**REPURPOSING A SAMPLING-BASED PLANNER FOR A SIX-DEGREE-OF-FREEDOM MANIPULATOR TO AVOID UNPREDICTABLE OBSTACLES**

**HAFIZ IMAN1\*, MD RAISUDDIN KHAN1**

*1Department of Mechatronics Engineering, Kulliyah of Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia*

*\*Corresponding author: hafiz.ghazman@gmail.com*

***ABSTRACT:*** This paper presents the use of a sampling-based planner as a reactive planning scheme to avoid obstacle between a robotic arm and a moving obstacle. Based on a planner benchmark on a obstacle-ridden environment, rapidly-exploring random tree (RRT) planner are used to populate the trajectories of the task space and map them into a configuration space using Newton-Raphson-based inverse kinematic solver. Two robot poses are defined in a cycle of back-and-forth motion; the initial and the goal poses. The robot repeatedly moves from the starting pose to the end pose via the midpoint pose. Each set of trajectory is unique. We impose periodically occuring synthetic obstacle that moves in and out of the robot arm workspace defined in a simulated environment. Within the robot's workspace, the obstacle moves and cuts through the cyclical space to emulate a dynamic environment. Based on the performance of the reactive planning strategy, our benchmark shows that the RRT has the lowest time planning time at 0.031 s followed by the Probabilistic Roadmap (PRM) at 0.035 s.The average time-to-collision is at 40 s which can be increased with the introduction of obstacle tracking and mapping sensor feedback such as the RGB-D sensor or the LiDAR.

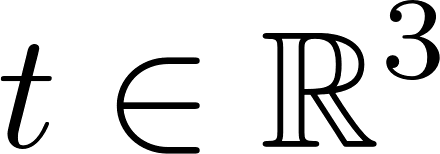
***ABSTRAK:*** Kertas kerja ini membentangkan penggunaan perancang berasaskan persampelan sebagai skim perancangan reaktif untuk mengelakkan halangan antara lengan robot dan halangan yang bergerak. Berdasarkan penanda aras perancang pada persekitaran yang dipenuhi halangan, perancang pokok rawak penerokaan pantas (RRT) digunakan untuk mengisi trajektori ruang tugas dan memetakannya ke dalam ruang konfigurasi menggunakan penyelesai kinematik songsang berasaskan Newton-Raphson. Dua posi (posisi dan orientasi) robot ditakrifkan dalam kitaran gerakan bolak-balik; posi awal dan posi matlamat. Setiap set trajektori adalah unik. Kami mengenakan halangan sintetik yang berlaku secara berkala yang bergerak masuk dan keluar dari ruang kerja lengan robot yang ditakrifkan dalam persekitaran simulasi. Dalam ruang kerja robot, halangan bergerak dan memotong ruang kitaran untuk meniru persekitaran yang dinamik. Berdasarkan prestasi strategi perancangan reaktif, penanda aras kami menunjukkan bahawa RRT mempunyai masa perancangan masa terendah pada 0.031 s diikuti oleh Pelan Hala Tuju Kebarangkalian (PRM) pada 0.035 s. Purata masa untuk perlanggaran adalah pada 40 s yang boleh ditingkatkan dengan pengenalan pengesanan halangan dan maklum balas sensor pemetaan seperti sensor RGB-D atau LiDAR.

***keywords:*** *mechatronics, robot manipulator, planner, motion planning, dynamic environment,*

1. **INTRODUCTION**

Robot manipulators such as industrial robots work well in repetitive and heavy tasks. They are high-performing, objective, and relentless at task that are too difficult to complete by an operator or a group of workers. However, given their rigid and massive construction, even a small-sized industrial robot impose significant hazards on the people that work near it. Hence, recently, robot manipulators are more compliant and are designed to work with workers cooperatively without risking their safety [1]⁠. Regardless, it is still an issue of hazard should a compliant or cooperative robot collide with a person working at close to its workspace [2]⁠.The collision also warrants expensive maintenance and repairs. To address a more intelligent motion and, evidently, a safer motion planning, an industrial robot system implements a certain degree of planning algorithm specifically for the motion of the manipulator.

A robot motion planner provides a collision-free motion solution for a manipulator. Here, a solution is defined as collections of waypoints or trajectories. In the case of the traditional definition of industrial robots, the planning is global because the robot is enclosed and isolated in workcell. A global planner takes in a set of initial and goal positions, , or as set of initial and goal poses, , as its input and generates constraint-informed trajectory as intermediate waypoints for the robot to follow. However, a global planner is offline, which implies that the trajectories are set before the task commences. The global planner also assumes a static workspace. Any unplanned changes in the workspace over the global planning-scheme, such as an unplanned introduction of a stationary object or a moving object into the robot workspace, renders the offline-planned trajectory outdated and, consequently, requires replanning. In the case of a compliant robot, unplanned changes are unavoidable.



Hence, it is imparative for a compliant manipulator or cooperative manipulator to have an efficient motion planner because replanning is computationally expensive and time-consuming; sampling-based planners are the pinnacle example of efficient planning [3]⁠. Unfortunately, sampling-based planner trade completeness and optimality with efficiency where the planner may fail to provide a solution [4]⁠. Also, if a solution exists under its metric space, the waypoints may not be the least cost path to a goal [5]⁠.

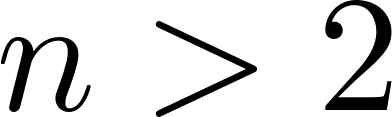
Regardless of lack of completeness and optimality, sampling-based planner excel at maintaining reasonable usage of computational resources that pave way to the near-online planning scheme. The sampling-based planners for robot motion are a family of planners that uses probabilistic approach to generate a graph structure encoding the free space and the robot configuration space. The sampling are stochastic, such that resampling will give a unique solution to the previous sampling. Most sampling-based planner are also tractable in higher-dimensional configuration and task space. However, sampling-based planners assume static workspace.

