Study of forward and inverse kinematics and test of ROS implementation of PSM arm in dVRK

Robotics Project
Master's degree in Computer Science and Engineering
University of Verona
Luca Arietti VR439211

***Abstract*—This document concerns the study of the forward and inverse kinematics of the PSM arm present in the da Vinci Research Kit (dVRK). Moreover, it deals with the test of the ROS implementation of the PSM created by John Hopkins University [1] contained in the dVRK and an implementation of the inverse kinematic in closed form. The test was done by installing the dVRK tool and using RViz to see how the robot moves. Furthermore, a Python script has been developed, which exploits the API made available by the dVRK, which allows information on the position of the end effector and the variables of the joints, to move the joints and therefore to have the final position of the EE (forward kinematic) and enter the final position of the EE and obtain the configuration of the robot joints (inverse kinematic).**

I. Robot structure

The psm arm has 7 degrees of freedom and moves a surgical instrument around a fixed fulcrum point, that acts as an invariant respect to the PSM joint configuration. This point is called Remote Center of Motion (RCM). The last degree of freedom - the seventh -, corresponding to the opening and closing motion of the gripper, is not considered because the focus is in computing the position and orientation of the center of the end effector, without considering the opening functionality of the gripper. This decision was also taken by those who developed the API of the dVRK, in fact, as you will see in the Python script, only 6 joints are requested and provided. These six degrees of freedom are combined in a RRPRRR sequence, where R correspond to Revolute joint and P correspond to Prismatic joint.

Referring to the figure 1, these six degrees of freedom are:

- all the arm rotate about axis J_1 of an angle ϑ_1
- the two parallelograms (in black in the figure) constitute a mechanism that allows rotation of the surgical instrument about axis J_2 of an angle ϑ_2
- the variable length d_3 allows the surgical instrument to translate along the J_3 axis
- the surgical instrument can also rotate of an angle ϑ_4 about axis J_4
- the surgical instrument rotate about axis J_5 of an angle ϑ_5
- the surgical instrument rotate about axis J_6 of an angle ϑ_6

The axes J_1, J_2, J_3, J_4 collide at one point: RCM. The position of the RCM does not depend on the variables of the joints.

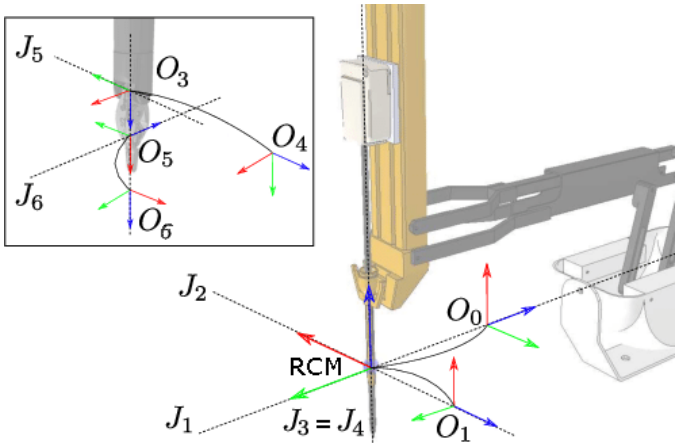

Figure 1: Patient Side Manipulator (PSM) kinematic description (partially taken from [2])

Donald L. Pieper [3] demonstrated that a sufficient condition for the kinematic structure of a six-degree-of-freedom manipulator to be resolvable is that the axes of three consecutive rotational joints intersect at a single point for all manipulator configurations. When the axes of the last three joints intersect at a single point, the last three joints are said to identify a spherical wrist. In our case, the joints on the axes J_4, J_5, J_6 are orthogonal, but they constitute a non-spherical wrist, because they don't intersect at the same point, as can be seen from the figure.

