Stochastic Motion Planning for the Telebot

Yonah Elorza, Miami Dade College, USA
Sevugarajan Sundarapandian*, Acharya Institute of Technology, India
Jerry Miller, Florida International University, USA

This paper evaluates stochastic motion planning in a three dimensional space for $TeleBot^{\#}$, Florida International Universitys telepresence robot. Stochastic motion planning is necessary in an operator controlled robot due to device latency, device failure, connection failure, or user error. As such, three different modeling methods and their corresponding motion planning algorithms are evaluated for use in stochastic motion planning. The modeling and planning algorithms used are Configuration Spaces (C-Spaces) with Sampling Based Motion Planning, Denavit-Hartenberg (DH) Parameters with Inverse Kinematics, and the Universal Robot Description Format (URDF) with the Open Motion Planning Library (OMPL).

Keywords: Telebot, telepresence, motion planning, Denavit-Hartenberg (DH), Inverse Kinematics, C-Spaces, URDF, OMPL, Robotic Operating System.

1. INTRODUCTION

Telebot's[Prabakar and Kim [2013]] role as a telepresence robot meant for law enforcement dictate that the majority of its usage would be in the field, i.e. in stochastic environments. Being an electronic representation of a human, it does have various limitations which affect operation of the robot, and creates limits on the operator. As such, the operator cannot be made reliable for complete control of the robot, as a number of failures could occur, and some form of semi-autonomy should be implemented in the form of motion planning algorithms which attempt to control the robot based off existing configurations and intended goal positions. The idea is to find a model and planning algorithm which can handle stochastic environments (in other words, pre-planning is not needed to encourage movement), and implement it into the robot.

The planning algorithms explored in this paper branch from the modeling methods used to describe the robot. The first modeling method, configuration spaces (C-Spaces), involve using a mathematical framework from sets which defines a robots configuration, the region where it can move, and obstacle regions which the robot cannot pass through. The second model, Denavit-Hartenberg (DH) Parameters, defines the serial-link mechanism geometry [Hartenberg and Denavit. [1955]; Corke [2007]] of a manipulator on a robot, through transformation matrices for all links to one base link, and then allows different algorithms to attempt motion planning. The third model, Universal Robotic Description Format (URDF) [Garage [2012]], is a modeling method coupled with the Robotic Operating System (ROS)[Quigley. et al. [2009]] which defines every part of the robot in terms of links and joints, and allows for exact description of maneuverability of any part of the robot.

Motion planning is a central problem in robotics, where any robotic or autonomous device can find efficient paths to any goal position. The problem is shown to be PSPACE-complete [Canny [1988]], indicating no complete solution to the problem. Existing motion planning algorithms settle on finding a weaker completeness for any problem solved. Sampling-based motion planning [Kavrak and LaValle [2008]] is a planning method which branches off from C-Space Modeling, and it places random nodes in the region of movement which wont position the robot in a collision with an obstacle. The nodes are then connected via vertices which attempt to navigate from the

*Corresponding author

#The TeleBot project is an ongoing effort at the Florida International University (FIU) Discovery Lab, where researchers are designing, developing and testing a new telepresence robot designated as TeleBot. More specifically, the work has been focused on the design and implementation of a smart telepresence robot with enhanced functionalities.

initial position of the robot to the goal position through the shortest amount of vertices. Various algorithms exist for this planning method which attempt to optimize the path as well as the robots orientation when it comes to planning where the robot moves to. Inverse kinematics, which branches from DH Modeling, defines any goal point in three dimensional space, and attempts to lead an end-effector from its three dimensional starting position to the goal position. It does this through a series of transformation matrices for each link starting from the base link and finishing at the end-effector. The Open Motion Planning Library [Sucan and Mark Moll [2012]] is an open source library coupled with ROS, and branches off from URDF Modeling. It consists of multiple planning algorithms which solve complex motion planning problems without much input from an operator. The OMPL is not a single planning algorithm, but is rather a collection of them, which accurately determines which planning algorithm should be used in any particular situation.

Throughout this paper, we evaluate the efficiency, simplicity, and usability of the different modeling methods and motion planning methods in order to determine the best model and algorithm to attempt stochastic motion planning.

