Work Report

Shivendra Srinivas

January 2020

1 Introduction

In the medical field and especially for surgeries, robots are being used for the benefit. Here, a 6-DOF UR5e robot is being used for aiding in spinal surgery. Such robots are now a necessity because of the possibility of having minimum error. The main focus of this project is to produce the simulation of the 6-DOF robot and to reduce any chances of collision.

2 Simulation of UR5e

2.1 Objective

To create the simulation of the UR5e 6-DOF robot.

2.2 Software verisons

Operating System: Ubuntu 16.04 LTS
Python: Python 3.5.2

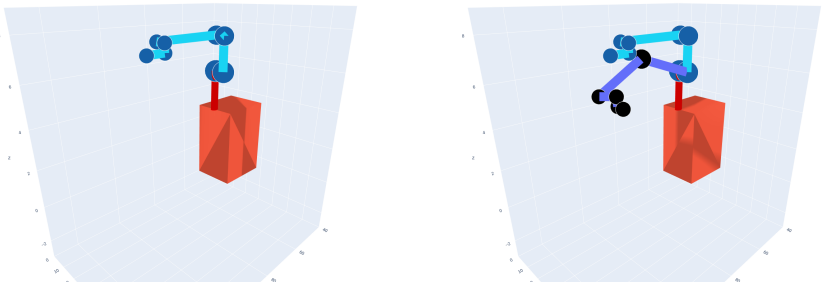
2.2.1 Python Packages

Numpy, plotly and scipy are the python packages that were used in this project.

2.3 Theory and Procedure

Here, the joint locations of the robot were found by using the values of the starting orientation of the given robot. This was done using the forward kinematics function. Once the coordinates of the six joints were found, two other coordinates were added to give an idea of how the actual robot is positioned. The cuboid represents the lower half of the robot. It is constructed by giving the coordinates of the cuboid and connecting them using triangular vertices. The cylinder is constructed so that the robot joints can be placed on top of it. It is being constructed by using parametric equations as given below:

$$x = r * cos(\Theta) \tag{1}$$



(a) Starting position of the robot (b) Ending position of the robot

Figure 1: Starting and ending position of the robot

$$y = r * sin(\Theta) \tag{2}$$

$$z = height \tag{3}$$

where

$$0 < \Theta < 2\pi \tag{4}$$

The joints are being shown by the markers. The cylinder is being created between the joints so that the links are visible clearly. The robot coordinates are moved with respect to the ending theta values of the given robot. The movement of the robot is happening between a certain time step which will be given by the user. The light blue cylinders helps in distinguishing between the starting position and ending position of the robot. The robot moves in frames and these frames are divided into the given time steps. This shows the simulation of the robot. This is visible in the figure 1(a)-1(b).

3 Collision

Collision is a process where two objects come together or collide with each other. In this module, we will be checking how the robot collides with obstacles, the patient and sometimes the robot itself.

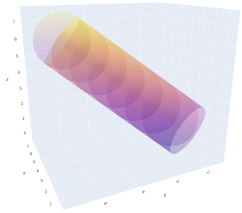
3.1 Obstacle Collision

3.1.1 Objective

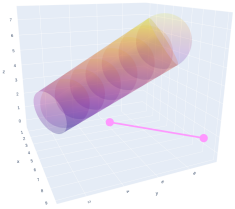
To check if the UR5e 6-DOF robot collides with any obstacle.

3.1.2 Theory and Procedure

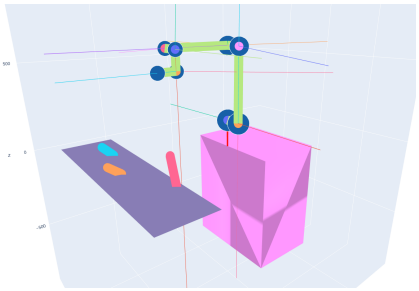
Here, the collision between the robot and the needles are being checked. During the spine surgery, the robot checks if the needle can be an obstacle to any of the robot joints.



