

BREAK	Stop in every end point of the path segment
DIST10	Distances in 0.1 mm
INCALL	Enable of external interrupts
PRINT	Enable of message print
SERROR	Enable of servo error detection

A kiemelt fontosságú mozgásutasításokat a *3.3. táblázat* foglalja össze. A táblázatban [ ] opcionális paramétert jelöl.

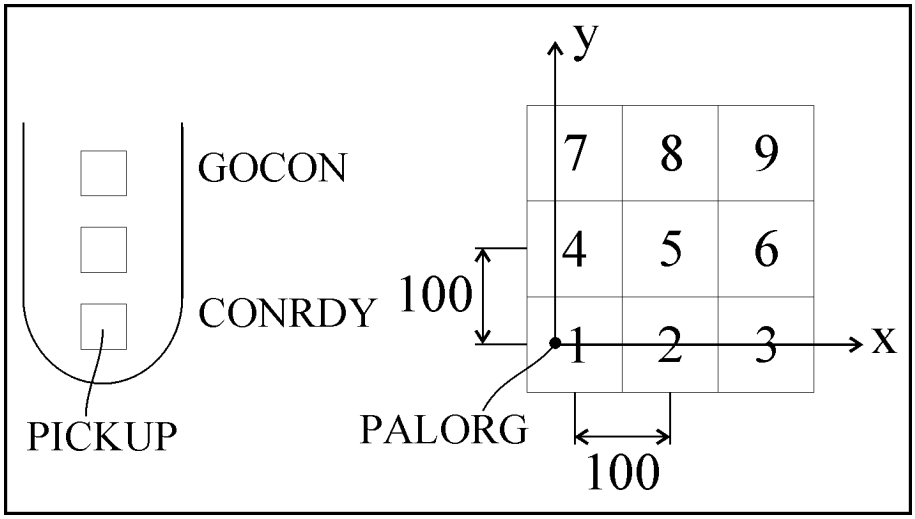
3.3. táblázat. Mozgásssal kapcsolatos utasítások az ARPS rendszerben

Short form	Instruction	Function
GO	GO location	Go to location (path design in q)
GOS	GOS location	Go to location along stright line
GON	GONEAR [loc], distance	Approach [locA,locp-d∗locn] (path design in q)
GOSN	GOSNEAR [loc], distance	Approach [locA,locp-d∗locn] along stright line
GO&C	GO&CLOSE location	GO and CLOSE during motion
GO&O	GO&OPEN location	GO and OPEN during motion
GOS&C	GOS&CLOSE location	GOS and CLOSE during motion
GOS&O	GOS&OPEN location	GOS and OPEN during motion
GOS&W	GOS&WEAVE location	GO and WEAVE during motion
GO READY	GO READY	Go into home position
LOC	LOCATE loc1=loc2	Copy loc2 into loc1
SH	SHIFT loc=[dx][,dy][,dz]	Shift location in base axis direction by saving the orientation
W I	WAIT IN in1[,in2][,in3][,in4]	Wait for in1&in2&in3&in4
OUT	OUT out1[,out2][,out3][,out4]	Set digital outputs

Az ARPS nyelvben ún. szubrutinok is képezhetők, melyek CALL utasítással hívhatók meg, és visszatérés belőlük a hívóhoz a RETURN utasítás hatására történik meg. A szubrutin híváskor nincs paraméterátadás, mivel az ARPS nyelvben minden változó globális, ezért a hívó és a meghívott ugyanazokat a változókat látja.

3.4 Az ARPS utasításainak illusztrálása: palettázó program

Az ARPS fontosabb utasításainak használatát egy palettázási program megírása keretében mutatjuk be. A feladat külső hardver eszközeit a *3.7. ábra* tartalmazza.



3.7. ábra. A conveyor és paletta fontosabb jellemzői

A szállítószalag (conveyor) tányérokat tartalmaz, amelyeken tárgyak érkeznek. A conveyor a GOCON digitális jel hatására indítható, és addig mozog, amíg a soron következő tányér az ún. PICKUP pozíciót el nem éri. A PICKUP pozíció korábban be lett tanítva, és értékét az ilyen nevű változó tartalmazza. Ha a soron következő tányér eléri a PICKUP pozíciót, megszólal egy végálláskapcsoló, amely generál egy digitális CONRDY kész jelet. Hatására a conveyor le is áll, és egy újabb GOCON hatására ismét elindul.

A conveyor hardver részleteivel nem foglalkozunk. A robot, mint intelligensebb komponens szempontjából értelmezzük az irányokat, eszerint a robot szemszögéből, amelyen a megírandó roboprogram futni fog, GOCON kimenő jel és CONRDY bemenő jel. A robot digitális I/O egysége kábellel össze van kötve a conveyor hajtás vezérlő egységével. Feltesszük, hogy a kábelezés szerint GOCON a robot digitális kimeneti egységének **3**, CONRDY pedig a digitális bemeneti egységének **2** címére

van kötve. Ne feledjük, hogy +address magas (high) szintet jelöl, és + el is hagyható, mert + a default érték. A jelek logikai true értéke a high szint (24V).