2. MODELING

The Telebot was first modeled using C-Spaces. An arbitrary center point was defined at the point where his neck and body connected (as this allowed for the most convenient reference point for defining the actuators and end effectors). From this, the following model was made:

$$W \subset \mathbb{R}^3$$
$$C \subset W$$
$$\{C_{obs} \subseteq C : -0.3025 < x < 0.3025, -0.185 < y < 0.185, -0.63 < z\}$$
$$C_{free} = \frac{C}{C_{obs}}$$
$$C_1 \subseteq C_{free} : \frac{(x-0.325)^2}{0.77^2} + \frac{(y)^2}{0.77^2} + \frac{(z)^2}{0.77^2} = 1\}$$
$$C_2 \subseteq C_{free} : \frac{(x+0.325)^2}{0.77^2} + \frac{(y)^2}{0.77^2} + \frac{(z)^2}{0.77^2} = 1\}$$

The workable region, W, for the robot is three-dimensional. In this workable region, a robotic configuration space, C, can be found. An obstacle configuration space, C_{obs} is found with the body, with a central point in the center of the chest, which creates a rectangular region as defined above. A free configuration space, C_{free} can then be defined by the components of the configuration space not made up of the obstacle space. From this, two configuration spaces, C_1 and C_2, for the arm can be created. This is done by creating an sphere-shaped region (the actual region an arm can travel) offset from a central point, again defined as the center of the chest, and using the shoulder as the origin point for the sphere. The 0.3025 in both arm configuration spaces represents the offset in meters from the chest to either shoulder, while the 0.77 represents the length of the arm in meters.

A major difficulty in this modeling was attempting to define the arms as obstacle spaces, as both arms can collide with each other, thus being obstacle spaces that can move around a designed free configuration space. Another major difficulty is trying to define moving objects in the configuration space as obstacles, as the C_{free}-space and the C_{obs}-space would have to be constantly updated with changes, making it non-optimal for use in stochastic environments where obstacle regions change frequently due to no pre-made map being used. Finally, making this model involves writing the code completely, and creating programs which can understand the framework of the model. This requires us to explore other options for modeling.

The second model used for the Telebot was DH parameters. We define each arm's hand actuator as the end effector, and the shoulder actuator as the base actuator for each arm, and define all other actuators in terms of this base. The arm is modeled as seen in Fig. 1.

From this, we can begin to describe all the actuators using transformations. For reaching the end effector from the base actuator, it is defined as the following:

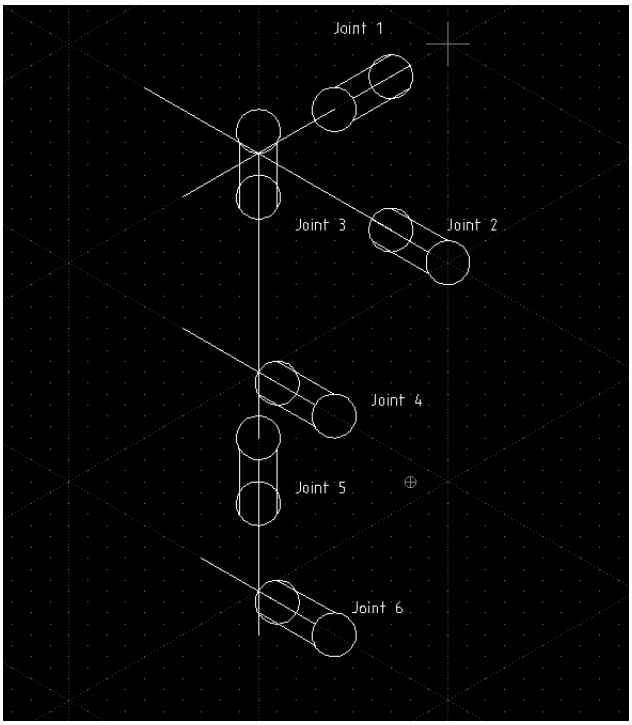

Figure 1. Denavit-Hartenberg Visual Representation

$$\text{Left Arm} =$$
$$T_z(d_0)R_z(\alpha_0)T_x(d_1)R_x(\theta_1)R_z(\alpha_1)...$$
$$T_z(d_2)R_z(\alpha_2)T_y(d_3)R_z(\alpha_3)T_z(d_4)R_z(\alpha_4)$$
$$\text{Right Arm} =$$
$$T_z(d_0)R_z(\alpha_0)T_x(d_1)R_x(\theta_1)R_z(\alpha_1)...$$
$$T_z(d_2)R_z(\alpha_2)T_y(d_3)R_z(\alpha_3)T_z(d_4)R_z(\alpha_4)$$