Given a certain pose of the End Effector, in general, not all positions of the joints are possible; in fact there could be configurations that require one or more joints to assume values that are outside the joint limits. So, this is an analysis of the possible admissible values of the joints (angles are in degree, except the third joint which is written in meters):

- Joint 1: -70 / +70
- Joint 2: -50 / +50
- Joint 3: 0 / +0,235
- Joint 4: -175 / +175
- Joint 5: -175 / +175
- Joint 6: -175 / +175

II. Forward kinematic

The forward kinematic has as its target to define the end organ pose in function of the values assumed by the joint

variables. The world triad W coincides with the triad 0 if the robot has the first two joint variables at 0. For this reason, the corresponding row W-0 was not put in the DH table as it is trivial. Below, the DH table:

i	d	ϑ	a	α
0 - 1	0	ϑ_1*	0	$-\frac{\pi}{2}$
1 - 2	0	ϑ_2*	L_2	0
2 - 2'	0	ϑ_2'* + $\frac{\pi}{2}$	L_2'	0
2' - 2"	0	ϑ_2"*	0	$-\frac{\pi}{2}$
2" - 3	d_3* + L_2"	0	0	0
3 - 4	L_3	ϑ_4 *	0	$+\frac{\pi}{2}$
4 - 5	0	ϑ_5 * + $\frac{\pi}{2}$	L_4	$-\frac{\pi}{2}$
5 - 6	0	ϑ_6 * - $\frac{\pi}{2}$	0	$-\frac{\pi}{2}$

Table 1: DH parameters

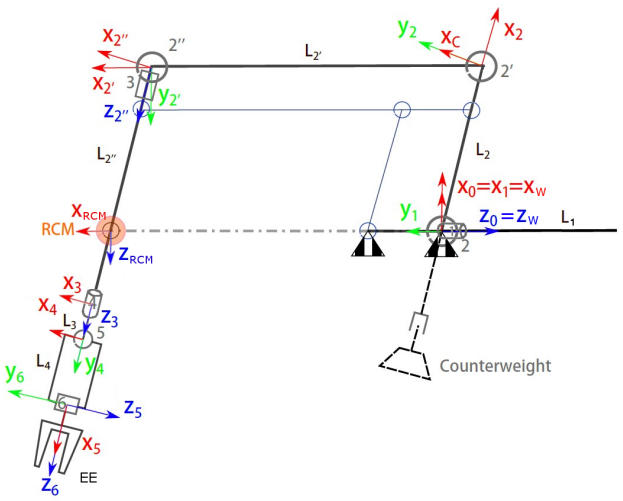

Figure 2: DH frames of PSM (partially taken from [4])

Figure 2 shows the frames used to define the parameters of the DH table. The notation used for the DH table provides that the joint variable is Z, that is the rotations and translations occur along the Z axis. Counterclockwise rotations are positive, while clockwise rotations are negative. Moreover, in order to pass from one reference system to the other, translation occurs first, and then occour rotation. Therefore, denoting with letter d the translation along the Z axis, ϑ the rotation along the Z axis, letter a the translation along the X axis, α the rotation along the X axis, the order used is: $d \rightarrow \vartheta \rightarrow a \rightarrow \alpha$

In the case of rotational joints, the joint variable is the ϑ, while in the case of prismatic joints, the joint variable is d. Be careful that for better readability, these letters have been indicated in the DH table followed by the asterisk symbol, just to indicate that they are free variables. Although there are free variables, I have preferred to indicate in any case also the rotations along the axes, as done by the teachers. For simplicity, in indicating the angles of rotation between one system and another, the robot was considered with the links L_1, L_2, L_2', L_2" perpendicular to each other.

Each frame was indicated above at a joint. Note that there are "two more" joints in the DH table. They show how the ϑ_2 angle is maintained between frame 1 and RCM point and frame 6, thanks to the structure made up of parallelograms.

A. *Explanation of DH parameters*

With reference to figure 2 and table 1, below is an exhaustive explanation of each parameter, collected for each step. Remember that the robot was considered with the links L_1, L_2, L_2', L_2" perpendicular to each other. Note that the reference systems have the number that identifies them one less than the number of the joint in which they are located. Therefore, the reference system number 6 corresponds to the end effector.