This paper repurposes the use of sampling-based motion planning, the rapidly-exploring random tree motion planning, to address operation task in a dynamic environment in Euclidean space, . Our method leverage the efficiency and the computationally reserved sampling-based motion planning without needing to apply purely reactive motion planning approach so that computational resources can be delegated to other tasks, i.e., motion-tracking, state estimation, mapping, localization, and motion control. In the following of this report, we will assume sampling planners provide solution in higher dimensional configuration space, which implicates a solution with a set of poses represented by the special Euclidean group, . This paper is a part of a collision avoidance system for a compliant robot manipulator design to prove an encoder-less concept, and only cover the formulation of the planning approach we adopt for our robotic system.



1. **RELATED WORK**

Kavraki et. al (1998) is the first group of researchers that used probability model for sampling the configuration space for holonomic robot motion such as a manipulator robot [4]⁠. The planner are called the probabilistic roadmap (PRM) motion planning. The algorithm construct a graph structure to find path between an initial pose to a goal pose in two-dimensional configuration space, . Kavraki et. al (1998) also proof a more general solution for higher dimensional configuration space, . With graph structure, more than one path connect the initial pose to the goal pose. Therefore, PRM is a multi-query type planner.



Kunz et. al (2010) improve PRM by redefining the distance metric of a robot manipulator so that the robot can move around a moving obstacle in real-time. Their approach performs well in an uncluttered environment [6]⁠. They also redefined the distance function of the PRM to address dynamic objects, such as a walking person, into a two-dimensional map. Although the configuration space of the manipulator is in , the map, constructed from a two-dimensional LiDAR scan, is in, .



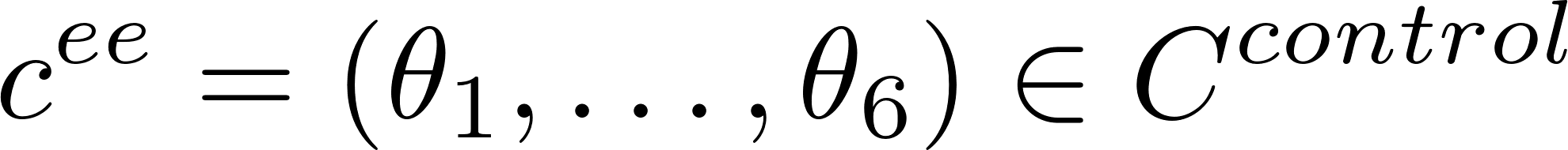
In retrospect, the RRT was formulated for non-holonomic motion targeting problems addressed in diffential-constrained motion such as a car on a plane [5]⁠. However, given the model of its metric space and consequently the configuration space, RRT are tractable for higher dimensional problem such as manipulator motion in 3D space [7]⁠. RRT assume as static environment but Wei & Ren (2018) successfully change the way RRT samples a robot metric space so that it is fast enough to react with a changing environment. Also, unlike PRM, RRT works well in a cluttered environment because of the randomized sampling on the robot configuration space in the metric space.

With the exception of the works in [6]⁠ and [7]⁠, so few have applied their planning algorithms on a robot manipulator despite both algorithms provides mathematical framework for planning for multi-body and multi-frame system. In this paper, we will use the method demonstrated by Kunz et. al (2010) and Wei & Ren (2018) to design our experiment of a moving obstacle avoidance with the implementation of the vanilla RRT to solve motion for robot manipulator in three-dimensional space, . Different from the implementation by Wei & Ren (2018), however, our method implement the vanilla RRT where we do not represent the obstacle configuration space.



1. **FORMULATION AND ALGORITHMS**

This paper will use the superscript notation to refer the control space and the subscript as the equivalent representation in the configuration space. For example, , refers to the control space of the end-effector where the controlling pipelines would take in , and the equivalent pose is in the configuration space, . Since, revolute joint topology is the 1-hypersphere, , we will assume that, for the case of 6R robot, it's joints are limited to a certain range which makes, .



* 1. **The Geometry of a Compliant Robotic Arm, r\_mini**

We prototype and build a 3D-printed robot called Richard Mini (*r\_mini*, see Fig. 1) based on the conditioned addressed by Pieper (1968) [8]⁠, which entails three collated joints sharing the same cross point of their, , shown in Fig. 2.



|  |  |
| --- | --- |
| (a) *r\_mini* hardware assemblage | (b) r\_mini CAD design |
| Fig. 1: A 3D printed compliant manipulator, *r\_mini*, are designed to replicate a common industrial robot construction | |

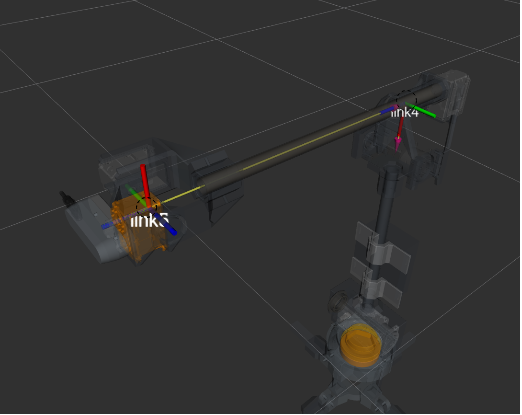
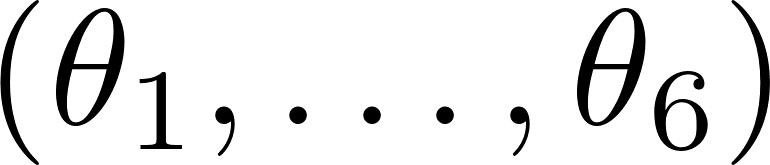
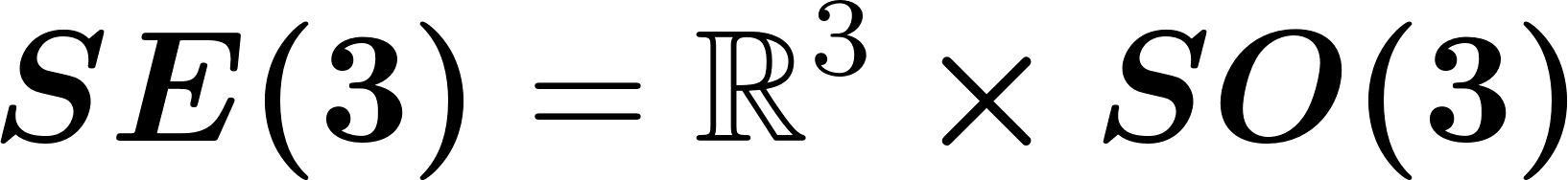
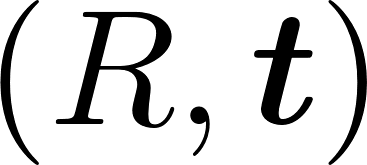
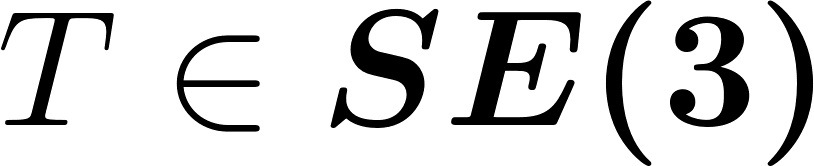
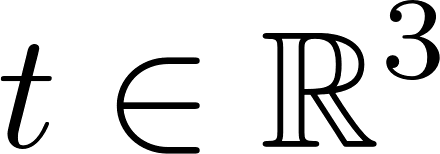
****

