Forward kinematics, closed-form Inverse kinematics analysis, and trajectory generation for dVRK patient side manipulator (PSM)

Chris Eubel† , Ali Asghari Adib† , Md Ferdous Alam†

***Abstract*— In this project we are aiming to leverage the class materials on forward kinematics, inverse kinematics and trajectory generation and apply them to the dVRK robot. The dVRK utilizes a double paralellogram in its design that enables the robot to maintain the remote center of motion (RCM) mechanically, at the point of entering the patient's body. This unique design makes the inverse kinematics problem challenging. The first aim of this project is to solve for the forward and closed-form inverse kinematics of the dVRK. The forward kinematics solution is compared with the built in forward kinematics in an incremental change of a joint angle θ and the error in end effector position is reported. The inverse kinematics is also validated in simulation. The second aim of this project is to write a path planning algorithm (trajectory generation) for the robot to follow a parameterized path of The Ohio State University logo (the Block "O") with the end effector maintaining an orientation square to the Block "O" plane and parallel to the velocity of each path segment. The algorithm is then tested in simulation and on the physical dVRK and the results are reported.**

I. INTRODUCTION

The daVinci Research Kit (dVRK) is an effort to facilitate robotics research in the area of telerobotic surgery [1]. The kit is composed of a patient side manipulator (PSM) and a master tool manipulator (MTM). The PSM is a 7 degrees of freedom (DOF) robot, demonstrated in Fig. 1. As shown, joint one and two are revolute joints and followed by a prismatic joint. The interchangeable tool is driven by the prismatic joint that will be inserted into the patient body. The interchangeable tool has three revolute joints associated with it, shown in Fig. 1 inset. There is also a degree of freedom associated with the opening and closing of the jaw. The dVRK uses a double parallelogram in its design which enables it to mechanically hold a fix position at the point of insertion into the patient body, known as the remote center of motion (RCM). In this project, we are aiming to solve the forward and inverse kinematics of the PSM using the product of exponential (PoE) method. In addition, a path planning scheme for following the Ohio State University logo (the Block "O") outline is developed. The rest of this report is structured as follow. In section II, the methods used to solve the forward kinematics problem are presented. In section III, the closed-form inverse kinematics methods are discussed. In section IV, the path planning and trajectory generation is defined. With underlying principles problem addressed, the software implementation will then be discussed, followed by our results in section VI and conclusion in section VII.

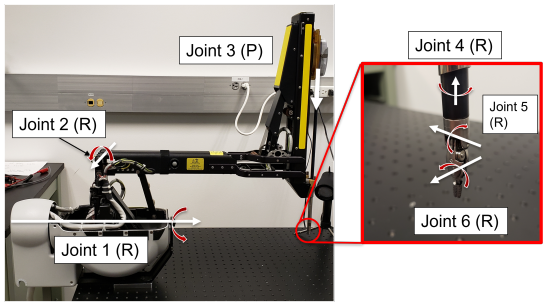

Fig. 1. The dVRK PSM and the joint configuration.

II. FORWARD KINEMATICS

We are neglecting the 7th DOF here, which is associated with the opening and closing of the forceps and does not contribute to the position and orientation if the end effector. Therefore, we are dealing with a 6 degree of freedom manipulator which has an RRPRRR set up. For the forward kinematics (FK), a base frame was placed at the remote center of motion (RCM) with respect to which the position and orientation of the other joints were calculated. To use the product of exponential method we first define our frames and their screw axes. Given the null position orientations in Fig. 2, the table of screw axes $S_i = (\omega_{si}, v_{si})$ can be formed (Table I). It is worth noting that the orientation of the frames was tried to be same as the orientation of the frames in

† Authors contributed equally.
A. Asghari Adib is with the Department of Mechanical Engineering, Ohio State University, Columbus, OH 43210, USA asghariadib.1 at osu.edu
Md. Ferdous Alam is with the Department of Mechanical Engineering, Ohio State University, Columbus, OH 43210, USA alam.92 at osu.edu
Chris Eubel is with the Department of Mechanical Engineering, Ohio State University, Columbus, OH 43210, USA eubel.1 at osu.edu
*project website