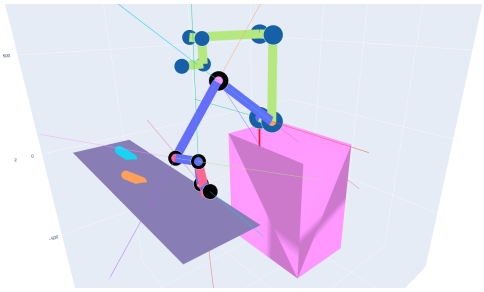
(a) Spheres being approximated as cylinders



(b) Intersection of line with spheres



(c) Initial position of the robot



(d) Final position of the robot

Figure 2: Obstacle Collision of the robot

To check the collision, obstacles are being created in the form of spheres which are being approximated to cylinders. Here, the cylinders are being considered as the needles used in surgery. The boundary circles of the cylinders are being created with the same radius as that of the sphere. These spheres are being created by using the parametric equations as given below:

$$x = xo + rsin\Phi cos\Theta \tag{5}$$

$$y = yo + rsin\Phi sin\Theta \tag{6}$$

$$z = zo + rcos\Phi \tag{7}$$

where

$$0 < \Theta < 2\pi \tag{8}$$

and

$$0 < \Phi < \pi \tag{9}$$

The cylinders are also being constructed using parametric equations as mentioned in equation (1-3). The spheres are being constructed between 2 points given by the user. A vector is being formed between the 2 points and the

euclidean distance of the vector is calculated. Now, this distance is being divided into certain steps. These points are being considered as the center for the spheres and by using the equations (5-7), the spheres are created. The x, y and z coordinates are being given by

$$p1 = p1 + \lambda(p2 - p1) \tag{10}$$

where λ is the step by which the vector moves.

This can be seen in figure 1(a).

Now, it is checked if the robot trajectory is colliding with any point on the constructed spheres. This is initially done by checking whether or not the line segment intersects the spheres. If the line segment intersects with the sphere, then 0 is returned as the line segment is intersecting the sphere and the minimum distance between the line segment and the sphere is zero. When the line segment does not intersect the spheres but the line intersects, the minimum euclidean distance between the intersection point and the points of the line segment is returned.

The intersection point between the line and sphere is done in the following manner:

Consider a line

$$p1 = p1 + u(p2 - p1) \tag{11}$$

Eq (11) comprises of x, y and z component. Now, it is known that, the equation of a sphere is given in eq (12) as:

$$(x - x3)^2 + (y - y3)^2 + (z - z3)^2 = r^2 \tag{12}$$

where x3, y3 and z3 are the coordinates for the center of the sphere.

Now, the line equation is substituted into the sphere equation and get a quadratic equation as follows :

$$a = (x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2 \tag{13}$$

$$b = 2[(x2 - x1)(x1 - x3) + (y2 - y1)(y1 - y3) + (z2 - z1)(z1 - z3)] \tag{14}$$

and

$$c = x3^2 + y3^2 + z3^2 + x1^2 + y1^2 + z1^2 - 2[x3x1 + y3y1 + z3z1] - r^2 \tag{15}$$

From eq (13)-(15), the discriminant can be found out by the equation

$$discriminant = b^2 - 4ac \tag{16}$$

if

- discriminant is < 0, then there is no intersection point.
- discriminant is $= 0$, then there is only one intersection point.
- discriminant is > 0, then there are two intersection points.

This is visible in figure 1(c)-1(d).

3.2 Plane - Line Collision

3.2.1 Objective

To check if the UR5e 6-DOF robot collides with a patient which is considered as a plane.

3.2.2 Theory and Procedure

Here, the obstacle is a patient which is considered to be a plane. Here, it is checked if the robot intersects with the patient. For this the idea of plane-line intersection is being done.

To check the plane-line intersection, a plane is being created. In 3D, a line intersects a plane in only one point or is parallel to the plane. A normal vector to the plane is considered here.

Consider the plane equation,

$$ax + by + cz + d = 0 \tag{17}$$

Consider the line equation,

$$p1 = p1 + s(p2 - p1) = p1 + su \tag{18}$$

Consider a point Vo on the plane. Now, to check if the line is parallel to the plane, we check if the dot product of the normal vector and the the line vector , ie.,

$$n.u = 0 \tag{19}$$

If the condition in equation (19) is satisfied, then the line is parallel to the plane. If this condition fails then the intersection point needs to be calculated.