Fig. 2: *r\_mini* wrist conforms to Pieper condition where axis of rotation for joint4, joint5, and joint6 share points of intercept. The dashed circles in the diagram refer to possible point of intercepts. Both points are valid frames for constructing the DH-table.

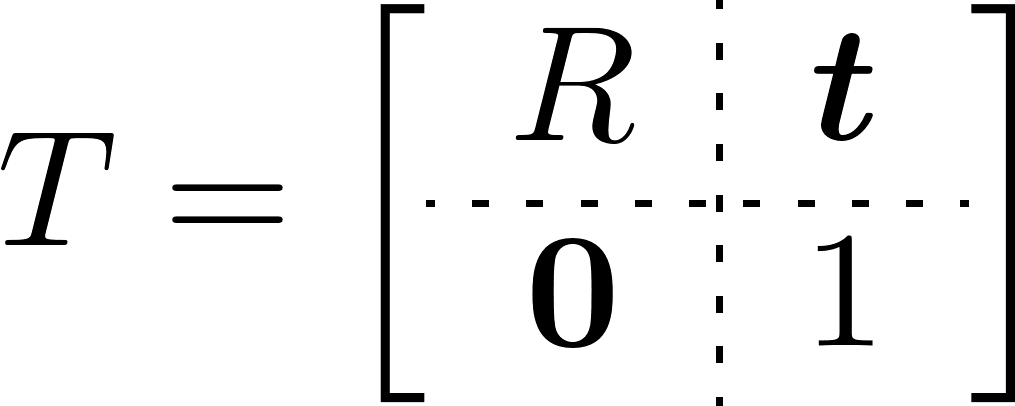
*r\_mini* has six revolute axes, . The first three axes moves the task space from one



point to another representing translation vector, . The last three axes of the manipulator rotate the task space representing the rotation operation about the task space frame, . Hence the complete transformation of the task space via the joint movement is represented by the homogenous transformation matrix, , where is homomorphic to ; . The matrix representation of the homogenous transformation is shown in Eq. (1).

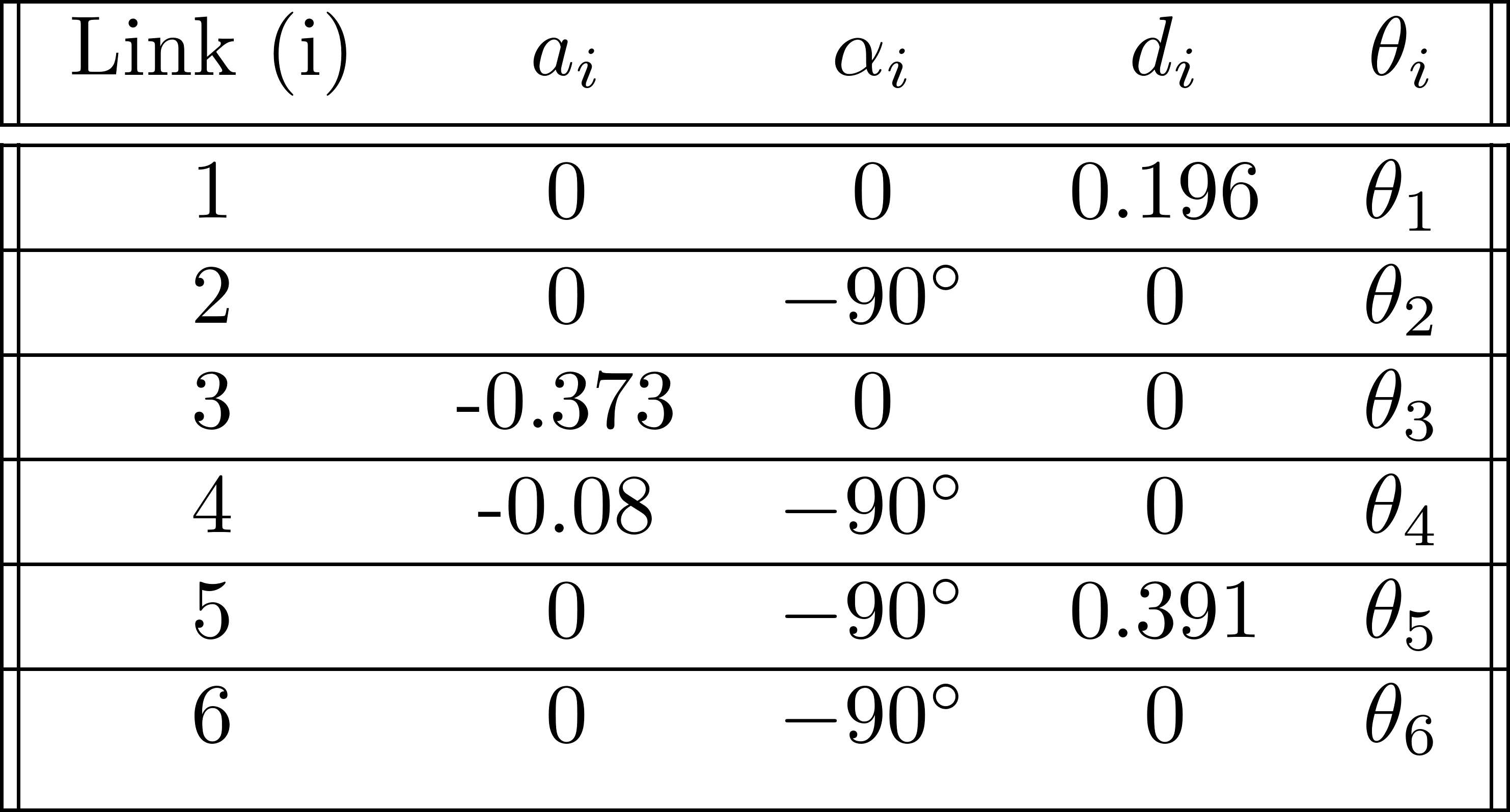


(1)



