Trabalho prático 2 - Robótica industrial

Artur Almeida - 123196

December 15, 2024

1 Introdução

Neste relatório será descrito o trabalho realizado para desenvolver o 2º trabalho prático de Robótica Industrial, em que envolve simular em MATLAB o robô Yaskawa SDA10F, que possui 15 graus de liberdade, mas que pode ser dividido em 2 robôs de 8 graus de liberdade que têm 1 elo em conjunto.

O robô tem como objetivo apanhar 2 blocos que vêm em tapetes diferentes, juntar os 2 blocos à sua frente, rodar 180 graus, colocá-los sobre um terceiro tapete e voltar a uma posição que volte a apanhar novos blocos.

Para a realização deste projeto, foi necessário fazer a tabela de Denavit-Hartenberg, calcular a cinemática inversa e a cinemática diferencial. No seguinte link está o vídeo realizado para o projeto https://www.youtube.com/watch?v=sGuYlxbCBMM.

2 Cinematica Direta

A primeira coisa a fazer foi desenhar os eixos no robô como se pode ver na imagem 1.

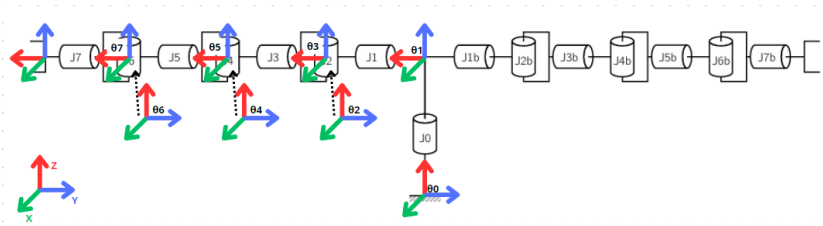

Figure 1: Eixos braço esquerdo do robô

A partir desses eixos foram feitas as tabelas com os parâmetros de DH para o braço esquerdo, tabela 1, e para o braço direito, tabela 2. É considerado o braço esquerdo do robô o braço que se encontra para Oeste e o braço direito o que se encontra para Leste.

elo	θ_i	li	di	α_i
1	0	0	H-LZ	0
2	θ_0	LX	LZ	$\pi/2$
3	θ_1	0	LA	$-\pi/2$
4	θ_2	0	0	$\pi/2$
5	θ_3	0	LB	$-\pi/2$
6	θ_4	0	0	$\pi/2$
7	θ_5	0	LC	$-\pi/2$
8	θ_6	0	0	$\pi/2$
9	θ_7	0	LD	0

Table 1: Tabela de parâmetros DH para o braço esquerdo (DH_L)

elo	θ_i	l_i	d_i	α_i
1	0	0	H-LZ	0
2	θ_0	LX	LZ	$\pi/2$
3	θ_1	0	-LA	$-\pi/2$
4	θ_2	0	0	$-\pi/2$
5	θ_3	0	LB	$\pi/2$
6	θ_4	0	0	$-\pi/2$
7	θ_5	0	LC	$\pi/2$
8	θ_6	0	0	$-\pi/2$
9	θ_7	0	LD	0

Table 2: Tabela de parâmetros DH para o braço direito (DH_R)

3 Cinemática Inversa

Um dos passos importantes deste projeto é calcular os ângulos usando a cinemática inversa, pois é usando a cinemática inversa que podemos dar ao robô um ponto na área de trabalho e a rotação desejada do end-effector e ele irá deslocar o seu end-effector até esse ponto com a rotação desejada.

Neste projeto foi usada a cinemática inversa analítica e, para calcularmos os ângulos, foi necessário fazer uma alteração temporária aos parâmetros DH de ambos os braços. Foi dividido o elo 7 em dois elos, como se pode ver na tabela 3.

elo	θ_i	li	di	α_i
1	0	0	H-LZ	0
2	θ_0	LX	LZ	$\pi/2$
3	θ_1	0	LA	$-\pi/2$
4	θ_2	0	0	$\pi/2$
5	θ_3	0	LB	$-\pi/2$
6	θ_4	0	0	$\pi/2$
6a	0	0	LC	0
7	θ_5	0	0	$-\pi/2$
8	θ_6	0	0	$\pi/2$
9	θ_7	0	LD	0

Table 3: Tabela de parâmetros DH para o braço esquerdo modificada para calculo da cinemática inversa

Como cada braço tem 8 graus de liberdade, é necessário considerar que algumas juntas serão fixas. No meu caso, foi considerado que o θ_0 será 0^{o} ou 180^{o}, dependendo do lado que o robô esteja virado, e o $\theta 3$ será 0^{o}. Com estas considerações, cada braço do robô fica sem redundâncias e é possível calcular a cinemática inversa.