Here, the vector between a point in the plane and a point on the line is considered to find the intersection point between the line and plane. ie.,

$$w = p1 - Vo \tag{20}$$

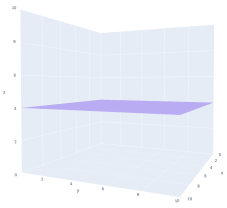
The intersection can be found out as follows :

$$SI = -n.w/n.u = n.(Vo - p1)/n.(p2 - p1) = -(ax + by + cz + d)/n.u \tag{21}$$

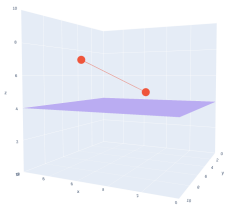
Once, SI has been calculated, from the equation (21), the intersection point I is calculated as follows:

$$I = p1 + (SIu) \tag{22}$$

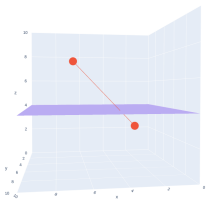
The intersection point is given by I in equation (22).



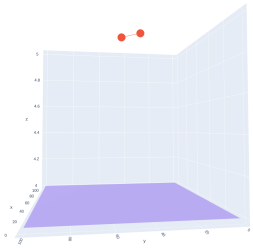
(a) The plane



(b) Intersection of line with plane



(c) Intersection of line segment with plane



(d) Line parallel to the plane

Figure 3: Plane - Line Collision

Now, the requirement is that when the line segment between p1 and p2 intersects the plane, the algorithm will return 0. If, the line intersects and the line segment does not, the minimum euclidean distance between the points of the line segment and the intersection point will be returned. If the line is parallel to the plane, then -1 will be returned.

Fig 3(a)-3(d) shows the different scenarios in which plane-line intersection takes place.

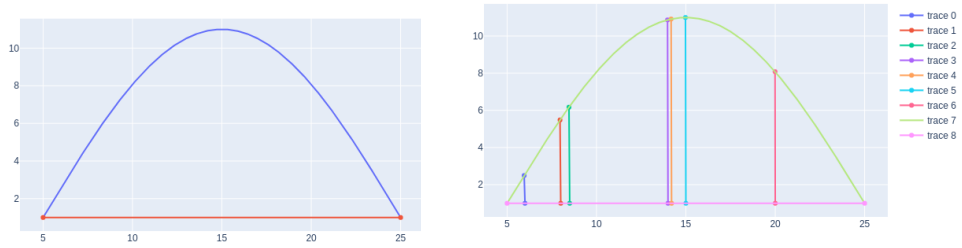
4 Path Planning

4.1 Objective

The objective is to create a smooth curve between two coordinates and find the distance between the line formed by the two points and the curve.

4.2 Theory and Procedure

The smooth curve is being created by sampling a set of points in between the two coordinates. Let the two coordinates be 'a' and 'b'. Let the maximum height of the curve be 'h'. The points are being sampled between the points a and b. Once sampled, these points are joined to form the curve. We know that x varies from the x component of 'a' to the x component of 'b'. ie.,



(a) The path curve

(b) Distance between the given set of points and the point of intersection on the curve

Figure 4: Path Problem

$$C \in [a[0], b[0]] \tag{23}$$

In eq.(23), C is the range of x axis.
For every value in x, y varies as follows.

$$B \in [a[1], b[1]] \tag{24}$$

$$y = hsin(t) + B \tag{25}$$

In eq(25), t varies as follows.

$$0 < t < \pi \tag{26}$$

To find the distance between the curve and a point on the line, the vector between two sampled points is being found and its corresponding y values are considered. A vector between the sampled points and a vector between its corresponding y values is considered. The Euclidean distance between the vectors a calculated and then the vector between the y coordinates is changed to that of the x distance. This helps us find the intersection point with the curve. Once that is calculated, the distance is calculated between the given point and the point of intersection.

5 URScript

5.1 Objective

The objective is to make a robot move in a given space using the commands in the URScript.