The kinematic models of the *r\_mini* follows the Denavit-Hartenberg (DH) formulation [9]⁠. The DH-parameters are shown in Table 1 and the visualization of these parameters in the form of frames transformation is shown in Fig. (3).

Table 1: DH-parameter table for *r\_mini*

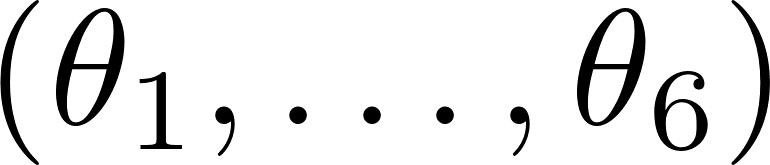
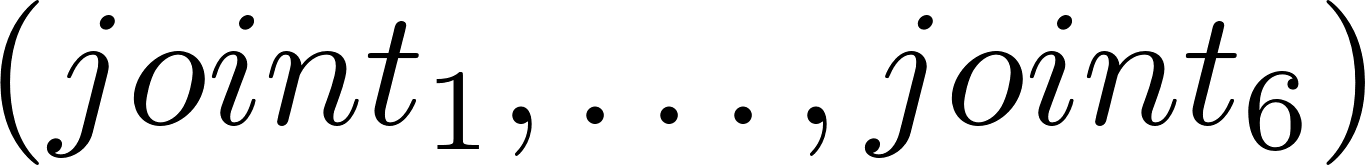


|  |  |  |  |
| --- | --- | --- | --- |
| (a) Frame 1 | (b) Frame 2 sharing origin with Frame 1 | (c) Frame 5 | (d) Frame 6 sharing origin with frame 5 |
| (e) *r\_mini* build completion | | | |

Figure 3: The location and orientation of *r\_mini*. The choice of the orientation for each frames is based on Denavit-Hartenberg convention. The joints’s values are represented by the angle between around the (the rotation axis of each joints) of the actuator



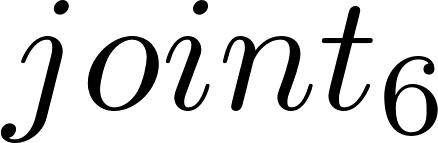
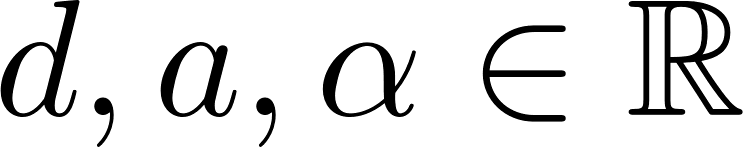
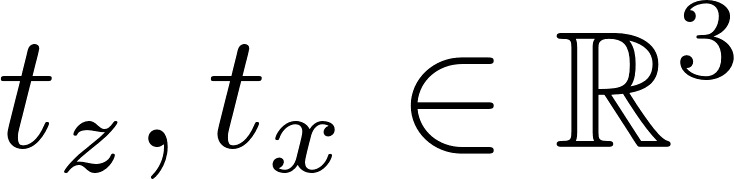
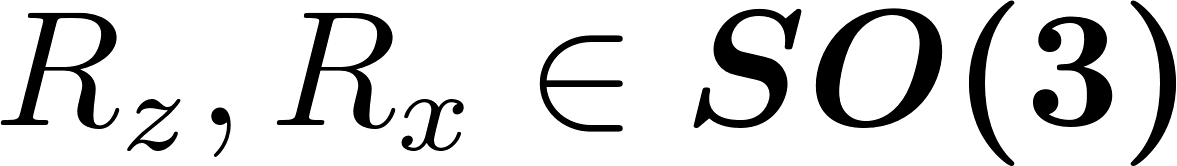
Following the DH-convention, each links rotates about the of each frames it is attached to, where , corresponds to respectively. In Table 1, each rows represents homogenous transformation, Eq. 2:



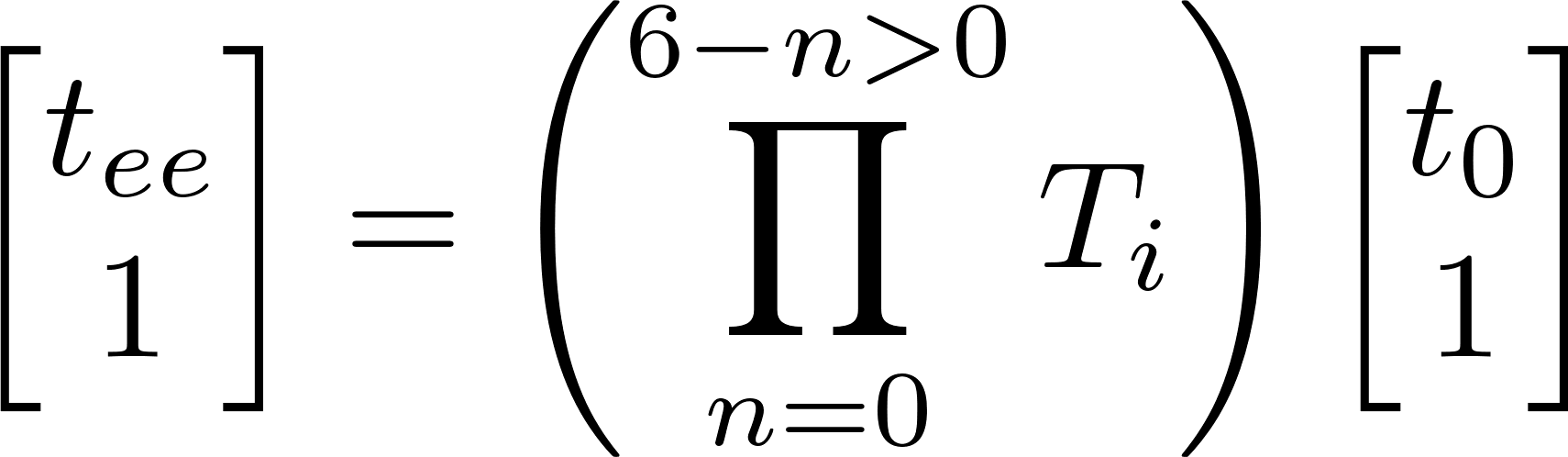
(2)