- [0 - 1] The Z axis corresponds to the rotational joint 1, this means that the joint can rotate around the Z axis with an angle ϑ_1. To bring this reference system to the next frame, that is to the rotational joint 2, it is necessary to simply rotate along the X axis of - $\frac{\pi}{2}$ (i.e. go to make a clockwise rotation along the X axis). The joint 1 is considered to coincide with joint 2, therefore no translations are required. In this way the Z axis is now oriented along the rotational joint 2.
- [1 - 2] The Z axis corresponds to the rotational joint 2, this means that the joint can rotate around the Z axis with an angle ϑ_2. To bring this frame to the next, that is the "fictitious" rotational joint 2', it is necessary to translate along the X axis by a length equal to L_2. We now have the Z axis oriented correctly to allow joint 2' to rotate along Z by a ϑ_2' angle.
- [2 - 2'] The ϑ_2' angle is the free variable of joint 2'. Now, to bring this frame to the next, rotate along the Z axis by an angle + $\frac{\pi}{2}$ (counterclockwise rotation) and then translate along the X axis by a length equal to L_2'. The new frame will have the Z axis on the fictitious joint 2".
- [2' - 2"] The ϑ_2" angle is the free variable of joint 2". To pass from this frame to the next, it's only necessary to perform a rotation along the X axis of - $\frac{\pi}{2}$. This is because the prismatic joint (J_3) was considered in conjunction with joint 2". The Z axis of the reference system 2" is now oriented correctly to make the prismatic joint (J_3) move along Z.
- [2" - 3] The prismatic joint J_3 can move along the Z axis by a free variable d_3. To move to the next frame, the length of the link L_2" must be added (always along the Z axis). Now we are on the reference system number 3, which has the Z axis correctly oriented on the rotational joint 4.
- [3 - 4] The rotational joint 4 can rotate around the Z axis by an angle ϑ_4. To move from this frame to the next, is first necessary translate along Z by a length equal to link

L_3, then rotate along the X axis by an angle + $\frac{\pi}{2}$. Now joint 5 can rotate along the Z axis.

[4 - 5] The joint 5 can rotate along the Z axis with its free variable ϑ_5. To go from reference system 4 to 5, rotate along Z by + $\frac{\pi}{2}$, then translate the length of the link L_4 along X and finally rotate along X by - $\frac{\pi}{2}$. We now have the Z axis on the rotational joint 6.

[5 - 6] The joint 6 can move with an angle ϑ_6 around Z. To position the triad on the end effector in such a way that the Z axis is outgoing, it is necessary to rotate first around the Z axis of - $\frac{\pi}{2}$ and finally rotate around the X axis of - $\frac{\pi}{2}$. Hence, we have the Z axis out of the end effector and the X axis "out of the sheet".

III. Inverse kinematic

The inverse kinematic problem concerns the determination of the joint variables, once the position of the end effector has been assigned. The closed-form determination of inverse kinematics requires geometric intuitions for the identification of those significant points on the structure with respect to which to express positions and / or orientations in function to a small number of joint variables, or requires algebraic intuitions for the identification of the significant equations containing the unknowns to be manipulated.

My solution of closed-form inverse kinematics uses a geometric approach. The rotation of the three end joints (near the end effector) does not affect the spatial position of the surgical instrument, therefore these three joints are neglected in the inverse kinematics calculation. Therefore, it was possible to calculate the joint variables of the first three joints (two rotational and one prismatic) using the desired position in the space of the end effector as information. To facilitate the calculation, the RCM point was used, that is characterized by the fact that it never changes in any robot configuration, that is, it always remains at a distance L_2' from the initial triad W (see image 2). Always referring to the image 2, note that the triad in RCM is: X axis lying on the axis of link L_1 (and therefore on the Z_W axis but with the opposite direction), Z axis is on the same plane as X_W axis and it's parallel to it but with the opposite direction, Y axis derived with the right hand rule, that is entering the sheet. The position of the end effector given in input, and referred to the W reference system, must be expressed according to the RCM triad. So, it's necessary to convert these coordinates from the W reference system to the RCM reference system. This is possible by doing these equations (each cell on the left is equivalent to the one on the right in the same row):