5.2 Theory and Procedure

When the robot is given a certain path to traverse, the robot will move in the given path. When intermittent pose of the robot is given, the robot must be in a position to move through to intermittent pose to the final pose. To achieve this, URScript is being used with the help of the python package urx.

Some of the commands in the URScript make the robot not have smooth movements. To stop this, there are various other functions in the URScript that will perform the necessary task.

6 Collision Testing

6.1 Objective

To test different kinds of collisions, every link in the robot is being approximated as a cylinder. The cylinder is formed by constructing many lines at a certain radius from the joint positions of the given link.

6.2 Theory and Procedure

Here, the line is being considered as a link and the link is being surrounded by a cylinder. To generate this cylinder, the lines are being generated around the link vector and then those lines are being approximated as cylinders.

Consider the two points 'p1' and 'p2' from which the link is described. Now, to get the lines surrounding the link, the vector between 'p1' and 'p2' is found, ie.,

$$v = p2 - p1 \tag{27}$$

Now, a vector which is not in the same direction as 'v' is used. Let this vector be assumed as 'V'.

Now, a vector perpendicular to v is found, ie.,

$$vPerpendicular = cross(v, V) \tag{28}$$

Once 'vPerpendicular' has been found, covert it into a unit vector and consider it to be 'n1'.

Now, a vector perpendicular to 'v' and 'n1' needs to be found, ie.,

$$n2 = cross(v, n1) \tag{29}$$

Now, form the lines in the by using the parametric form of the cylinder as follows,

$$X = p1[0] + n1[0] * r * sin(\theta) + n2[0] * r * cos(\theta) \tag{30}$$

$$Y = p1[1] + n1[1] * r * sin(\theta) + n2[1] * r * cos(\theta) \tag{31}$$

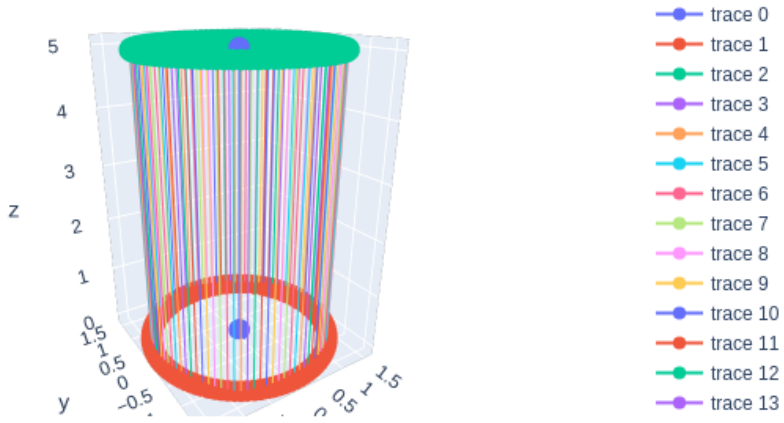


Figure 5: The cylinder

$$Z = p1[2] + n1[2] * r * sin(\theta) + n2[2] * r * cos(\theta) \tag{32}$$

Do the similar process with the point 'p2' and get 'X1', 'Y1' and 'Z1'.

Once the points 'X', 'Y' ,'Z', 'X1', 'Y1' and 'Z1' are available, create the lines between the x, y and z coordinates by using the commands in plotly.

7 TCP Calibration

7.1 Objective

The Tool Center Point (TCP) is being moved to another point by using an end effector. To check if the end effector is positioned in the right way, the robot's TCP needs to be calibrated.

7.2 Theory and Procedure

To calibrate the TCP, we need to first set the robot in the home position and place the end effector in the holder given. Once that is done, the position, orientation and payload needs to be calibrated.

7.2.1 Position Calibration

Attach the position calibration tool to the end effector and move the robot to the center of the table which consists of the position calibration point. Take 4 different sets of readings with different orientations. Once the position calibration is done, the next step is to do the orientation calibration.

7.2.2 Orientation Calibration

Remove the position calibration tool from the end effector. Once this is done, go to Plane which is there in the Features tab under installation. To design the plane, the robot end effector needs to be placed in the golden pole on the table. Place it in 3 such golden poles. Once these positions have been set, go to the Orientation option in the TCP tab and in the drop down menu, choose the plane that has been created from the above method. Then set the point of the final plane point. The orientation of the TCP has been calibrated. The payload must also be calibrated for the TCP.