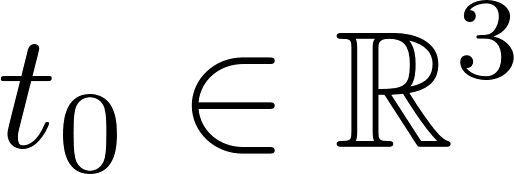
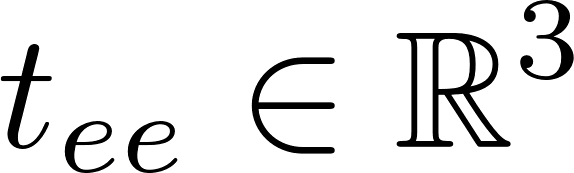
where, , are the rotation operations about the and the respectively, and are the translation vector on the and the respectively, while, , are scalars. The homogenours transformation between the origin of the base of the robot into the end-effector of the robot, which coincide with is shown in Eq. (3),



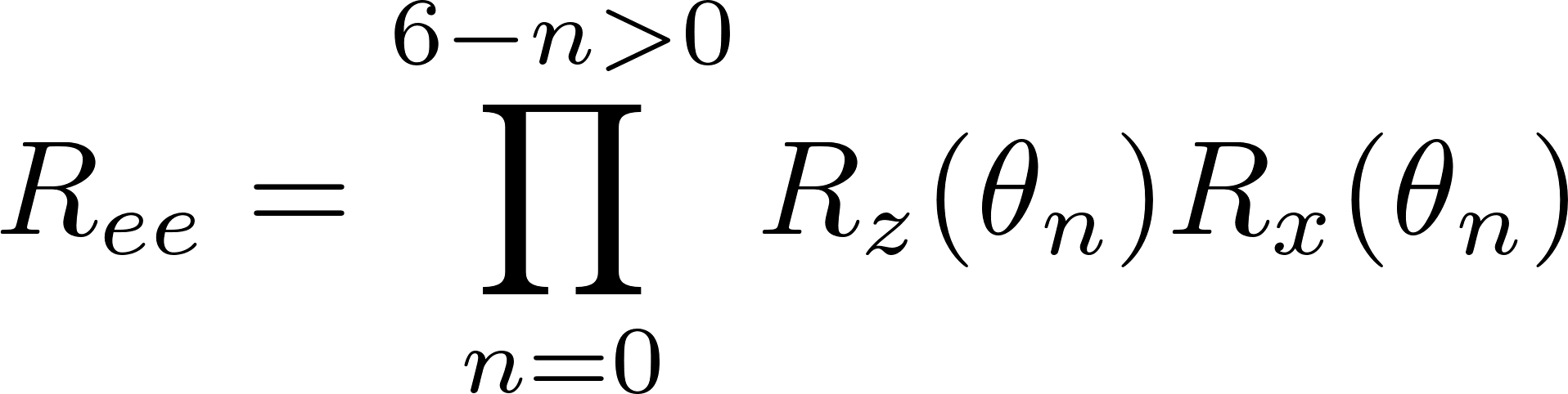
(3)



where , is the point location of the end effector in 3D space, and is the origin of the base of the robot. Since the rotation involve in Eq. 3 includes the rotation about the origin of the local frames, the orientation of the end-effector can be represented by,

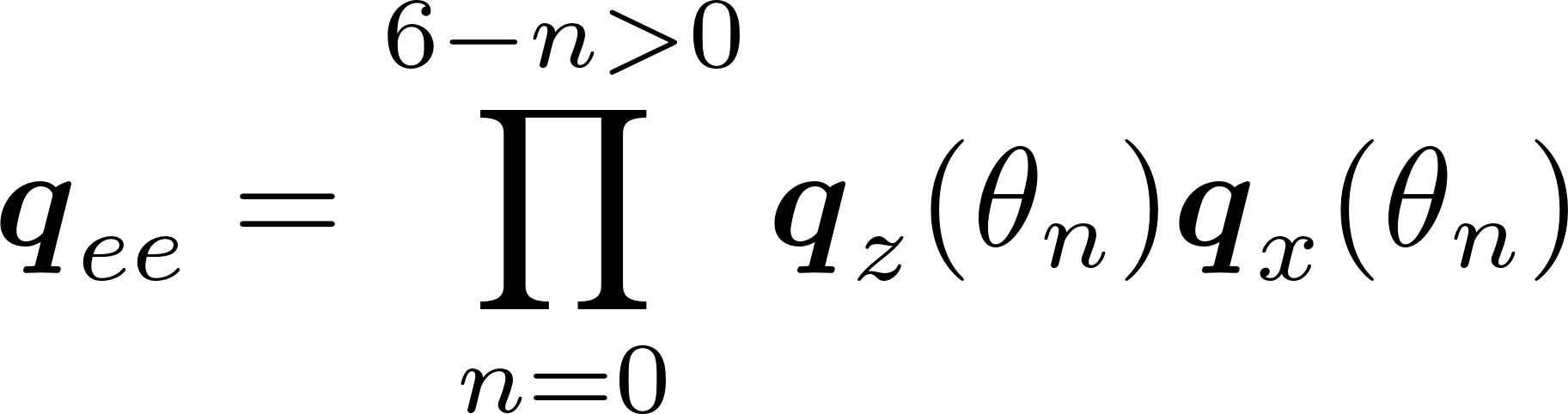


(4)



Here the operation is closed. Often, to reduce computation expenses and trailing errors due to matrix-matrix multiplication, the rotation operation of *r\_mini* are done over the quaternion (Eq. (5)),

(5)



where the Eq. (2) and Eq. (3) represent the forward kinematic solution for the end-effector of *r\_mini*.