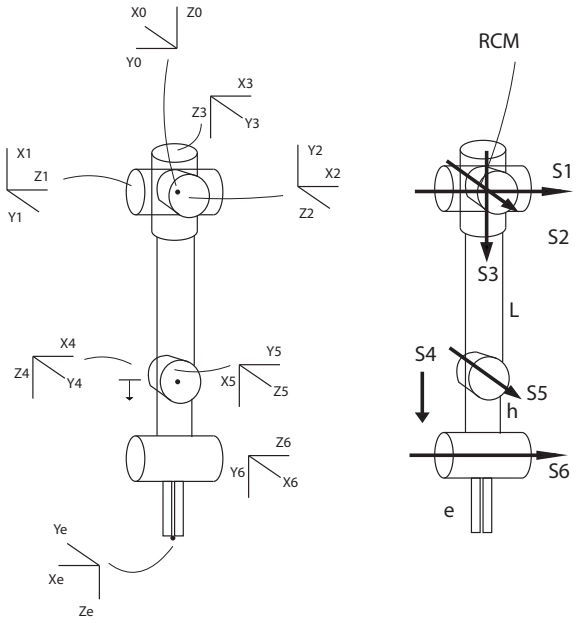

Fig. 2. Frames, joints, and screw axes schematic for the forward kinematics solution. Joints 1, 2 and 3 are coinciding at the base. Distance from base to joint 5 (or 3) is L, from joint 5 to joint 6 is h and from joint 6 to the end effector is e.

S_i	ω_{si}	v_{si}
1	(0 -1 0)	(0 0 0)
2	(-1 0 0)	(0 0 0)
3	(0 0 -1)	(0 0 0)
4	(0 0 0)	(0 0 -1)
5	(-1 0 0)	(0 L 0)
6	(0 -1 0)	(-(h+L) 0 0)

TABLE I

SCREW AXES WITH RESPECT TO THE SPACE FRAME.

the dVRK document so that comparison can be easily made. The only difference is defining Joint 3 and joint 4 in a slightly different manner and with an offset in location which was compensated for in the code so that the results can be validated with the actual system's FK. Knowing the screw axes, the matrix exponentials of the form $e^{[S_i]\theta_i}$ can be formed and the forward kinematics transformation matrix of the end effector frame in the base frame (RCM) can be calculated as:

$$T_{0e} = e^{[S_1]\theta_1} e^{[S_2]\theta_2} ... e^{[S_6]\theta_6} M \tag{1}$$

Once the homogeneous transformation was obtained, the result was validated with the actual system.

III. INVERSE KINEMATICS

For the inverse kinematics (IK), the first step was to put an imaginary base frame at the end effector location. This way, a virtual base is created, and the RCM joints will act as a spherical joint end effector. This will enable us to solve for the inverse kinematics using the PoE method. For the sake of simplicity, the orientation of all frames for this inverse kinematics problem was at the same orientation at the null position of robot.

$$T_{known} = M e^{[B_1]\theta_1} e^{[B_2]\theta_2} ... e^{[B_6]\theta_6} \tag{2}$$

By inverting M and body axis matrix exponentials we can derive equations that can be solved for the theta values. Once we get the solutions for the joint angels, it is important to note that this is an inverse kinematics code working from a virtual fixed base frame at the end effector, and is moving the RCM. In order to find the joint angles to get to a desired end effector position and orientation from the actual base frame at the RCM, we need to plug in ${T_{0e}}^{-1}$ or T_{e0} to the code.