Para calcular os ângulos da translação (θ_1, θ_2, θ_4), é preciso primeiro calcular a matriz de transformação ${}^0R_{6a}$.

$$\left[\begin{array}{ccc|c} - & - & - & LX + LC * (cos\theta_1\theta_2 * sin\theta_4 + cos\theta_1\theta_4 * sin\theta_4) + LB * cos\theta_1 * sin\theta_2 \\ - & - & - & -LA - LC * (cos\theta_2\theta_4 - sin\theta_2\theta_4) - LB * cos\theta_2 \\ - & - & - & H + LC * (cos\theta_2 * sin\theta_1\theta_4 + cos\theta_4 * sin\theta_1\theta_2) + LB * sin\theta_1\theta_2 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

Matriz de transformação ${}^0R_{6a}$ do braço esquerdo e com $\theta_0 = 0^{\text{o}}$

Com esta matriz de transformação ${}^0R_{6a}$, podemos obter o Pw_x, Pw_y, Pw_z para os cálculos dos ângulos. É necessário considerar que o punho do robô estará sempre virado para baixo, para poder apanhar os blocos, logo, fazemos as considerações observadas no sistema de equações 1, sendo x, y, z o ponto cartesiano final da garra.

$$\begin{cases} Pw_x = x \\ Pw_y = y \\ Pw_z = z + LD \end{cases} \tag{1}$$

Agora usando os Pw_x, Pw_y, Pw_z obtidos na matriz de transformação ${}^0R_{6a}$ podemos isolar os ângulos θ_1, θ_2 e θ_4.

- Começamos por isolar θ_4 que se obtém a partir da soma de todos os quadrados.
- Depois através de Pw_y isolamos θ_2 usando a seguinte fórmula:

$$k3 = \cos\theta(k2) + \sin\theta(k1) \Leftrightarrow \theta = 2 \cdot \arctan\left(\frac{k1 \pm \sqrt{k1^2 + k2^2 - k3^2}}{k2 + k3}\right)$$

- Por fim isolamos θ_1 dividindo Pw_z por Pw_x.

$$\begin{cases} \theta_1 = \arctan\left(\frac{(Pw_z - H)\cdot \text{sign}(LC\cdot\sin(\theta_2+\theta_4)+LB\cdot\sin(\theta_2))}{(Pw_x - LX)\cdot \text{sign}(LC\cdot\sin(\theta_2+\theta_4)+LB\cdot\sin(\theta_2))}\right) \\ \theta_2 = 2\cdot\arctan\left(\frac{LC\cdot\sin(\theta_4)\pm\sqrt{(LC\cdot\sin(\theta_4))^2+(-LC\cdot\cos(\theta_4)-LB)^2-(Pw_y+LA)^2}}{-LC\cdot\cos(\theta_4)-LB+Pw_y+LA}\right) \\ \theta_4 = \pm\arccos\left(\frac{(Pw_x-LX)^2+(Pw_y+LA)^2+(Pw_z-H)^2-LB^2-LC^2}{2\cdot LB\cdot LC}\right) \end{cases} \tag{2}$$

Com os sistemas de equações 1 e 2 é possível calcular os ângulos de translação.