A környezet másik komponense egy paletta, amelynek 3x3 rekesze van. A rekeszek sor-iránya és oszlop-iránya a bázis koordináta-rendszer x_B és y_B irányával esik egybe, amely fontos lesz a SHIFT utasítás használhatósága szempontjából. (A program bonyolultabb lenne, ha ez a megszorítás nem teljesülne, mert alkalmazni kellene a FRAME utasítást az általános irányok miatt, továbbá a SHIFT utasítás nem lenne alkalmazható). A feladat a conveyor-on érkező tárgyak elhelyezése a paletta egymásután következő rekeszeiben, ecélból a rekeszek meg vannak számozva. Az 1 indexű rekesz korábban be lett tanítva, helyzetét a PALORG változó azonosítja. Az alkalmazás esetleges újraismétlési igénye miatt PALORG értéke nem veszhet el.

A programot keretprogramra és szubrutinra bontjuk. A keretprogram elvégzi a ciklikus műveletek szervezését, a szubrutin pedig a conveyor-on érkező soron következő tárgyat elhelyezi a paletta soron következő rekeszébe. Ennek során a megfogó ütközését a tányérral és a benne lévő tárggyal, vagy a megfogóval már megfogott tárgy ütközését a paletta rekeszével meg kell akadályozni. Ezért a célhelyzeteket előbb megközelítjük (near), amely történhet csuklóváltozóban végzett pályatervezéssel (GONEAR), és a célhelyzetet csak ezután érjük el, kötelezően egyenes mentén mozogva (GOS). Hasonlóan járunk el eltávolodáskor is fordított sorrendben (GOSNEAR, GO). A megközlítésnél használt offset a tányér, tárgy és rekesz geometriai méretének ismeretében választandó meg, itt feltesszük, hogy 50 mm elegendő.

Egy másik fontos szempont több mozgó berendezés esetén a párhuzamos mozgatási lehetőségek kihasználása. Ehhez egyszerre célszerű indítani a conveyor mozgást és a robot mozgását a PICKUP megközelítése felé (de nem az elérése felé, mert akkor ütközne az érkező tányérral benne a tárggyal).

A robot a PICKUP pozíció elérését nem kezdheti meg addig, amíg a conveyor el nem érte a PICKUP pozíciót, azaz meg nem jött a CONRDY jel. Erre a célra felhasználható a WAIT IN CONRDY utasítás, amely addig felfüggeszti a mozgásutasítás elindítását eseményre várva, amíg a CONRDY jel meg nem jön. Az eseményfigyelést az ARPS real time kernel elvégzi.

Ügyeljönk a keretprogramban a ciklusváltozók helyes módosítására, összhangban a megoldandó feladattal és a paletta méretével.

Célszerű szoftverben könnyen lecserélhető címkiosztást alkalmazni, ezért a kimenő jel és bemenő jel címét a keretprogramban a jellel azonos nevű integer változókban GOCON=3 és CONRDY=2 értékűre állítjuk. Mivel a jelek high szintjének default jelölése a +, ami ezért el is hagyható, ezért OUT GOCON ekvivalens OUT +GOCON-nal, és ezért az előbbit használjuk. Ne feledjük azonban, hogy a címkiosztás a kábelezéssel is kapcsolatban áll, ezért a kábelezés módosításakor módosítani kell a címeket is a keretprogramban.

```
                C*****************************************************
                C                FRAME PROGRAM
                C*****************************************************
                LOCATE           PALLET=PALORG
                SET              GOCON=3
                SET              CONRDY=2
                SET              PX=1
                SET              PY=1
10              OUT              GOCON
                CALL             ONEOBJ
                IF               PX=3 THEN JUMP 20
                SET              PX=PX+1
                SHIFT            PALLET=100,0,0
                JUMP 10
20              IF               PY=3 THEN JUMP 30
                SET              PX=1
                SET              PY=PY+1
                SHIFT            PALLET= –200,100,0
                JUMP 10
30              HALT
                C*****************************************************
                C             END PROGRAM
                C*****************************************************
```

```
                C*****************************************************
                C          SUBROUTINE ONEOBJ
                C*****************************************************
                GONEAR           PICKUP,50
                WAIT IN          CONRDY
                GOS              PICKUP
                CLOSE
                GOSNEAR          PICKUP,50
                GONEAR           PALLET,50
                GOS              PALLET
                OPEN
                GOSNEAR          PALLET,50
                RETURN
                C*****************************************************
                C             END ONEOBJ
                C*****************************************************
```

4. ROBOTPROGRAMOZÁS I. Mérési segédlet.

Összeállította: Halász Péter, dr. Laczházi Gyula, Dr. Lantos Béla
Átdolgozta: dr. Halász Péter, Dr. Lantos Béla

A mérés célja, előismeretek

A mérés Nokia Puma 560 (RM-01) robot ARPS nyelven történő progamozásával ismertet meg egy egyszerű anyagmozgatási feladat keretében. A mérés kiterjed a lényeges pontok betanítására a pult segítségével, az előírt mozgás programjának megírására, editálására, tesztelésére és üzemszerű használatára.