7.2.3 Payload Calibration

Keep the center of gravity 'on' in the payload tab. Move it to four different points in the given region of the robot movement and set these points. Give finalize. Once this is done, the payload calibration is completed.

This is how the TCP is being calibrated for the robot. This needs to be done for any kind of end effector after it is removed.

8 Singularity of UR5 Robot

8.1 Objective

To find the singularities of the UR5 robot.

8.2 Theory and Procedure

A robot singularity is a configuration in which the robot end - effector becomes blocked in certain directions.At a singularity, the robot loses one or more degrees of freedom. Singularities can occur when two axes are aligned or when the robot is taken to its maximum limit in both position and orientation. There are three kinds of singularities which are the shoulder, elbow and wrist singularity.

The first type of singularity is the wrist singularity. Wrist Singularity occurs when joints 4 and 6 are coincident to each other. In a wrist singularity, the robot cannot move in the direction of the axis of joint 5.

The second type of singularity is the elbow singularity. The elbow singularity occurs when the robot center lies on the plane passing through the axes of joints 2 and 3. This singularity is the least unexpected and is easy to avoid.

The third type is singularity is the shoulder singularity. The shoulder singularity occurs when the robot center lies on the plane passing through joints

1 and 2. This singularity is the most complex as it does not depend on a single joint position, as do the other two.

The given algorithm helps in finding the singularities before the singularity occurs. The joint coordinates are found by passing the joint values into the FK function. The vectors are taken with respect to the FK function. To find the wrist singularity, two vectors are found between joint 2 and joint 3, and joint 4 and joint 5 respectively. If the vectors are parallel to each other, wrist singularity occurs. The elbow singularity is found by finding the vectors between joint 0 and joint 1, ad joint 1 and joint 2. If these 2 vectors are parallel to each other, elbow singularity occurs. To calculate the shoulder singularity, first the vector between the joints 0 and 1 of the actual robot needs to be found. Once, this vector is found a plane needs to be found that is in the direction of the center of the robot. If the joint 3 and 4 of the actual robot lies in this plane, then the shoulder singularity occurs.

This algorithm helps in finding if there is any singularity during the motion of the robot.

9 Accuracy Testing

9.1 Objective

The objective is to check the accuracy of the robot to a given position.

9.2 Theory and Procedure

The robot is being moved to different points throughout a graph sheet and the position is being checked when run via URScript. If there is a difference, the error is being noted and it is checked if the error is within the limits. The physical error and the robot system error is being calculated. The error is being kept to a minimum of 0.5mm. This is being checked for a set of entry points and a target point. The entry points are being generated by shifting the given vector by an angle in x - axis and an angle in the z - axis. The distance of the vector is given by the user. Once, a set of entry points are available, each entry and target point are taken and checked and the movement occurs to the target point.

10 Dexterous Workspace

10.1 Objective

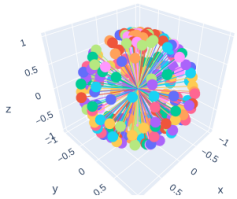
The objective is to find the dexterous workspace or dspace of the robot.

10.2 Theory and Procedure

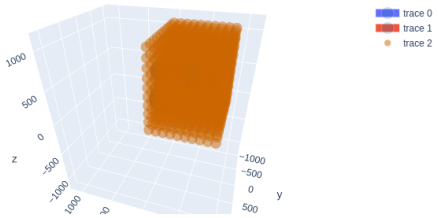
The workspace of a robot is the points that can be reached by the robot end effector. The reachable workspace is the positions that the robot can achieve in different orientations. The dspace on the other hand is the different 3-D positions that the robot can reach in all possible orientations. The dspace is a subset of the reachable workspace. The dspace can be found by first calculating all possible orientations.

To calculate all possible orientations, the robot is kept at the home position and from URScript, the rx, ry and rz are taken from the home position of the robot. The URScript gives the axis - angle rotation values from which the initial rotation matrix is being found. By using rodrigues rotation along both xy and xz rotation by keeping x as the axis, the rotation matrices are being found. y is being rotated with x around 360 degrees and z is being rotated with x around 180 degrees. Once, the rotation matrices are found, all possible positions must be found.