The self collision, robot collision checking is delegated to a collision and proximity query library, the Flexible Collision library (FCL). Later in the algorithm formulation of the RRT and the cycle space, subroutine from the FCL will be invoke to check collisions between the manipulator and the moving obstacles. The robot manipulator and the obstacles are encoded inside the collision scene which reside in the planning scene () when the RRT datastructure is initialized (refer to Algorihtm 2 line 1).



We use Newton-Raphson method to find the inverse kinematic solution of *r\_mini*, . The generalization of the method uses the current value of the robot's encoder, , and the termination value, , to end the iteration. Algorithm 1 delineate the the method:

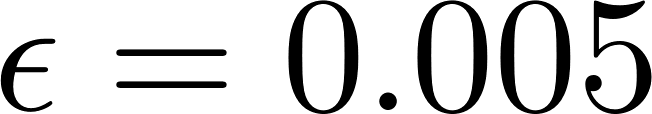
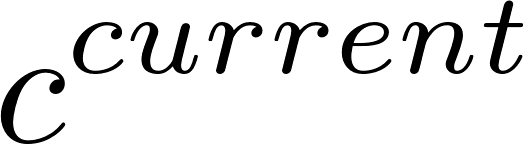
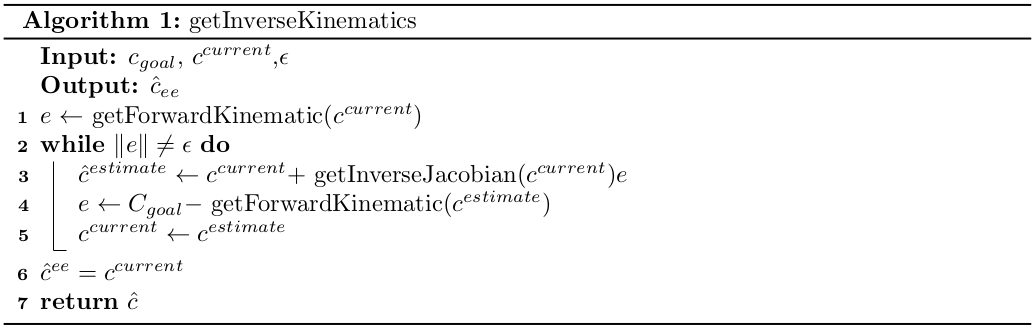


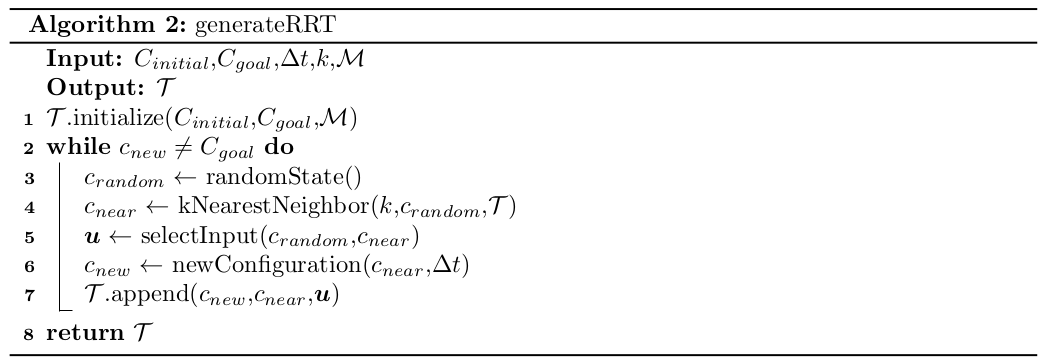
Table 2: The Newton-Raphson algorithm for *r\_mini*’s inverse kinematic solver



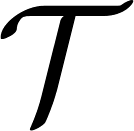
* 1. **The Rapidly-Exploring Random Trees and its Mathematical Background**

This research uses RRT implementation provided by OMPL library packaged in the MoveIt software. The algorithm for the purpose of this research is shown in Algorithm 2:

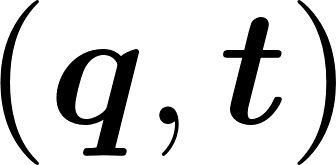
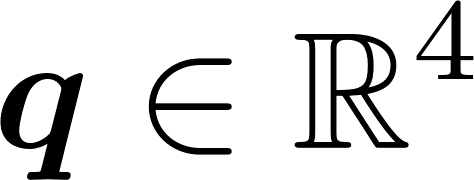
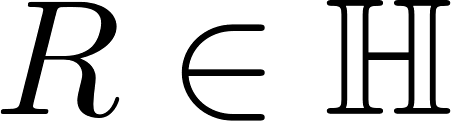
Table 3: The RRT algorithm



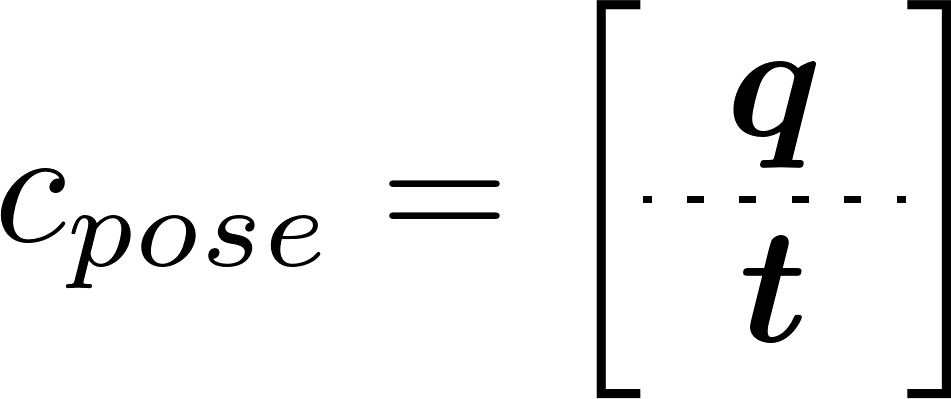
where represent the, number of node in the tree generated by the RRT, , represent the collision map which is part of the planning scene where all RRT sampling takes place and, , is the tree that points to a non-colliding space. In this RRT implementation, the map are loaded or queried in line 1 each time the *generateRRT()* is invoked. Line 3 generates a random state bias towards the . Line 4 invokes the k-nearest neighbor to find a selection of nodes that is closes to the state configuration, . Line 5 is the important part of the sampling in the RRT where it represent the controlling input of the robot motion. Since, the robot are control in the joint-configuration space, the angular joint limit address the shape of the workspace. However, given the angular velocity, these limits are translated into the configuration space via the kinematic Jacobian which requires the information on the . The limits implicitly ensures that the RRT, by executing Line 5 within the context of the robots Jacobian, does not pass through the singularities of the robot. Hence, the configuration space of the manipulator also includes, , containing configuration that abides the joint-configuration space range and angular velocity limit.