IV. TRAJECTORY GENERATION OVERVIEW

The path planning and trajectory generation task will consist of tracing the outline of a Block "O" figure, within the dexterous workspace of the PSM. The path of the Block "O"s outline will be encoded by the Cartesian coordinates of its eight corners, with a line connecting adjacent coordinates. The end effector will trace the outline from a fixed initial coordinate, ending where it began. The Block "O"s geometry was measured such that the lowest, horizontal segment was 0.1 m, with the final geometric configuration being parameterized by three operations - scaling, reorientation, and origin offset. The scaling is defined by a positive scalar that simply scales the coordinates' relative location smaller or larger. The reorientation is defined by a rotation matrix in SO(3) representing the orientation of the Block "O"'s frame in 3D space, defined with respect to the base frame of the dVRK. The 3x3 identity matrix represents no reorientation and thus the Block "O" is parallel to the surface of the table and algined with the base frame. Finally, the origin offset is defined by a 3D-vector representing the coordiante distance of the Block "O" origin from the base frame origin. Together, these three parameters can define any Block "O" size, orientation, and location in the dVRK's dextrous workspace. This modulation operation can be seen below, where $\{w_0, w_1, ..., w_7\}$ is the original set of coordinates, c is the scaling factor, R is the rotation transformation action on all coordinates, and (o_x, o_y, o_z) is the origin offset.

$$\{w_1\prime, w_2\prime, ..., w_7\prime\} = c{*}R[\{w_0, w_1, ..., w_7\}]{+}(o_x, o_y, o_z)$$

Now, for any desired modulation of the Block "O"s coordinates, the end effector's orientation along the path must still be addressed. The path has been defined such that joint 6's axis will always be parallel to the

linear velocity of each segment, and joint 5's axis will remain perpendicular to the Block "O"s surface for the entirety of the path. These constraints result in a path where a straight translation is generated between each segment, with no rotation, and then a pure rotation at each corner to align the end effector with the next segment's direction. This means there are seven total pure rotation movements and eight total pure translation movements, constantly alternating between. There is only one less rotation movement than translation, as the first/final coordinate does not have a specified rotation operation.

Defining the orientations along the path requires a bit of vector math. To define the normal vector of the Block "O", for any orientation, we take the cross product of the first coordinate with the second coordinate, excluding the origin offset parameter. Normalizing it gives us a unit vector, which helps us satisfy the constraints of SO(3). This remains constant for the entire path, and is shown below. A visual representation is also shown in Fig. 3, below.

$$\hat{r}_n = \frac{w_o \times w_1}{||w_o \times w_1||}$$

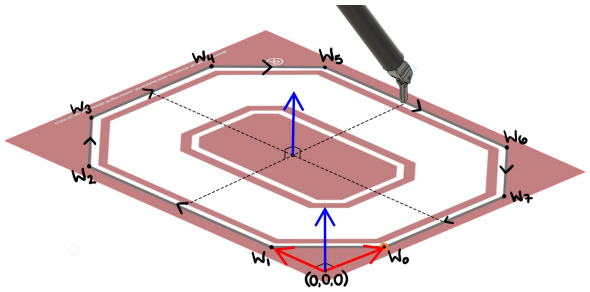

Fig. 3. Vector normal to Block "O" surface.

Now, unlike the normal vector which remains constant for the entire path, we must compute the vector parallel to the translational velocity for each travel segment. This is done by subtracting the end coordinate from the star coordinate of a given travel segment, and normalizing the result. This can be seen below, as $\hat{r}_p^{i,i+1}$ where w_i is the starting coordinate and w_{i+1} is the final coordinate of the current straight segment. This can also be seen visually in Fig. 4

$$\hat{r}_p^{i,i+1} = \frac{w_i - w_{i+1}}{||w_i - w_{i+1}||}$$

The last unit vector is simply the cross product of the normal and parallel component with the proper sign, completing a right-handed frame. Finally, these three unit vectors are packaged into the rotational matrix of each transformation, representing the desired end

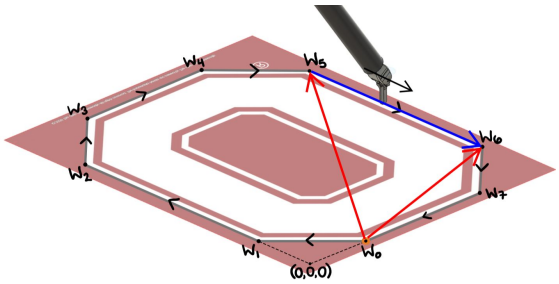

Fig. 4. Vector parallel to a Block "O" path segment.