$$\theta_1 = \theta_4 = \tfrac{\pi}{2}$$
$$\alpha_0 = \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \tfrac{\pi}{2}$$

Where every T and R is a translation and rotation pair(overall, one single transformation) from a link i to a link i+1. All d_i are distances between respective link pairs, α_i represents angle offset from link pairs, and θ_i represents the angle offset from link pair's axis of actuation. This is simple for the actuators in Telebot in particular, as all the z-axis of the actuators line up, eliminating the need for the θ_i parameter of the DH in a lot of cases, only requiring α_i and a_i. d_i is excluded by default since all the actuators are revolute, not prismatic. The difficulties of DH Parameters are mostly implementation, as all the code and modeling would have to be done from the ground up, as well as the fact that any modifications done to the actuators or positions of the hand would need an entire rework of the model. Additionally, it is relatively difficult to model obstacles, and often requiring some external mechanism.

The final modeling used for the Telebot is URDF. URDF is coupled with ROS, and is a XML based modeling format. The model can be visually represented with ROS's built in visualization software, Rviz. This visualization can be seen in Fig. 2.

Every component of the robot is defined using visual, collision, and inertial models with links and joints. This can be shown as a state model, a graphic representation of XML components in the robot model, in Fig. 3.

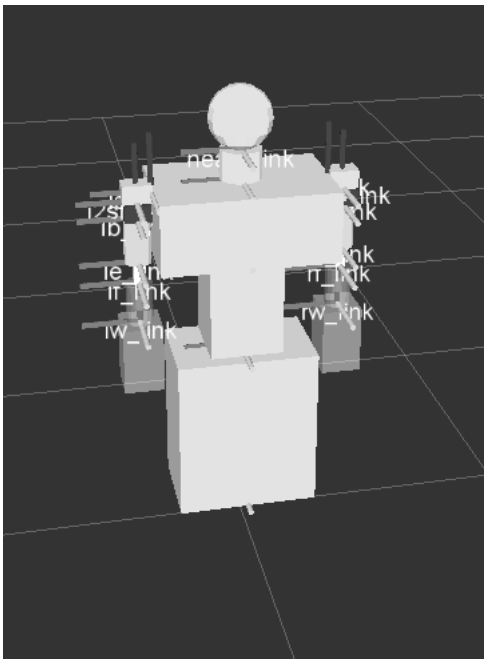

Figure 2. Telebot Model in URDF

URDF modeling has various advantages the other two models don't have, including it is an existing framework and modeling software is coupled to it. It also deals with the fact of collisions with each arm as well as the body, through collision modeling. Disadvantages include a lack of obstacle defining in the URDF, instead having to create custom URDF files for each obstacle.

3. MOTION PLANNING ALGORITHM

The first planning algorithm approached was sampling-based motion planning using C-Spaces. The goal of sampling based motion planning is to connect the initial configuration, which is the state that the robot is intended to begin operation, to the goal position through a series of nodes which navigate through the C_{free} space.

Implementing this algorithm would involve finding a way to create arbitrary nodes in three-dimensional space a unit distance from each other, and some sort of vertex implementation which attempts connecting the nodes and determining the shortest path to the goal from the initial position. It would also involve creating an obstacle region for the arms, where the arms can collide with obstacles, and each component of the arm must be maneuvered within the joint ranges to avoid collisions or else the planning would fail. This segmentation of the arm into components which can collide causes problems, as several obstacle spaces must be created which are also moving.