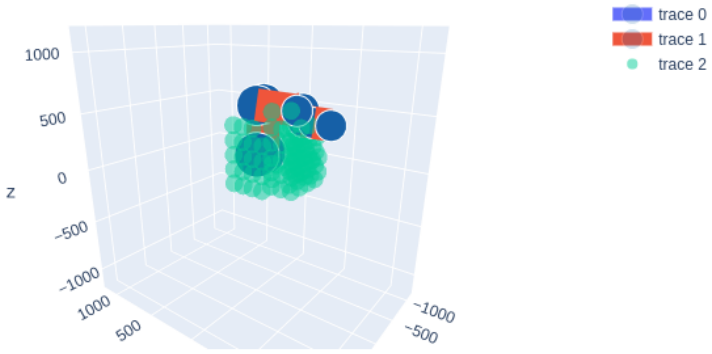
To calculate all possible positions, the robot end - effector is initially sent to the extreme right, extreme left and extreme top positions. The 3-D positions are taken into consideration and a cuboid is being created with all possible points from the aforementioned points.



(a) All possible Orientations



(b) All possible Locations



(c) Dexterous Workspace of the End - Effector

Figure 6: Dexterous Workspace

After performing the following functions, each 3-D point is being tested with all possible rotation matrices (orientations). If the 3-D point satisfies all possible orientations, then the given 3-D point is a part of the dspace. This is how the dspace is being found for the end - effector.

11 Reference Literature

In [12], substantial work in kinematics and dynamics has been done by the author. The entire process is teleoperated and the drilling mechanism has a high-precision force sensor. Haptic feedback is being used for teleoperation and due to this, there is a change in the caretesian coordinate system which fixes the new coordinates for the robot. The robot moves through these coordinates.

In [22], the robot is a multi-arm 6 DOF robot which is being used for facial surgeries. Here, the author has used DH parameters for the kinematics of the robot. Collision detection is based on modeling geometric shapes such as the cylinder and the sphere.

In [15], the robot is a 6 DOF robot which is being used for minimal invasive brain surgeries. Collision detection is based om the homogeneous transformation matrices of the tip and shaft of the robot. The collision between the shaft and the opening of the brain is considered as a collision here.

In [8], the robot is a 6 DOF robot which is being used for spinal surgery. Substantial work in kinematics and dynamics has been done by the author. Here, imaging (such as CT scan,etc) are being used to find the position of the patient and hence perform the surgery.

In [3], the author has done substantial work on kinematics and dynamics. Here, if there is any collision, the vibration armband gives the signal. Master-Slave (Teleoperation) system and haptic feedback is being used here. The joint angles are being calculated using Jacobian matrix. Here, collision is being found out be finding the position and orientation of the obstacle.

In [16], the author has done substantial work on kinematics and dynamics. Here, firstly all the registrations are being done for the robot, the patient and other significant objects. A collision constraint is being created by using the separate axis theorem (SAT). The error can be minimized by using optimization techniques. Here, the boxes created from triangular meshes is considered.

In [17], the author has explained the various models of medical robots available and what technology do these robots use. Here, robotic assisted navigation (RAN), telesurgical robotic systems and Virtual Augmented Reality techniques are being discussed.

In [9], the author has used a 6 DOF arm robot as an actuator to the venipuncture system. Substantial work has been done on the kinematics and dynamics of the robot. Here, A* algorithm is being used for path planning. For collision, the obstacles are being assumed to be shaped and they are kept in a certain boundary.

In [21], the author has tried to combine inverse kinematics method with a collision avoidance scheme. Here, the obstacle avoidance algorithm is being used

after computing the joint locations and then inverse kinematics is done.

In [4], augmented reality is being used for path planning of the robot. Here, the author is trying to use the image properties to find out the path virtually by finding the entry point on the patient. The minimum distance between the pedicle device and the pedicle wall was found for technical and clinical studies.

In [10], the author has done substantial work on kinematics and dynamics. Here, the real time position of the robot is taken and then the distance is found by considering the a boundary of the obstacle. Kinematic Continuous Collision Detection Library (KCCD) is used here for collision and path planning.