effector orientation. This rotation matrix referenced from the Block "O" frame, as seen below, must then be transformed to the base frame's perspective before being communicated to the dVRK.

$$R^{i,i+1} = \begin{bmatrix} \hat{-r}_p^{i,i+1} & -(\hat{r}_p^{i,i+1} \times \hat{r}_n) & -\hat{r}_n \end{bmatrix}$$

With all the rotational and positional components defined from the base frame, we can compute the end effector's desired homogeneous transformation matrices, in SE(3), for every path segment's beginning and end. We will call these waypoints, $T^{k,j} = (R^{k,k+1}, w_j)$. It should be noted that each segments' end is the beginning of the next segment, except at the final waypoint of the Block "O" path. Since we have eight pure translation segments, with a pure rotation in between at each corner, we must define an ordered list of 16 transformation matrices, $T = [T^{0,0}, T^{0,1}, T^{1,1}, T^{1,2}, ..., T^{7,7}, T^{7,8}]$. This list of transformations will then be used to compute time-scaled trajectories between each adjacent pair.

With the entire path fully defined, the next, and final, step is to provide a time-scaling, $s(t)$ and its straight-line interpolation, $T^{k,j}(s)$, in the task space. A trapezoidal time-scaling, or linear segments with parabolic blends (LSPB), was utilized for each translation and rotation segment. This scaling consists of an up-ramp of constant acceleration a, a plateau of constant velocity v, and then a down-ramp of constant deceleration $-a$ until stopping after time, T_i.

$$T_i = \frac{a + v^2}{va}$$

Two pairs of velocity and acceleration of the trapezoidal profile were selected for the pure rotation and pure translation segments. The pure translation segments were set to $v = 0.005\ ms^{-1}$, $0.01\ ms^{-2}$, and the pure rotation segments were set to $v = \pi/12\ rad * s^{-1}$, $\pi/6\ rad * s^{-2}$. These parameters were chosen with caution, but it was seen that faster speeds were also well managed by the dVRK's control. Whichever speeds

and accelerations chosen, it should be noted that the condition $v^2/a < 1$ must be respected to produce a trapezoidal profile.

To provide a properly timed trajectory of the physical dVRK's motion, the trapezoidal trajectory was sampled every 5 ms to generate a $s(t)$ value. This 5 ms sampling rate is aligned with the dVRK's 200 Hz communication rate. With $s(t)$ in hand, the proceeding step is to generate the intermediate homogeneous transformation between any two adjacent waypoints via straight-line interpolation. Since the pure translations and pure rotations are defined in the Cartesian space, $T^{k,j} = (R^{k,k+1}, w_j)$, as discussed above, straight-line interpolation between any two adjacent transformations will be achieved by decoupling the rotational motion from the translational motion, as seen below. Where exp is the matrix exponential, defined by Rodrigues' formula, and log is the matrix logarithm of rotation.

$$w(s) = w_j + s(w_{j+1} - w_j)$$
$$R(s) = R^{k,k+1} exp(log(R^{k,k+1T} R^{k-1,k})s)$$

Compiling the geometric path definition, straight-line interpolation, and trapezoidal time-scaling encompasses the entirety of the Block "O" trajectory generation problem. For validation of this process and the resulting trajectory, kinematic simulation in RViz was utilized. Once results in simulation were satisfactory, the same trajectory was implemented on the physical dVRK. It should be noted that the system's already-included inverse kinematics were used, not our own. To test the accuracy of the generated path, a physical Block "O" was modeled and 3D printed. The model can be seen in Fig. 5, with a deep channel as the path to follow. The idea was to distance the end effector from the Block "O"s surface, place a small piece of paper in the forceps, and then have the paper run through the seem of the Block "O". The piece of paper would act as a more precise reference for accuracy, and allow the end effector to keep a safe distance from any obstructions, such as the Block "O" itself or the table surface. However, as will be discussed in the Results section, this plan was not carried out as intended.