Additionally, creating nodes in three-dimensional space proves to be problematic, as the most non-trivial way to do so is to divide the three-dimensional space into various unit thickness two-dimensional slices, which involves huge amounts of grid polling to test for possible node positions, and either requires large unit distances between slices, or large computational power to do grid polling for large amounts of grid slices (as the thickness of the two-dimensional grid slices decreases, computational time increases, as more two-dimensional grids can be added). The amount of two-dimensional slices also determines the possible movement areas of the robot, and limiting the amount of slices limits move-able areas for the robot.

Finally, and most importantly, attempting to do sampling based motion planning in a stochas-

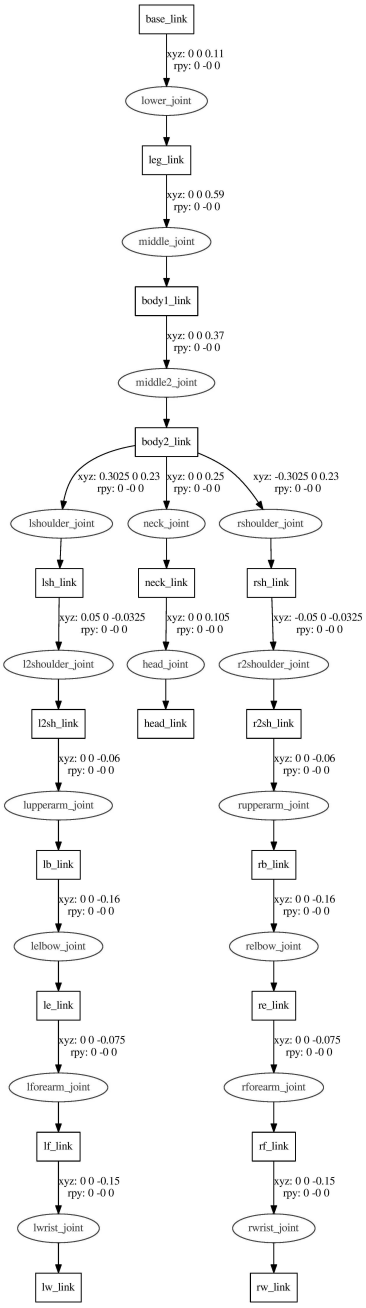

Figure 3. Telebot URDF Model State Diagram

tic environment in three-dimensions involves the earlier difficulty of C-Spaces having problems with moving objects in the spaces, as well has having the requirement of re-polling the three-dimensional grid as soon as an object moves, and doing so quickly enough that it's practical in real time, which is not easily achieved with sampling based motion planning.

The second planning algorithm analyzed for use was Inverse Kinematics, which attempts reach-

ing a three dimensional point using transformation matrices from an end effector, and moves all other actuators by manipulating the actuators through the DH model.

Implementation of this algorithm involves defining the arms using the parameters, as is done with the model, and then finding targeted three-dimensional points from this.

Immediate problems are seen with the model, the most basic of which involves the lack of any form of obstacle detection or obstacle accounting in the model. There is no way to define three-dimensional through DH Parameters, requiring some other modeling method to do so. There are also difficulties to modify transformation matrices based off of obstacles, requiring a different planning algorithm to do so. Finally, three-dimensional points must be somehow pre-defined or some way of picking arbitrary three-dimensional points has to be implemented for the planning to actually occur. In stochastic environments, it would constantly have to pick different three-dimensional coordinates through an operator instead of being semi or completely autonomous, and works more for reaching an obstacle, not avoiding them.

Finally, the OMPL is evaluated for possible Telebot implementation. The OMPL features a variety of planning algorithms[Sucan and Mark Moll [2012]]. It polls a desired goal position and then determines the best planning algorithm to arrive at said goal position.

An important feature of the OMPL is the native support with ROS, as this allows for easy communication with the model and other programs. Another key aspect is the integration with the visualization software Rviz, offering a visual aspect to the motion planning which would have been implemented separately with the other two models and planning algorithms. Finally, the visualization software allows for adjustment of actuator positions into intended goal positions, and polls whether the new position is possible, and then attempts to move to that position.

Weaknesses with OMPL are the fact that the user cannot choose which planning algorithm is to be used without changing a number of configuration settings. There is also no stochastic area gathering that is native, requiring code modification. However, there does exist a large amount of ROS packages which allow for obstacle detection and mapping onto Rviz, eliminating that problem. A final problem is the lack of ability to visualize robot movement over an area which the other two models do allow. However, this is circumvented with the additional use of a simulation software built closely to ROS, Gazebo.