In [5], the author has used DH parameters for forwards kinematics. Here, path planning method is combined with linear interpolation method and artificial potential field method. The robot arm is calibrated with the improved circular datum self-calibration method.

In [1], the robot is being used for laparoscopic surgery. Here, the author has done substantial work on the kinematics and dynamics of the robot. Here, the Probabilistic Roadmap (PRM) is a popular method for path planning that the author has used. Here, reinforcement learning is being used because of its performances in a probabilistic environment.

In [20], the author has done substantial work on kinematics and dynamics. Here, shadow space modelling is used for path planning and collision free path. An optimization problem is being defined here to find the collision free path.

In [7], the author has used an intermediate point obstacle avoidance (IPOA) algorithm to ensure smooth trajectory and efficient movement for obstacle avoidance. Here, the model of he robot is first being simplified. DH parameters is being used for forward kinematics. The IPOA is used to steer the arm away from the obstacles.

In [11], the author has done substantial work on kinematics and dynamics. Here, optimization techniques are used to find the best IK solution for the robot and then errors are also being minimized respectively. Here, the paths are being traced many times by using line tracing, spline tracing, polygon tracing etc. This gives the path of the robot.

In [13], the author has used deep neural network to detect the obstacles and in this process, he picks and places objects. Here realsense SR 300 RGB-D sensor is being used. Here environmental monitoring and obstacle tracking is done and then by using potential field algorithm, the obstacle avoidance algorithm is done.

In [18], the author uses an autoregressive model and optimizes the path and avoids obstacles easily. Here, controlling action is being performed on the joint variables. Here, the author has done substantial work on the kinematics and dynamics of the system.

In [6], the author has done substantial work on the kinematics and dynamics of the robot. Here, sensorless collision is trying to be performed by using Euler - Langrange equations.

In [19], here the author has done substantial work on the kinematics and dynamics of the robot. Here, the Octree model is used to express environment and detect collision.

In [2], the author has done substantial work on the kinematics and dynamics. Here, the author is trying to detect collision by using kinematics itself. Here, screw theory is being used to define a screw motion deviation (SMD) as the distance between the expected and the actual instantaneous screw axis (ISA) of motion.

In [14], the author has done substantial work on the kinematics and dynamics. Here, SpineAssist software is being used to handle the movement of the pedicle screw. The software consists of guided manual execution where the robotic device is brought to the necessary location.

12 Conclusion

These algorithms were performed in the project named "An image guided robotic system for minimally invasive spine surgery" under Mr. Manojkumar Lakshmanan (Project Lead, Spine Robotics Team) and Shyam A (Project Engineer, Spine Robotics Team). The codes can be found at

- Codes : https://github.com/shivendra718/HTIC

References

[1] Donghoon Baek, Minho Hwang, Hansoul Kim, and Dong-Soo Kwon. Path planning for automation of surgery robot based on probabilistic roadmap and reinforcement learning. In *2018 15th International Conference on Ubiquitous Robots (UR)*, pages 342–347. IEEE, 2018.

[2] Andrea Bajo and Nabil Simaan. Kinematics-based detection and localization of contacts along multisegment continuum robots. *IEEE Transactions on Robotics*, 28(2):291–302, 2011.

[3] Joao Bimbo, Claudio Pacchierotti, Marco Aggravi, Nikos Tsagarakis, and Domenico Prattichizzo. Teleoperation in cluttered environments using wearable haptic feedback. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3401–3408. IEEE, 2017.

[4] Gustav Burström, Rami Nachabe, Oscar Persson, Erik Edström, and Adrian Elmi Terander. Augmented and virtual reality instrument tracking for minimally invasive spine surgery: a feasibility and accuracy study. *Spine*, 44(15):1097–1104, 2019.

[5] Chengtao Cai, Boyu Wang, and Yi Li. Research on cooperative work path planning method based on double 6-dof manipulators. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 12–17. IEEE, 2018.

[6] Saixuan Chen, Minzhou Luo, and Feng He. A universal algorithm for sensorless collision detection of robot actuator faults. *Advances in Mechanical Engineering*, 10(1):1687814017740710, 2018.