V. Implementation on Physical Robot

We implemented the path planning algorithm on a physical dVRK robot. The robot is known as classic dVRK which is the retired version of the first generation surgical dVRK robot. High-level control is used to communicate with the robot, which is a form of abstraction on top of the sophisticated software stack of dVRK robot. Here, our objective is to develop a high-level understanding of how dVRK software and hardware communicate with each other and how the trajectory can be implemented on the physical robot. Note that we only discuss about the PSM arm as we used only this arm for implementation purpose. The full details of the robot software and hardware is available on the official dVRK website [2]. Most of the discussion is based on [3].

Fig. 5. CAD model of 3D printed Block "O" for testing

A. dVRK software and hardware

The software architecture for the PSM arm can be divided into three main levels, as seen in Fig. 6, high-level control that communicates with the robot operating system (ROS) [4] libraries to control the arm in real time, low-level control such as PIDs which make the high level control possible and finally the hardware interfaces that are used by the low level control.

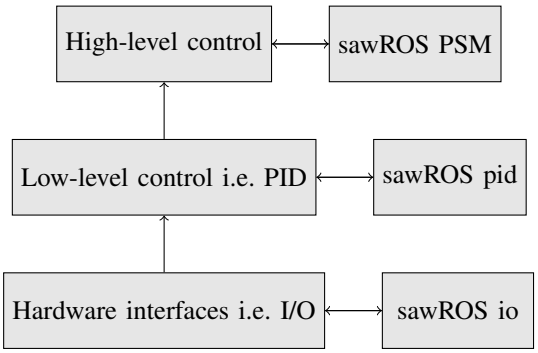

Fig. 6. High-level view of dVRK PSM software architecture

In general, the interfaces to many devices are implemented by the Surgical Assistant Workstation (SAW) package components which are built on open source *cisst* libraries. It also includes components that provide ROS interfaces. These components are basically a wrapper around software libraries provided by a primitive layer of C++ codes without any external libraries and a object oriented layer of C++ classes. The C++ classes provide

more convenient API than the primitive layer. Finally, we can use the sketch in Fig. 7 to develop a strategy for implementing the path planning for dVRK PSM arm. The SAW components provide high level control of the PSM arm to perform forward kinematics (FK), inverse kinematics (IK), and trajectory generation.

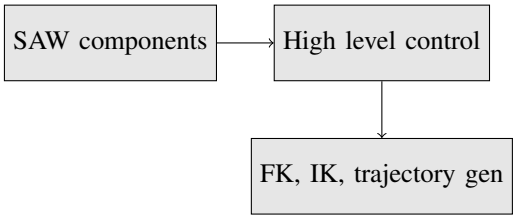

Fig. 7. Using high-level control commands to perform operations on dVRK PSM

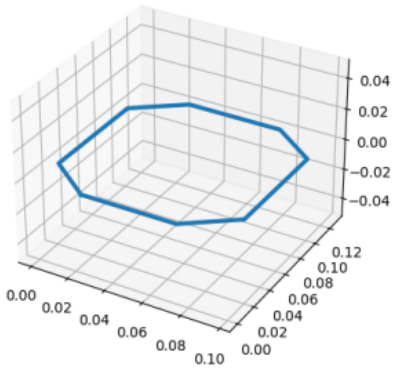

Fig. 8. Generated path for PSM arm

B. Path planning input

Based on the discussion from previous sections, we use the trajectory data, obtained from LSPB time-scaling, as the input to the PSM arm's high-level control commands for moving the arm. Note that the high-level control commands are implemented using dVRK-ROS python which uses Python scripts to perform various operations. For simplicity we let the built-in solver to take care of the inverse kinematics for figuring out the joint angles. We use a fixed frequency of 200 Hz, the same as the established ROS communication frequency, to discretize the time-scaling for all experiments. Additionally various values of velocity and acceleration are used for the LSPB time-scaling. The corresponding video demonstrations can be found here [5]. One benefit of using LSPB time scaling is that it can work with velocity and acceleration constraints. To demonstrate that, we use four different values of velocity and acceleration parameters while calculating the trajectories: (a) $v = 0.005\ ms^{-1}, a = 0.01\ ms^{-2}$, (b) $v = 0.0075\ ms^{-1}, a = 0.014\ ms^{-2}$, (c) $v = 0.01\ ms^{-1}, a = 0.02\ ms^{-2}$, (d) $v = 0.02\ ms^{-1}, a = 0.04\ ms^{-2}$. Corresponding kinematics simulations are also demonstrated in [5].