Az elvégzendő mozgatási feladat

Az anyagmozgatási feladat során egy palettamodellben (alsó sorban) helyet foglaló tárgyakat (3db hengeres tárgy) kell sorban kiemelni, lerakni egy tároló modellben, majd fordított elhelyezkedési sorrendben visszahelyezni a palettamodell másik (felső) részére.

Az alaplemez vázlatát a melléklet tartalmazza.

Előzetes tudnivalók a berendezésekről

A mérések elvégzéséhez szükséges ismereteket az 5-éves képzésben a Villamosmérnöki Szakon a "Robotok irányítása" és/vagy a "Robotirányítás rendszertechnikája", Műszaki Informatika Szakon az "Autonóm robotizált rendszerek" című tárgyak tartalmazták.
Az új BSC képzésben a mérés elvégzéséhez szükséges ismereteket a jelen ROBOTOK PROGRAMOZÁSA segédlet 1-3. fejezete tartalmazza.
Az ARPS robotprogramozási nyelv használatát segítő segédletet a melléklet tartalmazza.

A feladatok elvégzése maximális odafigyelést igényel, hogy a robot megsérülését és a balesetveszélyt elkerüljük. Vészhelyzetben a pulton vagy a vezérlőszekrényen lévő vészleállító gombbal a működést azonnal le kell állítani. A mérés szigorúan csak a mérésvezető oktató felügyelete mellett végezhető. A betanítást kis sebességek mellett kell végrehajtani, és a finombeállítást ezen belül is kisebb sebességnél végezzük.

A program tesztelése először mindig kis sebességnél történjen, majd az oktató előírása szerinti sebességnél szabad megismételni. A program tesztelése közbeni bármi-

féle hiba, esetleges ütközéshez való közeledés esetén a program futását meg kell szakítani.

A hengeres tárgynak a furatba történő behelyezése különös figyelmet igényel, a fokozott ütközésveszély miatt.

A robot sérülését okozó, figyelmetlenségből fakadó hibák esetén a mérés nem folytatható. Az elégtelen osztályzat miatt a mérést pótolni kell.

Előzetes felkészülés, otthoni feladatok

A mérésre való előzetes (otthoni) felkészülés során megválaszolandó/eldöntendő kérdések, elvégzendő feladatok:

- Érteni kell és helyesen használni a GO és GOS típusú mozgásutasításokat, köztük a különbségeket.
- Érteni kell a GONEAR utasítás pontos használatát.
- Érteni kell a SHIFT utasítás használatát.
- Érteni kell a kombinált pontok használatát, annak hatását (lásd ARPS összefoglaló és annak Függeléke).
- Érteni kell a betanító pult használatát.
- El kell dönteni, hány darab pont betanítása szükséges a feladat elvégzéséhez (a megfogó pontos geometriáját nem ismerjük), és mely(ek) legyen(ek) ez(ek). Törekedni kell arra, hogy elkerüljük a feleslegesen sok pont betanítását.
- Meg kell írni a feladatot elvégző ARPS programot.
  Szempontok a program felépítéséhez:
  - Az előzetesen betanított pontok helyzete a program futása során ne változzon (annak érdekében, hogy a program bárhol is állt meg előzőleg, azonnal újraindítható legyen).
  - A program ne tartalmazzon sebességet befolyásoló utasítást (azt majd kézzel állítják be futás során).
  - A progamban ismétlődő mozgássorozatot lehetőség szerint szervezzék ciklusba, esetleg alprogramba.

A mérés módja

A mérés során

- a mérésvezető oktató ellenőrzi az otthoni felkészülést,
- a hallgatók szükség szerint pontosítják a magukkal hozott programot,
- a hallgatók beírják a programot a vezérlőbe,
- elvégzik a szükséges egyéb beállításokat, betanításokat,
- kis sebességnél tesztelik a programot,
- végül nagyobb sebességgel is végrehajtják a tesztelt programot.

A mérésről készülő jegyzőkönyv tartalma

A mérésről mérőcsoportonként kell egy példány jegyzőkönyvet készíteni. Ezt legkésőbb a mérés elvégzése után egy héttel kell a mérésvezetőnek leadni.
A jegyzőkönyvnek tartalmaznia kell mindazokat az információkat, tevékenységek leírását, ahogy a mérőcsoport eljutott a helyes működést adó megoldáshoz a megoldást adó program listájával együtt.