$Triad_{RCM}$	$Triad_W$
+ X	- Z - L_2'
+ Y	- Y
+ Z	- X

Table 2: Conversion of coordinates from the W reference system to RCM and vice versa

Furthermore, to be more explanatory, in table 3 there are the DH parameters to bring the W reference system to the RCM reference system (also see image 2). Note that is necessary take two steps as it is not possible to rotate or translate along the Y axis.

d	ϑ	a	α
- L_2'	0	0	+ $\frac{\pi}{2}$
0	- $\frac{\pi}{2}$	0	+ $\frac{\pi}{2}$

Table 3: DH parameters from the W reference system to the RCM reference system

The joint variables of the first three joints can be calculated as follows (please see images number 3, 4, 5). Note that with X, Y, Z letters is indicated the coordinate of the end effector point, referred to RCM reference system.

$$d_3 = \sqrt{X^2 + Y^2 + Z^2}$$

$$\theta_1 = \arctan(\frac{Y}{Z})$$

$$\theta_2 = \arcsin(\frac{X}{d_3})$$

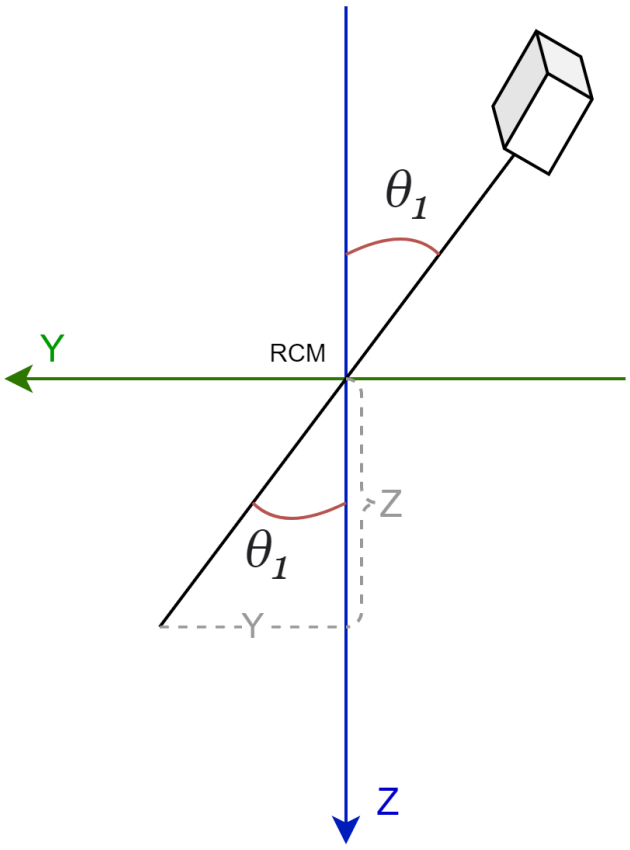

Figure 3: Diagram of the angle ϑ_1

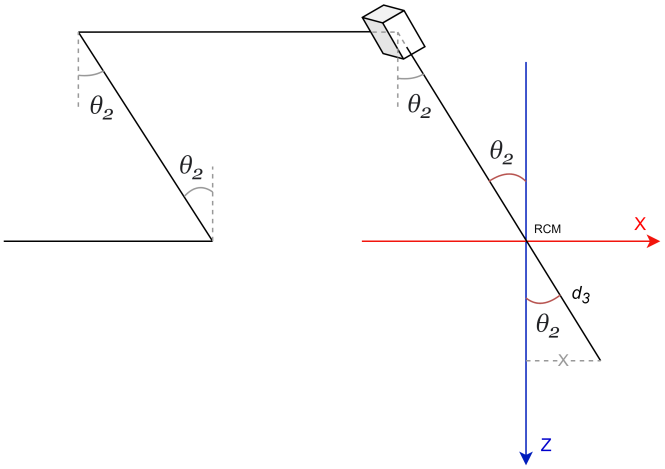

Figure 4: Diagram of the angle ϑ_2

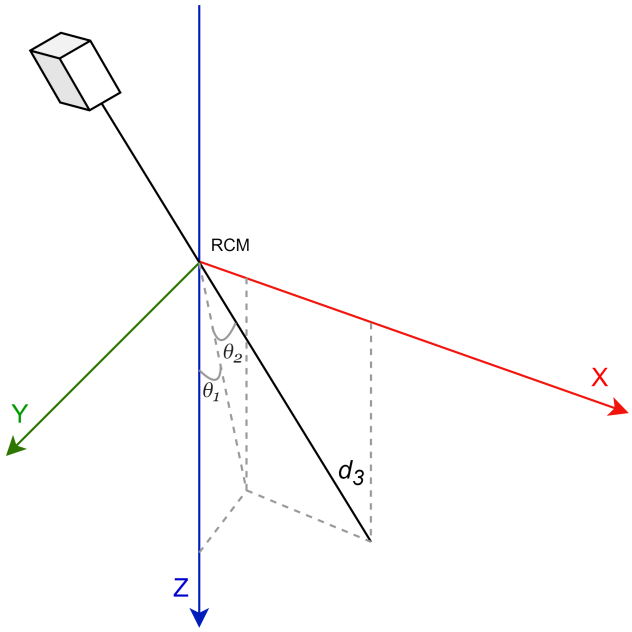

Figure 5: Diagram of the length d_3 (prismatic joint)

IV. Simulation

In the da VinciResearch Kit (dVRK), created by John Hopkins University, there is the ROS implementation of the PSM [1]. This implementation publishes robot status on various ROS topics. Using RViz to view the robot and the official API that are made available by the dVRK, it was possible to create a python script that interfaces with the robot, reads its information and gives it commands. This python script is available here [5].

In practice, the created python script allows you to:

- Get informations about the position of the end effector and the variables of the joints
- Move only a joint and consequently have the final position of the EE (forward kinematic)