VI. RESULTS

The results are reported in this section. First, the FK and IK numerical results are presented and validated in the experimental setting. The trajectory generation and path planning generation results are then presented, with some challenges addressed.

A. FK and IK validation

As demonstrated in the supplementary videos, firstly, using the dVRK QT widget, each joint was actuated individually in 5 degrees increment from $-\theta_{limit}$ to $+\theta_{limit}$ of the corresponding joint, and the end effector position and orientation was saved via the $r.measured\_cp()$ command. The same joint values were fed into the FK code and the obtained position results were compared to the actual values obtained from the dVRK platform. For example, the error in x, y and z position when joint one is only actuated from -80 to 80 degrees is demonstrated in Fig. 9, in mm. The source of such error which fluctuates at certain joint angles in still unknown and under investigation. The orientation matrices however, were very well aligned in all cases. As an example, the position and orientation of the end effector with respect to the base frame is demonstrated in Fig. 10 when joint 1 is rotated +35 degrees.

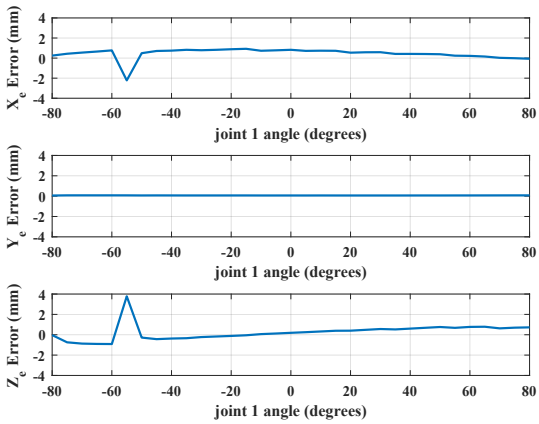

Fig. 9. End effector position error when joint 1 is only moving.

For the IK, the closed-form solutions of joint 1

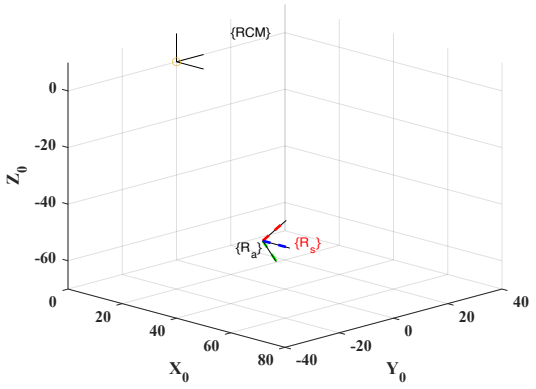

Fig. 10. Position and orientation of End effector when joint 1 is at 35 degrees. Ra is the frame obtained from the experiment and Rs is from simulation.

through joint 6 are reported below:

$$\theta_1 = atan2(px, pz - e)$$

$$\theta_2 = atan2(-py, px * s1 + pz * c1 - c1 * e - h)$$

$$\theta_3 = -L - h * c2 - py * s2 + (pz - e) * c1 * c2 + px * c2 * s1$$

$$\theta_4 = atan2(-Rot(1,2) * c1 + Rot(3,2) * s1, Rot(2,2) * c2 + Rot(3,2) * c1 * s2 + Rot(1,2) * s1 * s2)$$

$$\theta_5 = atan2(Rot(3,1) * (s1 * s4 + c1 * c4 * s2) - Rot(1,1) * (c1 * s4 - c4 * s1 * s2) + Rot(2,1) * c2 * c4, -Rot(3,1) * c1 * c2 + Rot(2,1) * s2 - Rot(1,1) * c2 * s1)$$

$$\theta_6 = atan2(-Rot(3,1) * c1 * c2 + Rot(2,1) * s2 - Rot(1,1) * s1 * c2, Rot(3,3) * c1 * c2 - Rot(2,3) * s2 + Rot(1,3) * c2 * s1)$$

