센서모션로봇실습

calc_base2end_tf

학　과 : 스마트시스템소프트웨어학과

이　름 : 20170404 한종수

제출일 : 2021. 04. 27

1. 이 패키지의 목표 (하고자하는 것)

1.1. dh 파라미터와 일치하는 homogeneous matrix를 구한다. 각 joint 에서의 matrix를 연속적으로 곱해 end-effector 좌표계를 base 좌표계로 변환시키는 homogeneous transform matrix 계산하는 것.

2. /joint_states Topic 의 message type 과 그 안의 자료구조를 제시할 것

```
Header header

string[] name
float64[] position
float64[] velocity
float64[] effort
```

JointState Message

2.1. 왼쪽은 ros wiki에서 sensor_msgs/JointState.msg 파일에 대한 설명을 가져온 것이다. header에는 시간, name은 각 joint의 이름. position(위치, rad or m), velocity(속도, rad/s or m/s), effort(힘, Nm or N)에는 각 joint에 대한 상태를 나타낸다. 그리고 각 이름과 상태는 배열로 관리되어 각 joint에 대한 정보를 인덱싱 하여 따로 받아올 수 있다.

3. final_mat 구하는 과정 설명 (원리) - 자세히 쓸 것 [채점에서 가장 중요시 볼 부분]

```
// get vector value
std::cout << "-------- Joint information ---------"<<std::endl;
std::cout << "shoulder_pan_joint\t" << ptr->position[0] << std::endl;    //shoulder_pan_joint ptr->position[0]
std::cout << "shoulder_lift_joint\t" << ptr->position[1] << std::endl;   //shoulder_lift_joint ptr->position[1]
std::cout << "elbow_joint\t\t" << ptr->position[2] << std::endl;           //elbow_joint ptr->position[2]
std::cout << "wrist_1_joint\t\t\t" << ptr->position[3] << std::endl;       //wrist_1_joint ptr->position[3]
std::cout << "wrist_2_joint\t\t\t" << ptr->position[4] << std::endl;       //wrist_2_joint ptr->position[4]
std::cout << "wrist_3_joint\t\t\t" << ptr->position[5] << std::endl;       //wrist_3_joint ptr->position[5]

//Homogeneous Matrix
Eigen::Matrix4d homo_mat_0, homo_mat_1, homo_mat_2, homo_mat_3, homo_mat_4, homo_mat_5;

homo_mat_0 = RotZ(ptr->position[0]) * TransZ(d_dh[0]) * TransX(a_dh[0]) * RotX(alpha_dh[0]);  //shoulder_pan_joint -> base
homo_mat_1 = RotZ(ptr->position[1]) * TransZ(d_dh[1]) * TransX(a_dh[1]) * RotX(alpha_dh[1]);  //shoulder_lift_joint -> shoulder_pan_joint
homo_mat_2 = RotZ(ptr->position[2]) * TransZ(d_dh[2]) * TransX(a_dh[2]) * RotX(alpha_dh[2]);  //elbow_joint -> shoulder_lift_joint
homo_mat_3 = RotZ(ptr->position[3]) * TransZ(d_dh[3]) * TransX(a_dh[3]) * RotX(alpha_dh[3]);  //wrist_1_joint -> elbow_joint
homo_mat_4 = RotZ(ptr->position[4]) * TransZ(d_dh[4]) * TransX(a_dh[4]) * RotX(alpha_dh[4]);  //wrist_2_joint -> wrist_1_joint
homo_mat_5 = RotZ(ptr->position[5]) * TransZ(d_dh[5]) * TransX(a_dh[5]) * RotX(alpha_dh[5]);  //wrist_3_joint -> wrist_2_joint

final_mat = homo_mat_0 * homo_mat_1 * homo_mat_2 * homo_mat_3 * homo_mat_4 * homo_mat_5;                  //wrist_3_joint -> base
```

3.1. end-effector 좌표계를 base 좌표계로 변환시키려면 각 joint에서 지금 좌표계에서 이전 좌표계로 변환하는 Homogeneous 행렬들을 구해야한다. 각 joint 간에 Homogeneous 행렬들을 homo_mat_i라고 했을때, 그 행렬은 RotZ(ptr->position[i])TransZ(d_dh[i])TransX(a_dh[i])RotX(alpha_dh[i]) 연산을 차례로 수행함으로써 구할 수 있다. homo_mat_0는 뒤에 shoulder_pan_joint 상의 좌표를 곱했을 때, shoulder_pan_joint에서 base로 변환시키는 역할을 하는 Matrix가 되고, homo_mat_1은 shoulder_lift_joint에서 shoulder_pan_joint로 변환시키는 역할을 하는 Matrix가 되고 이후 homo_mat_i는 마찬가지로 위의 코드의 주석대로 변환시키는 역할을 한다. 그 행렬들을 차례로 homo_mat_0~5 순서로 곱하게 되면, 각 변환 Matrix가 연속적인 좌표계 변환 연산을 하며 wrist_3_joint에서의 좌표계가 wrist_2_joint에서의 좌표계로, 그 좌표계가 또 wrist_1_joint 좌표계로, 변환되면서 결국 마지막에 base 좌표계로 변환되게 되고 이는 final_mat에 저장된다. 이로써 end-effector 좌표계를 base 좌표계로 변환시키는 homogeneous transform matrix를 구할 수 있다.

4. end_effector_origin 과 transformed_origin_on_base 이 각각 의미하는 바

4.1.end_effector_origin은 end_effector의 좌표계 상(wrist_3_joint)에서의 원점으로 (0, 0, 0, 1) 이다. 끝의 1은 Homogeneous Matrix에 곱할때 필요한 값으로 x, y, z에 해당하는 값은 아니다.

4.2.transformed_origin_on_base는 end_effector의 좌표계 원점을 base 좌표계로 변환시켰을 때 좌표값으로, base 좌표계에서 본 end-effector 좌표계의 원점이다.

5. 자신이 생각한 6 주차, 7주차 이해도 (0 ~ 100 %)

5.1. 80%인 것 같습니다.