- Move several joints at the same time and consequently have the final position of the EE (forward kinematic)
- Enter the final position of the EE that you want to achieve and obtain the configuration of the robot joints (inverse kinematic) using the KDL tool - an approximate solution
- Enter the final position of the EE that you want to achieve and obtain the configuration of the robot joints (inverse kinematic) using the closed form illustrated above in the chapter III

The official API uses KDL to solve inverse kinematics in open form, in python it's possible to use KDL through the pyKDL library. KDL [6] is a library (Kinematics and Dynamics Library) that develops an application independent framework for modelling and computation of kinematic chains, such as robots, biomechanical human models, computer-animated figures, machine tools, etc. It provides class libraries for geometrical objects (point, frame, line,...), kinematic chains of various families (serial, humanoid, parallel, mobile,...), and their motion specification and interpolation. As you can read on the official website: "Recursive forward kinematic solvers: for now we only have one generic forward and velocity position kinematics solver. It recursively adds the poses/velocity of the successive segments, going from the first to the last segment" [7].

An exhaustive video demonstration of the python script can be found here [8]. Observing it in execution, one notices in particular how the two solutions concerning inverse kinematics give rather similar results. This confirms the correctness of the solution proposed in closed form, since it should be emphasized that the discrepancies present are due to approximations in the calculations, for example in the use of *arcsin* to calculate ϑ_2: if there are small angles, large approximations occur.