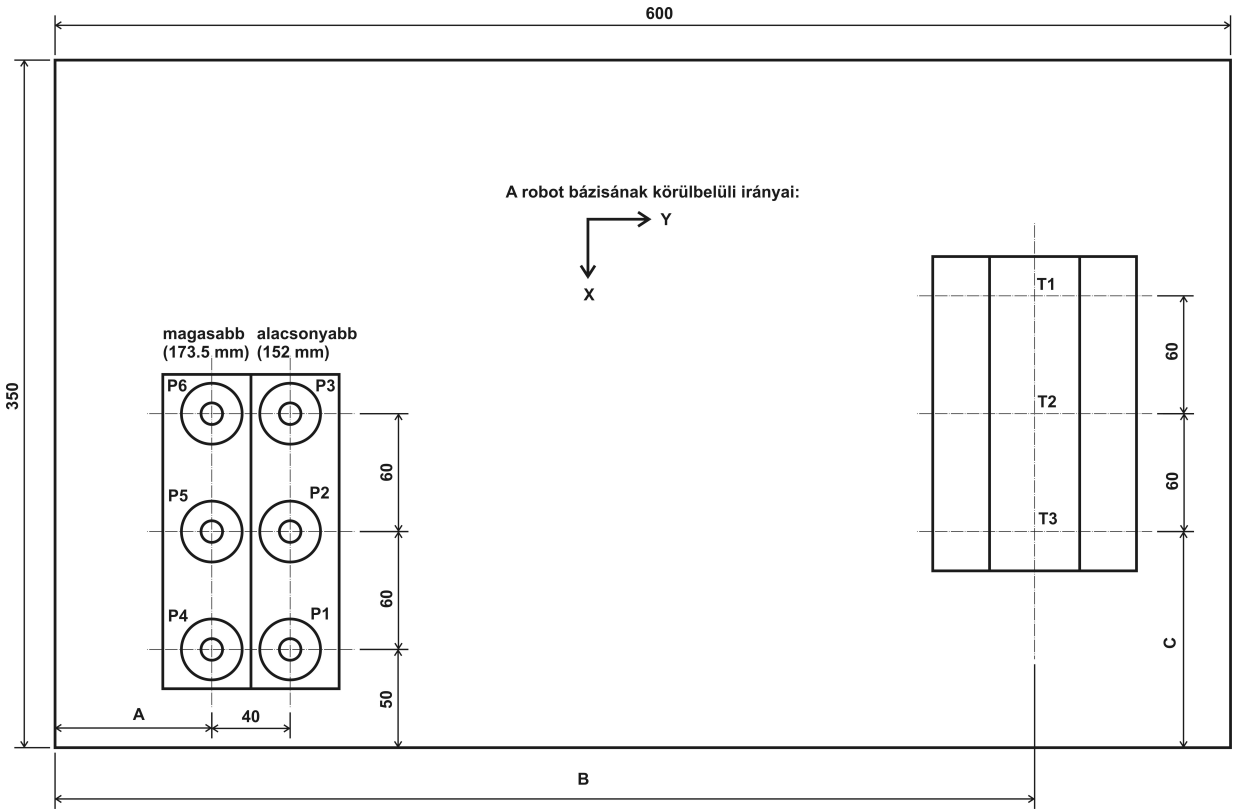
Az elvégzendő feladat paraméterei

Az "Elhelyezési vázlat"-on betűkkel jelzett méretek az egyes mérőcsoportok számára a következők (minden méret milliméterben adott az ábrán is és itt is):

Mérőcsoport sorszáma	"A" méret	"B" méret	"C" méret
1	70	490	60
2	70	490	80
3	70	490	100
4	70	510	80
5	90	490	80
6	90	510	60
7	90	510	100
8	110	490	60
9	110	490	100
10	110	510	80

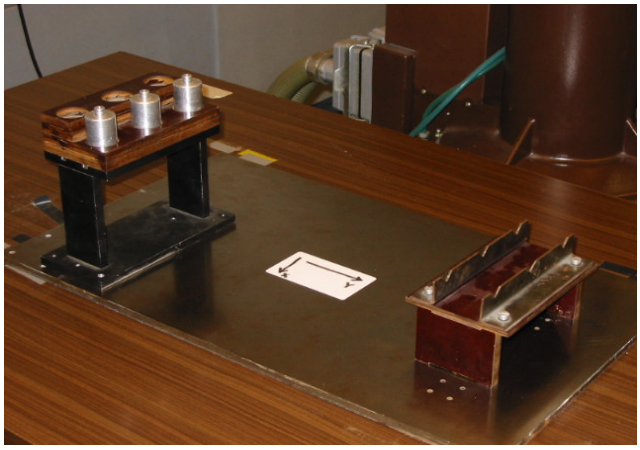
Értékelés

A mérésvezető az otthoni felkészülés ellenőrzése, a mérés elvégzése és a jegyzőkönyv alapján ad osztályzatot.

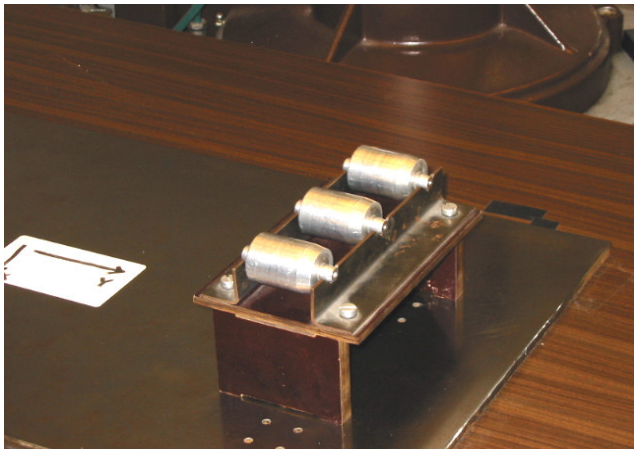


Elhelyezési vázlat

A feladat kiinduló helyzete (tárgyak a paletta alsó sorában):