Para calcular a cinemática inversa do braço direito, basta fazer as seguintes alterações antes de calcular os ângulos, estas alterações fazem mais sentido no contexto de MATLAB:

$$\begin{cases} Pw_x = -x; \\ Pw_y = -y; \\ Pw_z = -(z - LD); \\ LX = -LX; \\ H = -H; \end{cases} \tag{3}$$

Para calcular a cinemática inversa, caso o θ_0 seja 180^{o}, é necessário fazer as seguintes alterações:

$$\begin{cases} Pw_x = -Pw_x; \\ Pw_y = -Pw_y; \end{cases} \tag{4}$$

Obtendo as 4 possibilidades de ângulos da translação, são escolhidos os ângulos que impliquem um menor movimento de juntas e garantindo que os ângulos escolhidos estão dentro dos limites de juntas.

De seguida é necessário calcular os ângulos do pulso θ_5, θ_6 e θ_7. Para começar é calculada a matriz de transformação ${}^{6a}R_9$ ainda usando a tabela 3.

$$\left[\begin{array}{ccc|c} - & - & cos(\theta_5)*sin(\theta_6) & - \\ - & - & sin(\theta_5)*sin(\theta_6) & - \\ -cos(\theta_7)*sin(\theta_6) & sin(\theta_6)*sin(\theta_7) & cos(\theta_6) & - \\ 0 & 0 & 0 & 1 \end{array}\right]$$

Matriz de transformação ${}^{6a}R_9$ do braço esquerdo

Agora, tendo a matriz de transformação ${}^{6a}R_9$ podemos obter o sistema de equações 5 da seguinte maneira:

- $ax3^2 + ay3^2 = (\cos(\theta_5)^2 \cdot \sin(\theta_5)^2) \cdot \sin(\theta_6)^2 \rightarrow \frac{\sin(\theta_6)}{\cos(\theta_6)} = \frac{\sqrt{ax3^2+ay3^2}}{az3}$
- $\frac{ay3}{ax3} = \frac{\cos(\theta_5)\cdot\sin(\theta_6)}{\sin(\theta_5)\cdot\sin(\theta_6)}$
- $\frac{sz3}{nz3} = \frac{\sin(\theta_6)\cdot\sin(\theta_7)}{-\cos(\theta_7)\cdot\sin(\theta_6)}$

$$\begin{cases} \theta_5 = \arctan\left(\frac{\pm ay3}{\pm ax3}\right) \\ \theta_6 = \arctan\left(\frac{\pm\sqrt{ax3^2+ay3^2}}{az3}\right) \\ \theta_7 = \arctan\left(\frac{\pm sz3}{\mp nz3}\right) \end{cases} \tag{5}$$

A única diferença para o pulso direito é a inversão do sinal no θ_5.

$$\theta_5 = \arctan(\frac{\mp ay3}{\mp ax3})$$

Depois, sabendo a posição final e a rotação desejada da garra (${}^0R_9$) e tendo já calculados os ângulos de translação (${}^0R_{6a}$), conseguimos obter os valores da matriz de transformação do pulso usando a seguinte equação.

$${}^0R_9 = {}^0R_{6a} \cdot {}^{6a}R_9 \Leftrightarrow {}^{6a}R_9 = ({}^0R_{6a})^{-1} \cdot {}^0R_9$$

A partir da matriz de transformação ${}^{6a}R_9$ com valores, basta substituir os valores do sistema de equações 5 para calcular os ângulos do pulso.

4 Cinemática Diferencial

Outro passo importante deste projeto é calcular os ângulos usando a cinemática diferencial, pois é usando a cinemática diferencial que podemos fazer movimentos lineares em que só damos ao robô a deslocação desejada do end-effector. Um problema deste movimento é a incerteza, o movimento não vai exatamente na direção pedida mas, ao aumentar o número de iterações, NN, este erro deverá ser menor.

Sendo $\vec{dr}$ = [x y z ϕ θ ψ] o deslocamento e rotação do end-effector, $\vec{dq}$ = [θ_1 θ_2 θ_4 θ_5 θ_6 θ_7 θ_8] os valores dos ângulos das juntas e J o Jacobiano da matriz de transformação do robô, é possível obter $\vec{dq}$ fornecendo $\vec{dr}$ com a seguinte equação.

$$\vec{dr} = J \cdot \vec{dq} \Leftrightarrow \vec{dq} = J^{-1} \cdot \vec{dr}$$

Para calcular o Jacobiano foi usada a cinemática diferencial geométrica. O código usado é semelhante ao criado nas aulas práticas, os únicos cuidados a ter foram:

- Garantir que não há alterações nos ângulos dos primeiros 2 elos
- Todas as juntas são rotacionais
- Ser necessário usar a função "*pinv*" do MATLAB para inverter uma matriz não quadrada.
- Garantir que os ângulos das juntas não passam os valores estipulados.

