Assignment 3

ELEC 442 - Introduction to Robotics

Sondre Myrberg (81113433) Ola Helbaek (68776772)

November 20, 2017

1 Computation of representation

As we have the relationship between the initial frame $\underline{C}_i$ and the final frame $\underline{C}_f$ given as

$$\begin{aligned}\underline{C}_f &= \underline{\underline{R}}\underline{C}_i \\ &= \mathrm{e}^{\theta \underline{\boldsymbol{s}}\times}\underline{C}_i\end{aligned}$$

we could rearrange this as

$$\begin{aligned}\underline{C}_f &= \underline{C}_i R \\ &= \underline{C}_i \mathrm{e}^{\theta \boldsymbol{s}\times}\end{aligned}$$

where

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

as given in the text. This gives us the relationship

$$\mathrm{e}^{\theta \boldsymbol{s}\times} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

which we want to solve for $\boldsymbol{s}$. Firstly use that

$$\begin{aligned}\mathrm{e}^{\theta \boldsymbol{s}\times} &= \sum_{n=0}^{\infty} \frac{(\theta \boldsymbol{s}\times)^n}{n!} \\ &= \boldsymbol{I} + (\theta \boldsymbol{s}\times) + \frac{1}{2!}(\theta \boldsymbol{s}\times)^2 + \dots\end{aligned}$$

and with the property that $(\boldsymbol{s}\times)^3 = -(\boldsymbol{s}\times)$ for a skew-symmetric matrix we get

$$\begin{aligned}\mathrm{e}^{\theta \boldsymbol{s}\times} &= \boldsymbol{I} + \underbrace{\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots\right)}_{=\sin\theta}(\boldsymbol{s}\times) + \underbrace{\left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \dots\right)}_{=1-\cos\theta}(\boldsymbol{s}\times)^2 \\ &= \boldsymbol{I} + \sin\theta(\boldsymbol{s}\times) + (1-\cos\theta)(\boldsymbol{s}\times)^2 = R\end{aligned}$$

With

$$\boldsymbol{s}\times = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}$$

and

$$(\boldsymbol{s}\times)^2 = \begin{bmatrix} -{s_2}^2 - {s_3}^2 & s_1\, s_2 & s_1\, s_3 \\ s_1\, s_2 & -{s_1}^2 - {s_3}^2 & s_2\, s_3 \\ s_1\, s_3 & s_2\, s_3 & -{s_1}^2 - {s_2}^2 \end{bmatrix}$$

we get

$$
\begin{aligned}
\operatorname{Tr}(R) &= \operatorname{Tr}(\boldsymbol{I}) + \cancel{\operatorname{Tr}(\sin\theta(\boldsymbol{s}\times))} + \operatorname{Tr}\big((1-\cos\theta)(\boldsymbol{s}\times)^2\big) \\
\implies r_{11} + r_{22} + r_{33} &= 3 + (1-\cos\theta)\underbrace{(-2s_1^2 - 2s_2^2 - 2s_3^2)}_{=-2} \\
\implies \cos\theta &= \frac{r_{11}+r_{22}+r_{33}-1}{2}
\end{aligned}
$$

and

$$
\begin{aligned}
\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \sin\theta \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix} \\
&+ (1-\cos\theta) \begin{bmatrix} -{s_2}^2 - {s_3}^2 & s_1\, s_2 & s_1\, s_3 \\ s_1\, s_2 & -{s_1}^2 - {s_3}^2 & s_2\, s_3 \\ s_1\, s_3 & s_2\, s_3 & -{s_1}^2 - {s_2}^2 \end{bmatrix} \\
&\implies \begin{cases} r_{21} &= \sin\theta s_3 + (1-\cos\theta)s_1 s_2 \\ r_{12} &= -\sin\theta s_3 + (1-\cos\theta)s_1 s_2 \\ r_{13} &= \sin\theta s_2 + (1-\cos\theta)s_1 s_3 \\ r_{31} &= -\sin\theta s_2 + (1-\cos\theta)s_1 s_3 \\ r_{32} &= \sin\theta s_1 + (1-\cos\theta)s_2 s_3 \\ r_{23} &= -\sin\theta s_1 + (1-\cos\theta)s_2 s_3 \end{cases}
\end{aligned}
$$