4. SELECTION OF BEST METHOD

From comparison of the three different modeling and planning methods, the most suited for stochastic motion planning for Telebot (as well as stochastic motion planning in general) seems to be the OMPL with URDF modeling.

For modeling, URDF offers an existing framework for creating a model computationally which is based off of XML, while Configuration Space modeling and Denavit-Hartenberg Parameter modeling only exists mathematically, requiring some computational interface to be created in order to actually model the robot. There is also visualization software for URDF modeling through Rviz, which is not offered for the other methods and as thus would have to be created.

In terms of motion planning, the Open Motion Planning Library is more efficient than sampling based motion planning and inverse kinematics, as the Open Motion Planning Library includes a host of different planning libraries which are selected in runtime for optimal efficiency.

The main issue that determined which model and planning algorithm to use was how each responded to stochastic environments. While none of the models have native support for stochastic environments, URDF modeling works off a simple three dimensional coordinate system, and as such moving objects are easily described as changing coordinates, while Configuration Spaces involve manipulation of three-dimensional spacial sets, and Denavit-Hartenberg Parameters involve three dimensional matrices of an object relative to a defined base link. For motion planning, sampling based motion planning has to wait for sets to be properly defined (no movement for any instant) before any planning can be done, and inverse kinematics involves series of complex transformation matrices for objects to move, while the Open Motion Planning Library only requires

the position of the object, and can respond to movements in close to real-time.

5. IMPLEMENTATION

The actual model and motion planning algorithm to be implemented onto the Telebot is URDF modeling with OMPL. As such, implementation of these two components must be considered.

The first aspect of implementation was to connect the model made in URDF with the robot itself. With the model itself, on the Rviz visualization software, the robot is capable of displaying the model as well as controlling it through a built-in graphic user interface(GUI) which controls all moveable joints. Through rviz, a ROS publisher is running which constantly outputs each joint's position (in its own reference frame) to a ROS topic called joint_states. As such, to reflect these changes to the robot, a simple subscriber/publisher node(telebot_pub) has to take this input, convert it into positions that the Telebot understands, and then publish these new coordinates to a node(telebot_sub) which outputs the positions to the actual robot. A diagram of this can be seen in Fig. 4.

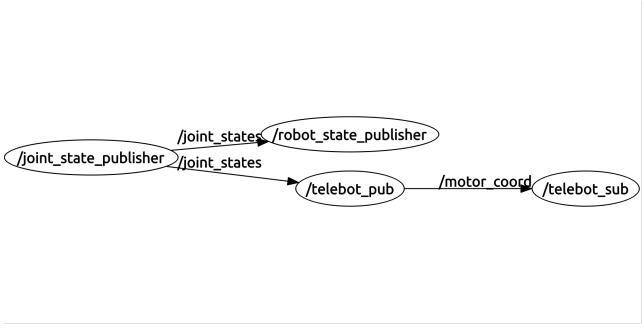

Figure 4. Telebot ROS Node Map

From this, the next step is to implement the OMPL with the URDF model, thereby implementing the OMPL onto the Telebot. This involves creating a package using the URDF which launches files such as controllers, kinematics solvers, and more. ROS has a built in assistant to do this called the MoveIt Setup Assistant which creates all these files for the user with some input through a setup window. After creating this, launching the startup file will open the existing model with a OMPL GUI in Rviz.

However, this Rviz visualization and OMPL GUI do not have the capability to capture real-time images to use as obstacles in the visualization software, instead needing previously created models. As such, some form of capture device needs to be connected and needs to broadcast obstacles in the region of the robot to the visualization software, to allow for motion planning with real-time obstacles.

The final issue to be implemented is some form of statistical analysis as well as a predictability model into the OMPL, as the current library is not capable of predicting future goal positions, instead requiring the user to specifically state where the intended goal position is. Instead, we want more autonomy in the cases of user error, equipment error, or latency, and as such we need the robot to predict for itself where the operator intended any part of it to go. Some form of statistical framework has to be created alongside the OMPL, and needs to be efficient enough that it can predict with minimal latency where the operator intended any particular part of it to go.