5 *Main* e Funções criadas

O programa é corrido a partir do ficheiro *Main*. Dentro do *Main* vão ser chamadas as funções criadas de maneira a gerar o percurso do robô, guardando numa variável de 5-D os movimentos gerados. Ao fim de todo o percurso ser calculado e gerado, este será animado com a opção de mostrar a previsão do end-effector e o seu movimento feito. Durante os cálculos do percurso é considerado que o robô sabe a posição do bloco e o seu ângulo antes de este chegar ao fim do tapete.

Foram criadas as seguintes funções:

-*DrawTable*: Desenha graficamente as 3 mesas, fornecendo as dimensões e qual a mesa a ser desenhada.

-*InitRobot*: Responsável por desenhar graficamente o robô em posição zero hardware.

-*InvKinWorkArea*: Verifica se o ponto fornecido, neste caso o ponto onde serão soltos os blocos na mesa final, está dentro da área de trabalho. Como durante o passo de largar os blocos sobre a mesa final é feito com a cinemática diferencial, é preciso verificar se este ponto se encontra dentro da área de trabalho, pois a cinemática diferencial não dá erro se a mesa se encontrar fora dessa área.

-*InitBlocks*: Gera os blocos para a simulação, com uma rotação e posição no eixo y aleatória, sendo y sempre dentro dos limites do tapete.

-*InvKinMov*: Fornecendo o ponto de destino do end-effector, o ângulo desejado para o end-effector e os ângulos das juntas atuais, usando a cinemática inversa, vai retornar uma matriz 5-D com os movimentos para cada braço.

-*InvKinBraco*: Usada dentro de *InvKinMov*, é responsável por calcular os ângulos θ_1, θ_2 e θ_4, usando os cálculos descritos no capítulo 3. Garante que os ângulos não passam os limites e usa sempre a combinação de ângulos que impliquem uma menor soma do valor absoluto.

-*InvKinPulso*: Usada dentro de *InvKinMov*, é responsável por calcular os ângulos θ_5, θ_6 e θ_7, usando os cálculos descritos no capítulo 3. Garante que os ângulos não passam os limites e, para o movimento do pulso ser mais natural, usa a combinação de ângulos em que o θ_5 seja negativo para o pulso esquerdo, caso tal combinação exista, e positivo para o pulso direito.

-*JacobianMov*: Fornecendo o $\vec{dr}$ desejado para fazer o movimento linear e os ângulos atuais, devolve uma matriz 5-D com os movimentos lineares para cada braço. É garantido que os ângulos das juntas não passam os limites. Os movimentos lineares estão a ser feitos em $\frac{1}{5}$ das iterações que os outros movimentos, pois estes movimentos são normalmente feitos em menor distância e faz a simulação ser mais fluida.

-*JacobianGeom*: Usada dentro de *JacobianMov*, devolve a matriz Jacobiana, fornecendo a matriz de transformação atual.

-*J0Mov*: Fornecendo o ângulo do θ_0 e os ângulos de juntas atuais, usando cinemática direta, vai retornar uma matriz 5-D com o movimento de rodar o robô para cada braço.

-*AnimateRobot*: Vai usar a matriz 5-D que contém todos os movimentos necessários para animar o movimento do robô por todos os passos.

6 Parâmetros iniciais

Ao inicializar o programa, vai ser pedido o número de vezes que irão ser gerados blocos para apanhar e o número de vezes que deseja ver a previsão e o percurso feito do end-effector.

Será lido um ficheiro com o nome tp2.txt; este ficheiro poderá conter valores referentes aos que existem no guião e, adicionalmente, também pode alterar os valores *SideL* e *SideH* que são relevantes às dimensões dos lados das mesas, *DS* relevantes ao espaço entre o tapete e os lados das mesas e *LegC* e *LegL* relevantes às dimensões das pernas das mesas.