[7] Zhuo Chen, Weihua Su, Bin Li, Baosong Deng, Hang Wu, and Baozhen Liu. An intermediate point obstacle avoidance algorithm for serial robot. *Advances in Mechanical Engineering*, 10(5):1687814018774627, 2018.

[8] Asham Khan, Joshua E Meyers, Ioannis Siasios, and John Pollina. Next-generation robotic spine surgery: first report on feasibility, safety, and learning curve. *Operative Neurosurgery*, 17(1):61–69, 2019.

[9] Fenglei Li, Zexin Huang, and Lin Xu. Path planning of 6-dof venipuncture robot arm based on improved a-star and collision detection algorithms. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2971–2976. IEEE, 2019.

[10] Dennis Mronga, Tobias Knobloch, José de Gea Fernández, and Frank Kirchner. A constraint-based approach for human–robot collision avoidance. *Advanced Robotics*, 34(5):265–281, 2020.

[11] Pragathi Praveena, Daniel Rakita, Bilge Mutlu, and Michael Gleicher. User-guided offline synthesis of robot arm motion from 6-dof paths. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8825–8831. IEEE, 2019.

[12] Sina Rezazadeh, Weibang Bai, Mingjing Sun, Shihang Chen, Yanping Lin, and Qixin Cao. Robotic spinal surgery system with force feedback for teleoperated drilling. *The Journal of Engineering*, 2019(14):500–505, 2019.

[13] Kai-Tai Song, Yu-Hsien Chang, and Jen-Hao Chen. 3d vision for object grasp and obstacle avoidance of a collaborative robot. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 254–258. IEEE, 2019.

[14] W Sukovich, S Brink-Danan, and M Hardenbrook. Miniature robotic guidance for pedicle screw placement in posterior spinal fusion: early clinical experience with the spineassist®. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 2(2):114–122, 2006.

[15] Hiroaki Ueda, Ryoya Suzuki, Atsushi Nakazawa, Yusuke Kurose, Murilo M Marinho, Naoyuki Shono, Hirofumi Nakatomi, Nobuhito Saito, Eiju Watanabe, Akio Morita, et al. Toward autonomous collision avoidance for robotic neurosurgery in deep and narrow spaces in the brain. *Procedia CIRP*, 65:110–114, 2017.

[16] A Uneri, T De Silva, J Goerres, MW Jacobson, MD Ketcha, S Reaungamornrat, G Kleinszig, S Vogt, A Jay Khanna, GM Osgood, et al. Intraoperative evaluation of device placement in spine surgery using known-component 3d–2d image registration. *Physics in Medicine & Biology*, 62(8):3330, 2017.

[17] Gianluca Vadalà, Sergio De Salvatore, Luca Ambrosio, Fabrizio Russo, Rocco Papalia, and Vincenzo Denaro. Robotic spine surgery and augmented reality systems: A state of the art. *Neurospine*, 17(1):88, 2020.

[18] Yiwei Wang, Yixuan Sheng, Ji Wang, and Wenlong Zhang. Optimal collision-free robot trajectory generation based on time series prediction of human motion. *IEEE Robotics and Automation Letters*, 3(1):226–233, 2017.

[19] Lu Wu and Yoichi Hori. Real-time collision-free path planning for robot manipulator based on octree model. In *9th IEEE International Workshop on Advanced Motion Control, 2006.*, pages 284–288. IEEE, 2006.

[20] Hyun Joong Yoon, Seong Youb Chung, and Myun Joong Hwang. Shadow space modeling and task planning for collision-free cooperation of dual manipulators for planar task. *International Journal of Control, Automation and Systems*, 17(4):995–1006, 2019.

[21] Liangliang Zhao, Jingdong Zhao, Hong Liu, and Dinesh Manocha. Efficient inverse kinematics for redundant manipulators with collision avoidance in dynamic scenes. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2502–2507. IEEE, 2018.

[22] Xiang-yu Zhu, Xing-guang Duan, Chao Chen, Tian-bo Liu, and Qing-bo Guo. A model of collision-detection for maxillofacial multi-arm surgery robot. In *2012 IEEE International Conference on Automation and Logistics*, pages 539–544. IEEE, 2012.