Furthermore, the API used reported the coordinates differently than my work, which reports them based on RCM. In particular, these were the equalities between the axes used in this work and those used by the API:

$\text{Triad}_{\text{RCM}}$	$\text{Triad}_{\text{API}}$
+ X	+ Y
+ Y	+ X
+ Z	0.0065 - Z

Table 4: Conversion of coordinates from the API reference system to RCM and vice versa

It can be seen that the API's reference point is displaced along the Z axis by 0.0065 meters below the RCM. The script shows both reference systems when it is necessary to show the position of the end effector and when it is entered to calculate the inverse kinematics.

Furthermore, the ϑ_2 angle considered by the API has the opposite sign to that in this work (in my work, ϑ_2 has a positive sign when it is as shown in the figure 4).

Testing the implementation in ROS, it was possible to notice how the ros nodes varied when the python script was running.

Using the `rosrun rqt_graph rqt_graph` command it was possible to create images 6 and 7 that show the dVRK tool running with and without the simultaneous execution of the script in python.

REFERENCES

[1] J. H. University. sawintuitiveresearchkit. [Online]. Available: https://github.com/jhu-dvrk/sawIntuitiveResearchKit/wiki

[2] M. Selvaggio. Psm kinematic description. [Online]. Available: https://www.researchgate.net/figure/PSM-kinematic-description_fig3_328241499

[3] D. L. Pieper, "The kinematics of manipulators under computer control," Ph.D. dissertation, Stanford University, 1968.

[4] L. V. B. S. G. A. Fontanelli, F. Ficuciello, "Modelling and identification of the da vinci research kit robotic arms."

[5] L. Arietti. Robotics project about the psm arm of the dvrk. [Online]. Available: https://github.com/LucaArietti/robotics_project_dvrk_psm

[6] O. Kinematics and Dynamics. Kdl wiki. [Online]. Available: https://www.orocos.org/wiki/orocos/kdl-wiki.html

[7] ——. Kinematic and dynamic solvers. [Online]. Available: https://www.orocos.org/wiki/Kinematic_and_Dynamic_Solvers.html

[8] L. Arietti. Robotics project - psm arm of dvrk. [Online]. Available: https://www.youtube.com/watch?v=V8fZ8WUCJoM

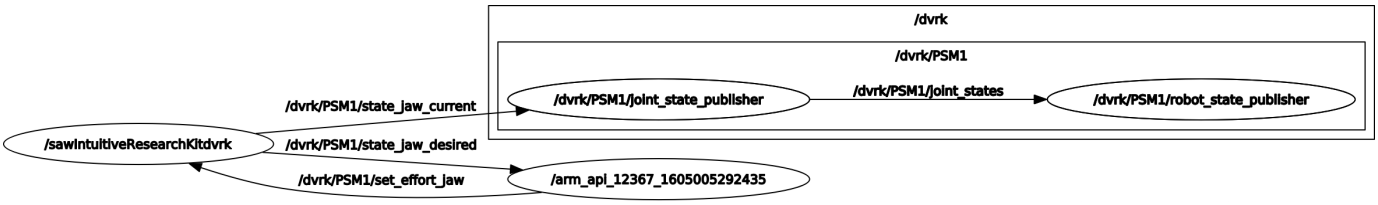

Figure 6: Rosgraph while the python script is running

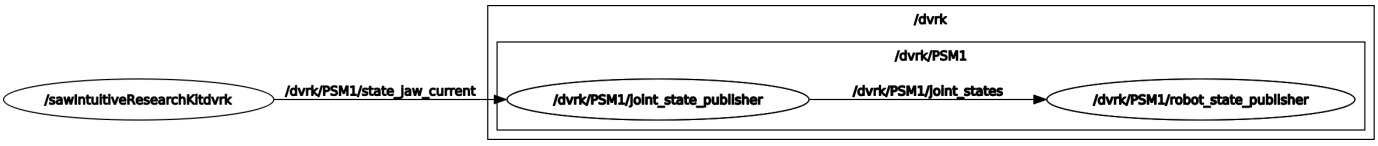

Figure 7: Rosgraph without running the python script