Átmeneti helyzet (a tárgyak a tárolóban):



Végső helyzet (tárgyak a paletta felső sorában):

5. Az ARPS nyelv összefoglalója

Összeállította: Pilipár Gábor, Bézi István BME Automatizálási Tanszék.
Átdolgozta: dr. Halász Péter BME Irányítástechnika és Informatika Tanszék

JELÖLÉSEK

Az alábbi anyagban a következő jelöléseket alkalmazzuk:

Parancsok, utasítások nevei

Parancsok, utasítások neveiben az aláhúzott karakterek használata kötelező, az alá nem húzott nagybetűs részeket nem kötelező kiírni (rövidítések).
A [ ] jelek közé írt argumentumok, kiegészítések használata opcionális, a programozó szándékának megfelelően (ezeket a '[' és ']' jeleket nem szabad a programba beleírni).
A `courier` betűtípussal az argumentumra utalunk, annak típusát megadva a következők szerint:

Argumentumok

Jelölés	Típus, jelentés	Lehetséges érték, példa, megjegyzés
nnn	egész típusú konstans	-32768 .. 32767
var	egész típusú változó (változók helyén mindig állhat egész típusú szám is)	-32768 .. 32767
var	sebességtényező (százalékban értendő)	3 .. 300
var	digitális I/O címek	Az ARPS 1 .. 256 és -1 .. -256 értéket enged meg, a NOKIA-PUMA robot vezérlője esetén 1 .. 32 és -1 .. -32 értékek értelmesek. Pozitív értékek logikai "1"-et, negatív értékek logikai "0"-t jelentenek. Csoportos címzés esetén lehetséges értékek: 1 .. 2 (a 16 bites csoportot, mint szót címezi)
nn.n	decimális konstans	-3276.8 .. 3276.7
nn.n	sebességérték (mm/s)	2 .. 500
n.nn	decimális konstans	-327.68 .. 327.67
dis	egész típusú konstans, távolságot [mm] jelent	-1024.00 .. 1023.99

Jelölés	Típus, jelentés	Lehetséges érték, példa, megjegyzés
ang	egész típusú konstans, szögértéket [fok] jelent	-180.000 .. 179.995
jnt	egész típusú konstans, csuklóindexet jelent	1 .. 6
prg	programnév	Szimbolikus név.
file	filenév	Lehetséges formája: szimbólikus név.kiterjesztés kiterjesztés: .P program, .L pontok
aopr	aritmetikai operátor	+, -, *, /, MOD, AND, OR
cmp	összehasonlítási operátor	=, <>, >, <, >=, <=
str	szövegkonstans	Példa: ALMA
lbl	címke	pozitív egész szám
swit	programkapcsoló	Lehetséges értékek, alapértelmezés: BREAK default: DISABLED DIST10 default: DISABLED INCALLS default: DISABLED PRINT default: ENABLED SERROR default: ENABLED
loc	precíz pont (csuklókoordinátákat tárol)	Példa: #PONT
loc	koordinátapont (P=x,y,z,o,a,t -t tárol, értelmezés: alap (bázis) koordináta-rendszerben)	Példa: PONT
loc	relatív pont (egy bázisframe-nek és az ehhez képest relatív pontnak az eredő transzformációja)	Példa: BASE(PONT) (kombinált pontnak is nevezik)
loc	inverz pont	a pontnak, mint transzformációnak az inverze (Pont: alap koordináta-rendszer origójából a pontba. Inverz pont: a szerszám koordináta-rendszerében a pontból az alap koordináta-rendszer origójába)
loc	tükörpont	relatív pont egy bázisponthoz képesti speciális tükörképe bázispontra való hivatkozáskor (A relatív pont pozíciója a szerszám koordináta-rendszerének xy síkjára, orientációja pedig az alap koordináta-rendszer xy síkjára tükröződik)

MONITOR PARANCSOK

Pontok meghatározása

CHANGE `loc`

Listázza a pont értékét, meg lehet változtatni (új pont generálása). loc: precíz- vagy koordináta pont

HERE `loc`

Pont értékének bevitele a memóriába (aktuális robothelyzet szerinti új pont generálása). `loc`: precíz-, koordináta- vagy relatív pont

WHERE[#]

Folyamatosan listázza az aktuális pontot. Kilépés: RETURN billentyűvel.