7 Diagrama de funcionamento

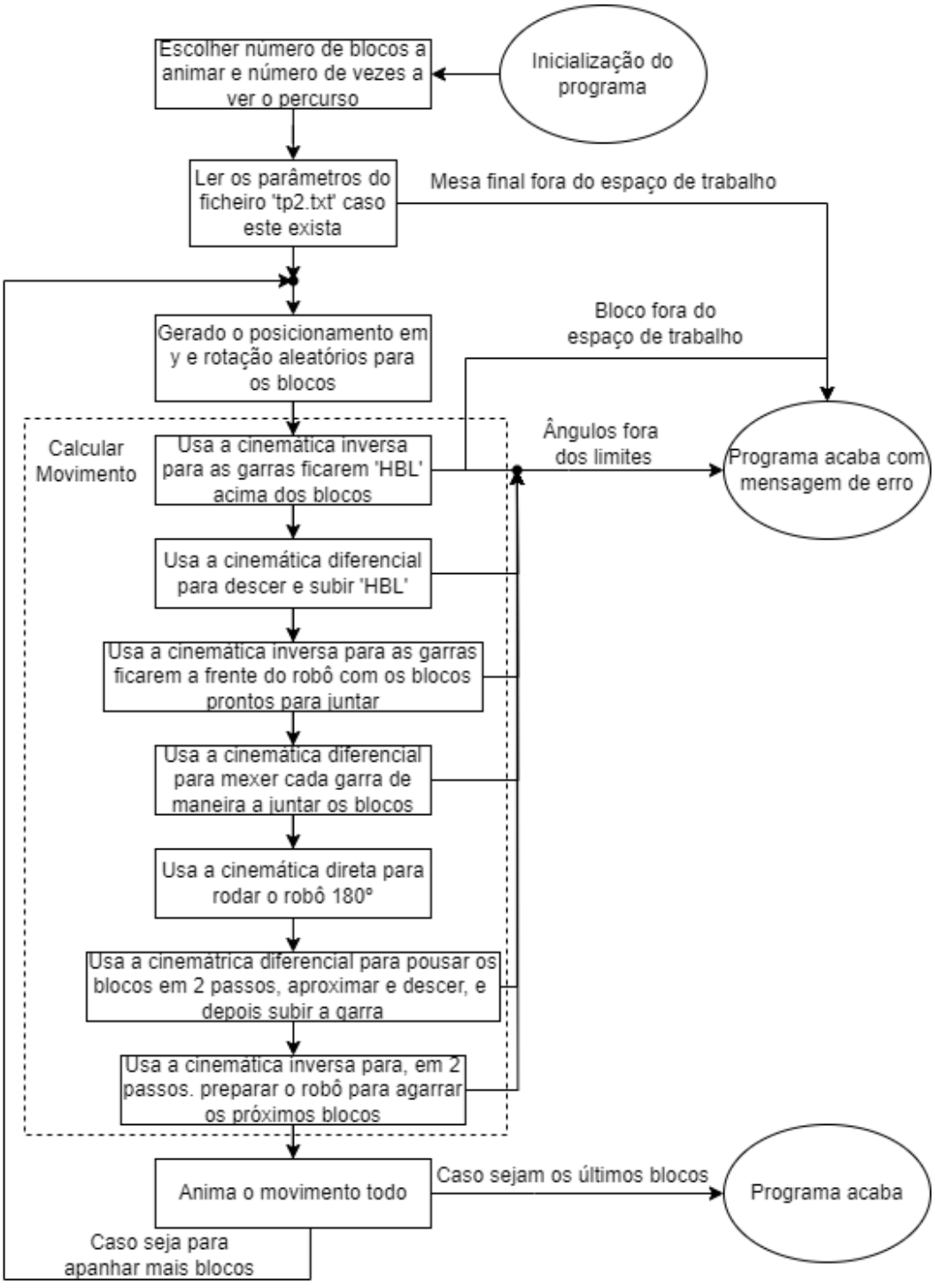

Figure 2: Diagrama de funcionamento geral do programa