The configuration space where sampling occurs is modified in this paper where, the rotation representation and its sampling is in , such that the parameterization of the Hamiltonian-space is the quaternions, . Therefore, the representation of the robot poses and also the non-colliding poses, , are explained in Eq. 6.



(6)



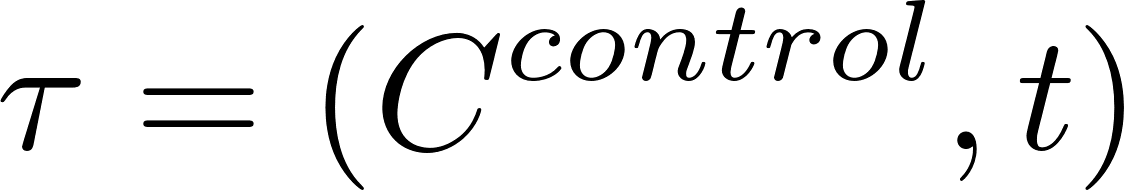
The RRT sampling involves query into a map, that stores collision objects. This is the planninq scene, denoted as collision map, where the RRT sampling occurs. The query are called when both initial pose and a goal pose are sent to the RRT planner input. The output of the pipeline is a set of non-colliding space where from there another pipeline, transform the configuration space into a control space. We will define the control space in the following section.

* 1. **The Cyclical Space**

The cyclical space is the subset of the planner solution where the RRT algorithm is invoke twice. During the generation of the cyclical space, the RRT output the a trajectory from the initial pose, to the goal pose, , into a controlling pipeline. The trajectory are then map from the configuration space into the joint-configuration space via the Newton-Raphson inverse kinematic solver (see algorithm 1). To complete the set of the cyclical space, the entries in the initial pose and the goal pose are swapped, while invoking query into the collision map,



(7)



which forms a cyclical motion between the initial pose and the goal pose. Here, .



Algorithm 3 block explains how the space is constructed.



The control space are represented by the trajectory in the joint-configuration space , where is the 6-hypersphere homomorphic to because each joints are constrained to its angle limit. In Eq. (7), the joint-configuration space are equivalent with the configuration space in Eq. (6). The control space is the direct controlling parameters for the movement of *r\_mini* where it only handles the control space (or joint-state space). The sampling of the RRT to generate the tree data structure, , are done within the topology. The free configuration space, or the non-colliding pose, are represented by, . According to LaValle et. al (1998), the also covers the physical contraint of the non-holonomic movement of the robot. However, in the case of an articulated robot arm in this research, the configuration limitation are the range of the joints and the angular velocity limit. Since, all of these measurements are in the 6-space, to map them into the , we use the kinematic Jacobian.

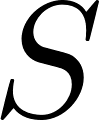
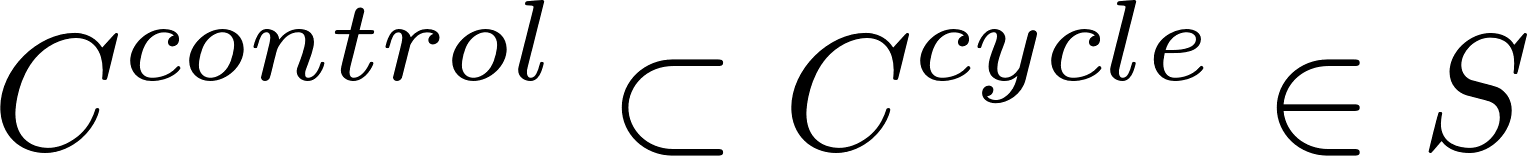
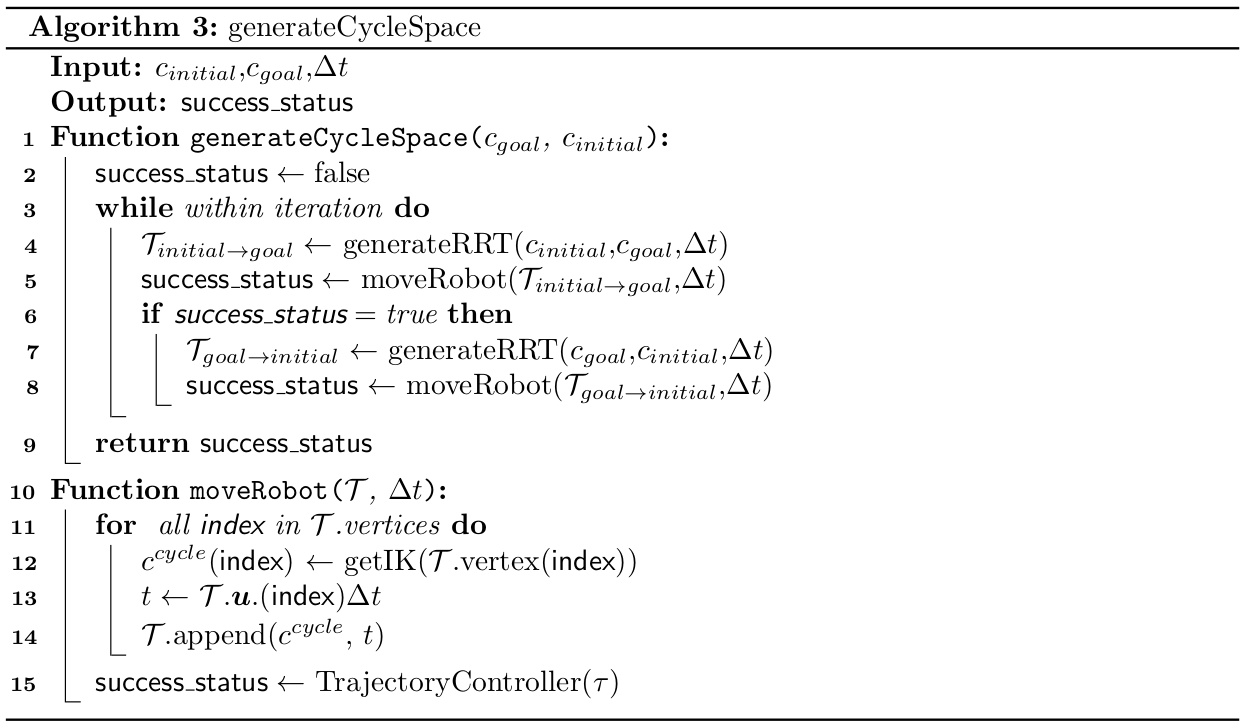