LTEACH `loc`

Pont betanítási üzemmód, a betanító pulton lévő STEP gomb megnyomásakor mindig egy új pont tárolódik el. `loc`: precíz-, koordináta- vagy relatív pont. Pl: LTEACH A és a STEP háromszori használata után az eredmény A0, A1, A2

Floppy kezelés

STORE `file[=prg][,prg]...`

Floppy-ra tárol a megadott filenéven programo(ka)t és/vagy pontokat. Kiterjesztés: .P programok, .L pontok, kiterjesztés nélkül mindkettő

LOAD `file`

Program(ok) és/vagy pontok betöltése floppy-ról. Kiterjesztés: .P programok, .L pontok, kiterjesztés nélkül mindkettő

FDIR

Floppy directory - programok neve, mérete, szabad kapacitás

FPACK

Lemezen lévő adatok tömörítése, átrendezése

FDEL `file`

File törlése floppy-ról. Kiterjesztés: .P programok, .L pontok, kiterjesztés nélkül mindkettő

ZERO DISK
Floppy lemez formázása. Végrehajtás előtt megerősítést vár: Are you sure (y/n)?

Memória kezelés

PDIR
Memóriában lévő programok nevének listázása

PLIST [prg][,prg]...
Program(ok) kilistázása (név nélkül az összes memóriában lévő programot)

LLIST [loc][,loc]...
Pontok kilistázása (név nélkül az összes memóriában lévő pontot)

LDEL loc[,loc]...
Pontok törlése a memóriából

DLOAD file
Azokat a programokat és/vagy pontokat törli ki a memóriából, amelyek a floppy-n a file-ban vannak

ZERO MEMORY
Teljes memória törlése. Végrehajtás előtt visszakérdez: Are you sure (y/n)?

Program vezérlés

RUN prg[,nnn]
A megadott nevű program indítása.

nnn	nincs, vagy 0	egyszer fut le a program
	n	n-szer fut le a program
	<0	végtelen ciklusban fut

ABORT
Program megállítása (az éppen futó programlépés végén állítja meg)

CONTINUE [nnn]
A megállított program folytatása

nnn	>=0 vagy nincs	a következő lépéstől folytatja
	<0	a megszakított lépéstől folytatja

EXIT
A program befejezése (a program végén állítja le és a RUN számlálóját nullázza)

ENABLE `swit`
Programkapcsoló engedélyezése

DISABLE `swit`
Programkapcsoló tiltása

DIA `file`
Diagnosztikai programok betöltése, indítása

COM `prg`
Parancsprogram indítása (parancsprogram: olyan program, amely csak monitorparancsokat tartalmaz)

Sebesség megadása

MAXSPEED `nn.n`
Maximális sebesség megadása [mm/s]

SPEED `nn.n`
Aktuális sebesség megadása [mm/s]

SPEED% `var`
Sebességtényező megadása (3%..300%)

Egyéb

EDIT [`prg`]
Program szerkesztése. Ha nem adunk meg programnevet, akkor az utoljára szerkesztettet hozza fel

C `str`
Megjegyzés parancsprogramokban

LIST COMMANDS
Lista az érvényes parancsokról és utasításokról

LIST STATUS
Státusz listázása (BASE, TOOL értéke, kapcsolók állása, sebességek, éppen futó program aktuális lépésszáma)

CAL [nnn],[nnn],[nnn],[nnn],[nnn],[nnn]

Kalibrálás.

nnn	>0	az adott csuklót pozitív irányú mozgással kalibrálja
	<0	az adott csuklót negatív irányú mozgással kalibrálja

CLOAD file

Kalibrálási értékek betöltése file-ból

WEAVE dis[,n.nn][,n.nn]

A pásztázó mozgást paraméterezi. Paraméterek sorrendben: amplítúdó, periódusidő, tartózkodási idő.

ZERO OUTS var TO var

Digitális kimenetek nullázása

SYSMON

Rendszermonitor (ODT) hívása Végrehajtás előtt visszakérdez: Are you sure (y/n)?

.<pont>

Pont után írva bármelyik program utasítás végrehajtható monitorból

PROGRAM UTASÍTÁSOK

Pontokkal kapcsolatos utasítások

HERE `loc`
Az aktuális robothelyzetet a megadott néven eltárolja a memóriában. `loc` lehet precíz, koordináta vagy relatív pont

LOCATE `loc=`[INVERSE] `loc`
Pont másolása, átalakítása. `loc` lehet precíz, koordináta vagy relatív pont

FRAME `loc=loc,loc,loc[,loc]`
Üzemi keret megadása. Paraméterek sorrendben: origó, x tengely pozitív pontja, xy sík pozitív y oldali pontja, eltolt (végleges) origó

SHIFT `loc=[dis],[dis],[dis]`
Pont eltolása x, y, z tengelyek mentén

DISTANCE `var=loc,loc`
Két pont távolságát adja meg. Az eredmény egész!

MIRROR `loc`
Tükröző pontot definiál

NO MIRROR
A tükrözést tiltja

SCALE loc=var,var,var
A skálázás bázispontját és nagyítását definiálja. Például: 'SCALE BASE=2000,2000,2000' után 'BASE(A)' esetén A pont mindhárom tengely irányában kétszer olyan messze lesz BASE-től, mint skálázás nélkül

NO SCALE
A skálázást tiltja

Program elágaztatások

JUMP `lbl`
Feltétel nélküli ugrás a megadott címkére

IF [INGROUP] `var cmp` [INGROUP] `var` THEN JUMP `lbl`
Feltételes ugrás

IF IN `var[,var][,var][,var]`
Digitális bemenetek állapotától függő feltételes ugrás (maximum 4 bemenet ÉS kapcsolata)

CALL `prg`
Szubrutinhívás (paraméterátadás nincs, minden változó és pont globális)

RETURN [`var`]
Szubrutinból való visszatérés. Ha van `var`, akkor annyi darab programsort kihagy a hívó programból visszatéréskor. Ha szubrutin végén nincs RETURN, a program végrehajtás a szubrutin végén befejeződik.