Solving for an example desired position and orientation of the end effector:

$$\theta_{IK} = [20, \quad 22, \quad 1.5, \quad 15, \quad -150, \quad 25.71]$$
$$\theta_{actual} = [20, \quad 22, \quad 1.5, \quad 15, \quad 30, \quad 25.71]$$

Unfortunately the result for joint 5 was generated in a wrong quadrant. We still has not been able to solve this issue.

B. Implemented Trajectories

After the generated trajectories were validated via kinematic simulations in RViz, they were communicated to the physical hardware. Implementing on the physical robot provided better understanding of the appropriate speeds and dynamic control of the robot, as well as its accuracy in tracing a 3D printed Block "O". As in simulation, the dVRK followed the Block "O" path and end effector orientations as desired, with several caveats. These caveats include the robots z-axis positioning, its dexterous workspace, and positioning the Block "O", which will be discussed in more detail below.

Originally, the plan was to test the trajectory accuracy with a piece of paper placed in the forceps of the end effector that would follow inside a thin slit of 3D printed Block "O" path. This way accuracy could be validated without putting the end effector too close to an object. However, early on, we noticed that the height of the end effector creeped lower, risking collision with the Block "O" and table beneath it. This was not a drastic drop, only about centimeter, but enough to void our testing method. At first, we thought this was an error of the produced trajectory, but with quick investigation we learned that it was not and was a product of the hardware or control.

The next challenged we faced when getting our results was the limited dexterous work space of the dVRK for certain positions. This characterized limitation is only with respect to our off-characteristic task of tracing a relatively large path. This was not an issue we were expecting, as we assumed the dVRK was sufficiently coordinated. And the issue was not easily rectified as the dextrous workspace was not well understood nor visualized, unlike with articulated industrial robots. The dexterity limitations were due to the lack of coordination of the forceps mechanism. Still, even when a part of the path was outside the dextrous workspace and inside the reachable workspace, the dVRK was able to automatically satisfy the orientation goals as best it could while still maintaining proper positioning. As soon as the end effector re-entered the dextrous workspace, the desired orientation resumed.

The final challenge had to do with arbitrary positioning the 3D printed Block "O". The trajectory generation was designed to handle any scaling, reorientation, and offset of the Block "O" path. However, actually fixing the physical Block "O" in the desired way was quite challenging. Since all path transformations were with respect to the base frame at the RCM, three orthogonal measurements, in 3D space must be made to properly position the Block "O". Then there is the angular positioning of the Block "O" about three axes. Thus, with these measurement challenges, we resorted to testing several planar rotations of the Block "O" directly below the RCM. This reduced any extra error introduced from positioning and orienting the path. Besides occasional orientation issues of the dextrous workspace, as discussed above, the end effector was able to complete its path accurately.

VII. Conclusion

Here we solved and discussed the forward kinematics, closed form inverse kinematics, developed and implemented trajectories for the PSM arm of the dVRK robot. By taking advantage of the high-level control of the robot it was fairly convenient to generate interesting path planning algorithms. We successfully traced a Block "O" for a specific end effector orientation scheme, producing the results in a kinematic simulation and on the physical dVRK.

References

[1] The da vinci research kit (dVRK) – intuitive foundation. [Online]. Available: https://www.intuitive-foundation.org/dvrk/

[2] sawIntuitiveResearchKit. (2021) . [Online]. Available: ://github.com/jhu-dvrk/sawIntuitiveResearchKit/wiki

[3] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da vinci surgical system," in *IEEE Intl. Conf. on Robotics and Auto. (ICRA)*, Hong Kong, China, 2014, pp. 6434–6439.

[4] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[5] dVRK Inverse Kinematics and T. G. Project. (2021) . [Online]. Available: https://github.com/ferdous-alam/DVRK_robot_project