By doing subtractions we get

$$
2\sin\theta \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}
$$

and connect this to three equations on the form

$$
\begin{aligned}
\tan\theta = \frac{\sin\theta}{\cos\theta} &= \frac{\frac{r_{32}-r_{23}}{2s_1}}{\frac{r_{11}+r_{22}+r_{33}-1}{2}} = \frac{r_{32}-r_{23}}{s_1(r_{11}+r_{22}+r_{33}-1)} \\
&= \frac{\frac{r_{13}-r_{31}}{2s_2}}{\frac{r_{11}+r_{22}+r_{33}-1}{2}} = \frac{r_{13}-r_{31}}{s_2(r_{11}+r_{22}+r_{33}-1)} \\
&= \frac{\frac{r_{21}-r_{12}}{2s_3}}{\frac{r_{11}+r_{22}+r_{33}-1}{2}} = \frac{r_{21}-r_{12}}{s_3(r_{11}+r_{22}+r_{33}-1)} \\
\implies \boldsymbol{s} &= \underline{\frac{1}{\tan\theta(r_{11}+r_{22}+r_{33}-1)} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}}
\end{aligned}
$$

In the special case of $\sin\theta = 0$ we know that $\theta = n\pi$. For n even we get $e^{\theta\boldsymbol{s}\times}v = v$, and for n odd we get that $e^{\theta\boldsymbol{s}\times}v = -v$. This applies to all $\boldsymbol{s}$ and all $\theta = n\pi$ where $n \in \mathbb{Z}$.

2 Trajectory generation

Using the fuction `forward_kinematics` defined in Listing 1 we get the initial end effector position and orientation with $q(0) = \begin{bmatrix} 0 & 0 & 90 & 0 & 90 & 0 \end{bmatrix}$ given as

$$C_6 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad \underset{\sim}{o_6} = \begin{bmatrix} 471.48 \\ 149.09 \\ 433.07 \end{bmatrix}$$

```
function [k_vectors, origins, J, Q_end] = forward_kinematics(theta, d, a, alpha)
    k_0 = [0;0;1];o_0=[0;0;0]; J = [];
    T = zeros(4,4,6); Q = zeros(3,3,6); k_vectors = zeros(3,1,6);
    origins = zeros(3,1,6);
    T_end = eye(4); Q_end=eye(3);
    for i = 1:6
        [T_temp, Q_temp] = DH_homog(theta(i), d(i), a(i), alpha(i));
        T_end = T_end*T_temp; Q_end = Q_end*Q_temp;
        T(:,:,i) = T_temp; Q(:,:,i) = Q_temp;
        k_vectors(:,:,i) = Q_end(1:3,3);
        origins(:,:,i) = T_end(1:3,4);
    end
    %Generate Jacobian for puma 560
    for i = 1:6
        if i==1
            J = [J, [cross(k_0, origins(:,:,6)-o_0); k_0]];
        else
            J = [J, [cross(k_vectors(:,:,i-1), origins(:,:,6)-origins(:,:,i-1)); k_vectors(:,:,i-1)]];
        end
    end
end
```

Listing 1: MATLAB code to do forward kinematics

Further we should use equation (25) and (26) from the notes, but we couldn't find a way to do this properly stepwise or without using inverse kinematics on wach step. We ended up with circular dependencies all the way, but have sort of a framework for the matlab script, given in Listing 2. So this is how far we got.

```
close all
```

```matlab
clear variables
clc

%% Task 2

theta_init = deg2rad([0 0 90 0 90 0]);
theta_offset = deg2rad([0 0 90 0 0 0]);
d = [0 0 -149.09 433.07 0 60];
a = [0 431.80 20.32 0 0 0];
alpha = deg2rad([-90 180 90 90 -90 0]);

Fs = 50; dt = 1/Fs;

% Initial end effector pose
[k_vectors, origins, J_init, Q_end_init] = forward_kinematics(theta_init + theta_offset, d, a, alpha);
o_end_init = origins(:,:,6);
k_end_init = k_vectors(:,:,6);

% Get desired ende effector poition and pose
o_d = [317;506;673];
j_d = [-0.389; -0.325; 0.862];
k_d = [0.769; 0.401; 0.498];

% Or ask for input
% o_d_in = input('Input desired end effector origin');
% k_d_in = input('Input desired k_d'); k_d = k_d_in/norm(k_d_in);
% j_d_in = input('Input desired j_d');  j_d = j_d_in/norm(j_d_in);
% while k_d'*j_d ~= 0
%     disp('k_d and j_d not orthogonal, try again');
%     j_d_in = input('Input desired j_d');  j_d = j_d_in/norm(j_d_in);
% end
% i_d = cross(j_d, k_d)/norm(cross(j_d, k_d));

% Calculating omegas, theta_dots and Jacobians
o_n = o_end_init; %end effector position
q = theta_init;

for ts = 0:dt:1
    q_prev = q;
```

```
    dq = (J\v_n)*dt; % equivalent to inv(J)*v_n but faster
    q = q_prev + dq;

    q = q_prev + q_dot_prev*dt + q_dotdot_prev*(dt)^2/2;

    o_n_prev = o_n;
    o_n = o_n_prev + o_n_dot;
    v_n = [o_n_dot ; omega_n];
end
```
Listing 2: MATLAB framwork for assignment 3