Mozgató utasítások

GO `loc`
A megadott pontba megy interpolált pályán. Speciális eset: 'GO READY' home pozícióba megy

GOS `loc`
Egyenes pályán megy a kívánt pontba

GO&OPEN `loc`
A megadott pontba megy interpolált pályán, eközben nyitja a megfogót

GO&CLOSE `loc`
A megadott pontba megy interpolált pályán, eközben zárja a megfogót

GOS&OPEN `loc`
A megadott pontba megy egyenes pályán, eközben nyitja a megfogót

GOS&CLOSE `loc`
A megadott pontba megy egyenes pályán, eközben zárja a megfogót

GONEAR [`loc`],`dis`
Interpolált pályán megközelíti a célpontot. A mozgás végpontja az a pont, amely a megadott `loc` ponttól a szerszám -z tengelye irányában a megadott távolságra van. Ha a pontot nem adjuk meg, az aktuális pontból megy a szerszám -z tengelye irányában az adott távolságra

GOSNEAR [loc],dis

Abba a pontba megy egyenes pályán, amelytől a megadott pont a szerszám z tengely irányában az adott távolságra van. Ha a pontot nem adjuk meg, az aktuális pontból megy a szerszám -z tengelye irányában az adott távolságra

GOS&WEAVE loc

A megadott pontba megy egyenes pályán és közben pásztázó mozgást végez a szerszám xy síkjában

MOVE [dis],[dis],[dis]

Relatív mozgás az alap koordináta-rendszer x, y, z irányában interpolált pályán

MOVES [dis],[dis],[dis]

Relatív mozgás az alap koordináta-rendszer x, y, z irányában egyenes pályán

TMOVE [dis],[dis],[dis]

Relatív mozgás a szerszám koordináta-rendszer x, y, z irányában interpolált pályán

TMOVES [dis],[dis],[dis]

Relatív mozgás a szerszám koordináta-rendszer x, y, z irányában egyenes pályán

MOVE JOINT jnt,ang

A megadott csuklót az adott szöggel mozgatja

ALIGN

A szerszám z tengelyét a legközelebbi bázis koordináta-tengely irányába beforgatja úgy, hogy közben a pozíció változatlan marad. A beforgatás a két tengelyre merőleges egyenes menti forgatással történik. (A művelet elvégzése *nem jelenti azt*, hogy a forgatás után a megfogó x és y tengelyei is párhuzamos helyzetbe kerülnének a bázis bármelyik tengelyével.)

Sebesség beállítása

SPEED nn.n

Aktuális sebesség megadása [mm/s]

SPEED% var
Sebességtényező megadása (3%..300%)

SPEED NEXT nn.n
A következő mozgató utasítást ezzel a sebességgel hajtja végre

Szerszám megfogó vezérlése

OPEN
Nyitja a megfogót

CLOSE
Zárja a megfogót

ODELAY n.nn
Az OPEN utasítást paraméterezi, nyitás után a megadott ideig vár

CDELAY n.nn
A CLOSE utasítást paraméterezi, zárás után a megadott ideig vár

I/O utasítások

OUT var[,var][,var][,var]
Digitális kimenetek állítása egyedileg

OUTGROUP var=[INGROUP] var [aopr [INGROUP] var]
Digitális kimenetek állítása csoportosan

RUNOUT var[,var][,var][,var]
A megadott kimeneteket adott állapotba állítja, ha a program futása megszakad

WAIT IN var[,var][,var][,var]
Adott bemeneti kombinációra vár (ÉS kapcsolat). Speciális esetben jelre vár. Az esemény bekövetkeztéig a program további futása fel van függesztve.

INCALL var,prg [NOBREAK]
Megszakítás engedélyezése. Megszakítás esetén `prg`-ra adódik a vezérlés (a megszakításból való visszatéréskor a megszakított utasítás végrehajtására tér vissza.)