6. CONCLUSIONS

This paper has discussed three different modeling methods and planning algorithms: Configuration Spaces with Sampling Based Motion Planning, Denavit-Hartenberg Parameters with Inverse

Kinematics, and Universal Robot Description Format models with the Open Motion Planning Library, and we discussed the difficulties and advantages of them.

We then determined which model and planning algorithm would be best suited for stochastic motion planning for the Telebot, and how to implement said model, which in our case was URDF modeling with the OMPL.

Acknowledgement

This research work is based upon work supported by the National Science Foundation under Grant No. CNS-1263124. The authors thank Professor S. S. Iyengar, who is the founding Director of the Discovery Lab for his support of this project. Thanks to Lt. Cmdr. Jeremy Robins for his initial contribution to the project. The authors like to thank all the individuals in the FIU Discovery Lab for their help and support in making this paper, especially Irvin Cardenas and Shadeh Ferris-Francis.The authors would also like to thank DoD for sponsoring the internship for Yonah Elorza for ten week duration.

References

Canny, J. 1988. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge.

Corke, P. 2007. A simple and systematic approach to assigning denavit hartenberg parameters. *IEEE Transactions on Robotics Vol.23,* No.3, pp.590–594.

Garage, W. 2012. Urdf - unified robot description format.

Hartenberg, R. S. and Denavit., J. 1955. A kinematic notation for lower pair mechanisms based on matrices. *J. Appl. Mech. 77,* 2, pp.215–221.

Kavrak, L. E. and LaValle, S. M. 2008. *Motion Planning.* Springer Handbook of Robotics. Springer, Berlin.

Prabakar, M. and Kim, J. H. 2013. Telebot: Design concept of telepresence robot for law enforcement. *Proceedings of the 2013, World Congress on Advances in Nano, Biomechanics, Robotics, and Energy Research*, 34–42.

Quigley., M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. 2009. Ros: an open-source robot operating system. *Proc. Open-Source Software Workshop Int. Conf. Robotics and Automation, Kobe, Japan*.

Sucan, I. A. and Mark Moll, L. E. K. 2012. The open motion planning library. *IEEE Robotics and Automation Magazine. Vol.19,* 4, pp.72–82.

Yonah Elorza is an undergraduate currently finishing his B.S. in Computer Engineering from Columbia University in New York.

S. Sevugarajan is currently working as a professor at Acharya Institute of Technology, Bangalore, India. He obtained his BSc degree in 1995 with Physics as a major subject and completed his MSc in the field of material Science in 1997 at Anna University. He joined Indian Institute of Science (IISc) at Bangalore in 1997 and there he earned an additional research Master's degree (MSc[Engg]) in Instrumentation Engineering. After submission of his Master's thesis at IISc in 1999, he continued his research work at IISc and obtained his PhD degree in 2005. Immediately after completing his PhD, he Joined Indiana University, USA as a post-doctoral fellow and in 2006 he joined Vanderbilt University, USA as a research associate. At Vanderbilt University he designed and developed a Dual-Source Electrospray/MALDI Ion Mobility-Mass Spectrometer (IM-MS) for Biomolecular Structural Characterization. During 2015 he worked as a visiting faculty at Florida International University, USA, where he worked on the TeleBot project. His research interests are Design and development of Ion-Mobilty Mass Spectormeters, Telepresence Robots, Automated Unmanned vehicles and Deep Learning techniques.

Jerry Miller, USAF (Ret.) is the Research Coordinator at Florida International University's (FIU) Discovery Lab–an undergraduate robotics and autonomous vehicles lab–within the School of Computing and Information Sciences. Col. Miller holds an MS in Telecommunications and Networking from FIU, an MA in Management and Human Resources from Webster University, and a BS in Basic Sciences (Engineering) from the U.S. Air Force Academy. He served for over 27 years in a variety of positions in the United States Air Force, including rescue/special operations helicopter pilot and Foreign Area Officer. He has been at FIU since 2006 and has conducted research as a Principal Investigator in renewable energy and strategic culture studies, and as Co-PI in cybersecurity and privacy of computer networks.