NO INCALL `var`
Megszakítás tiltása

ZERO OUTS `var` TO `var`
Digitális kimenetek nullázása

Koordináta-rendszer módosítása

BASE `[dis],[dis],[dis],[ang]`
Alap koordináta-rendszer transzformálása, a paraméterek sorrendben: dx, dy, dz, do (utóbbi az alap z tengelye körül fordul)

TOOL `[dis],[dis],[dis],[ang],[ang],[ang]`
Szerszám koordináta-rendszer transzformálása. A paraméterek sorrenben: dx, dy, dz, do, da, dt

LTOOL `loc`
Ugyanaz mint az előbb, csak egy ponttal lehet megadni a transzformációt

Konfiguráció módosítása

J2 RIGHT
Kettes csukló (váll) jobbkezes

J2 LEFT
Kettes csukló (váll) balkezes

J3 UP
Hármas csukló (könyök) felső állásban

J3 DOWN
Hármas csukló (könyök) alsó állásban

J5 PLUS
Ötös csukló (csuklóizület) pozitív

J5 MINUS
Ötös csukló (csuklóizület) negatív

Leállító utasítások

STOP ['str'][,var]
A programot úgy állítja meg, hogy CONTINUE paranccsal tovább futtatható, kiírja 'str'-t és 'var' értékét

HALT ['str'][,var]
Ugyanaz mint az előbb, de nem folytatható a programvégrehajtás

EXIT
Program futás befejezése, a program a végén áll meg (a RUN számlálóját nullázza)

Floppy kezelés

LOAD file [NOBREAK]
Betölt egy file-t a floppy-ról

DLOAD file [NOBREAK]
Azokat a programokat és/vagy pontokat törli a memóriából, melyek a floppy-n a file-ban vannak

WAIT LOAD
A floppy művelet végrehajtása után fut tovább a program

Egyéb utasítások

C [str]
Megjegyzés

COM
Monitor parancs használata programon belül, például: COM CAL

DELAY n.nn
Megadott időt késleltet

ENABLE swit
Programkapcsoló engedélyezése

DISABLE swit
Programkapcsoló tiltása

PRINT ['str'][,var]
Képernyőre való kiíratás

SET var=[INGROUP] var [aopr [INGROUP] var]
Változó értékadása

TOL NARROW
Szerszám beállítási pontosság vezérlése, pontos

TOL WIDE
Ugyanaz, mint az előbb, kevésbé pontos

WEAVE dis[,n.nn][,n.nn]
A pásztázó mozgást paraméterezi. A paraméterek sorrendben: amplítúdó, periódusidő, tartózkodási idő

EDITOR PARANCSOK

E
Kilépés (exit)

I
Beszúrás (insert). A kurrens sor elé szúr be, kiszállás: üres sor

D [nnn]
Törlés (delete). Ha `nnn` nincs megadva, akkor csak a kurrens sor törlése

P [nnn],[nnn]
Listázás. Ha nincs paraméter, akkor az első sort adja. Paraméterek: kezdősor, sorok száma

R str^str
A kurrens sorban cseréli az első string-et a másodikra (replace)

RA str^str
Ugyanaz mint az előbb, de az egész file-ra vonatkozik (replace all)

T GO loc
A következő utasításokat generálja a kézi betanítópult STEP gombja megnyomására:
ha a megfogó nyitott, akkor 'GO&OPEN `loc`'
ha a megfogó zárt, akkor 'GO&CLOSE `loc`'
a pontok koordinátái tárolódnak a memóriában növekvő indexszel

T GOS `loc`

A következő utasításokat generálja a kézi betanítópult STEP gombja megnyomására:

ha a megfogó nyitott, akkor 'GOS&OPEN `loc`'
ha a megfogó zárt, akkor 'GOS&CLOSE `loc`'

a pontok koordinátái tárolódnak a memóriában növekvő indexszel

FÜGGELÉK

Az alábbiakban néhány utasítás hatását, működésének részleteit mutatjuk be, matematikai formába öntve.

Koordinátapont eltárolása és értelmezése

$$x, y, z, o, a, t \Leftrightarrow T_{BE} = \begin{bmatrix} & & & x \\ & Euler(o,a,t) & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

`HERE B(REL)`

Tegyük fel, hogy az aktuális pozíció T_{AKT}. Az utasítás hatására a $T_{REL} := T_B{}^{-1} * T_{AKT}$ pozíciót fogja eltárolni.

`GO B(REL)`

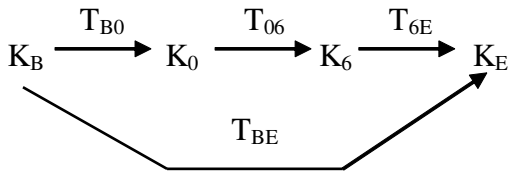
Az utasítás hatására a $T_B * T_{REL}$ pozíció lesz a mozgás célpontja.

`BASE x,y,z,φ`

(Az x, y, z, φ értékek *kizárólag* numerikus értékek lehetnek!)

Az utasítás végrehajtása után a T_{B0} transzformáció értéke a következő lesz:

$$T_{B0} = \begin{bmatrix} & & & -x \\ & Rot(z,\varphi)^{-1} & & -y \\ & & & -z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_\varphi & s_\varphi & 0 & -x \\ -s_\varphi & c_\varphi & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



`GONEAR PONT,d`

(d érték *kizárólag* numerikus érték lehet!)

Az utasítás végrehajtása végén a megfogó pozíciója:

$$T_{BE} = \begin{bmatrix} \mathbf{A}_{PONT} & \mathbf{p}_{PONT} - \mathbf{n}_{PONT} * d \\ \mathbf{0}^T & 1 \end{bmatrix}$$