Table 4: The cycle space generator where the movement within the constraint of the cycle space (also a cyclical space) is dependent on the map, .





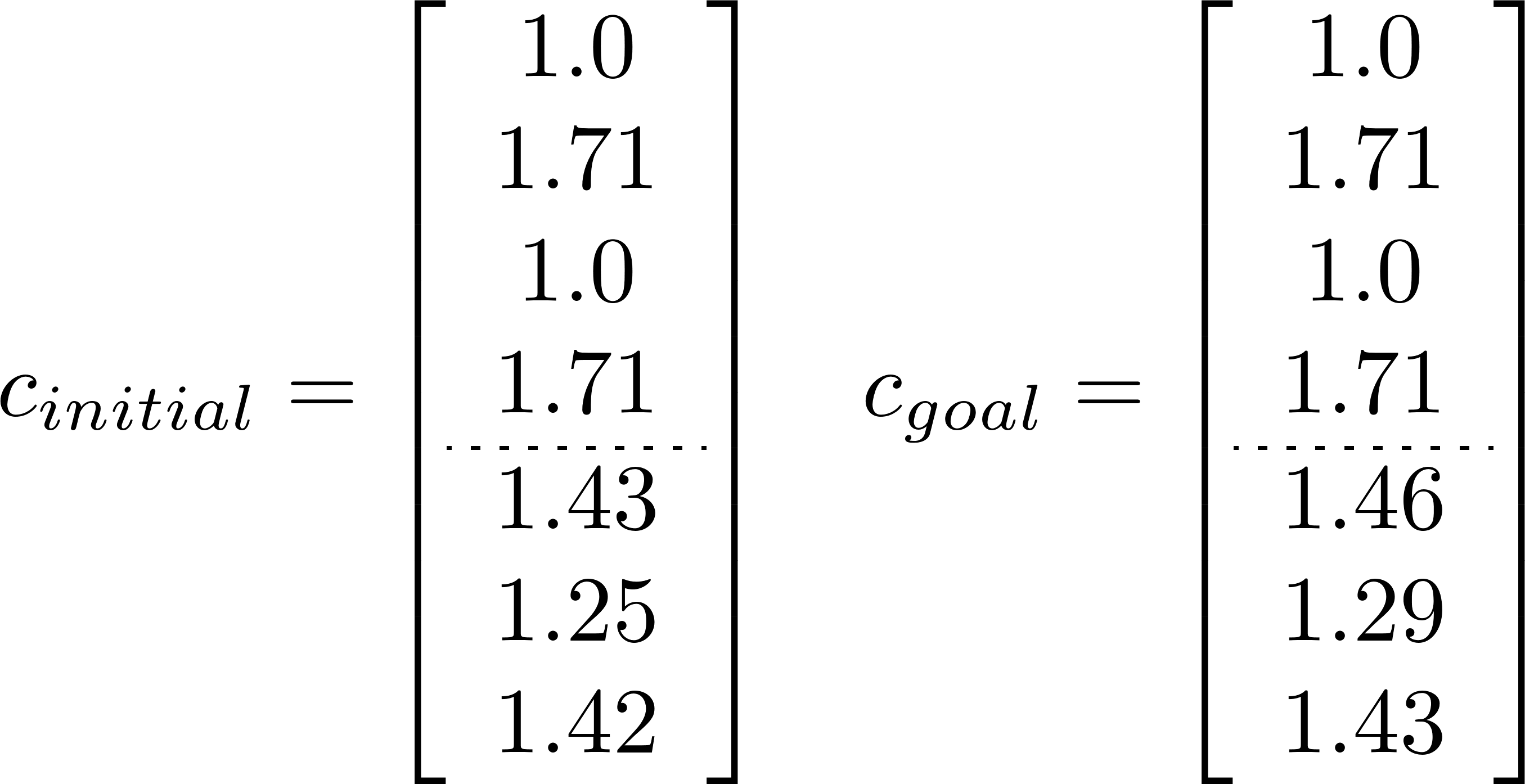
1. **METHODOLOGY**

The methodology starts with the benchmarking of sampling-based planners available in the OMPL library and comparing the performance of the planners with the RRT. The benchmark is followed by experimentation in the simulated environment with a simulated robotic arm (*r\_mini*) followed by the coupling of the simulated environment with *r\_mini* hardware. The experimentation involves the moving obstacle that is introduced synthetically the in the collision space. This is necessary since, at the time of this experimentation, the feedback from the mapping sensor are unavailable.

* 1. **Benchmarking of Sampling-Based Motion Planners**

In this research, the planner for the dynamic obstacle avoidance are selected based on the performance of a benchmarking activity. Here, the procedure is explained. Two poses are set for the benchmark, pose initial are represented in the form of Eq. (6). The following vectors explain the numerical value of these poses with respect to the frame attached to the base of *r\_mini*.

(8)



A box, with dimension, 0.5 m 0.05 m 0.575 m, are place in front of the robot, it's pose is described by the vector in Eq. (9),



(9)

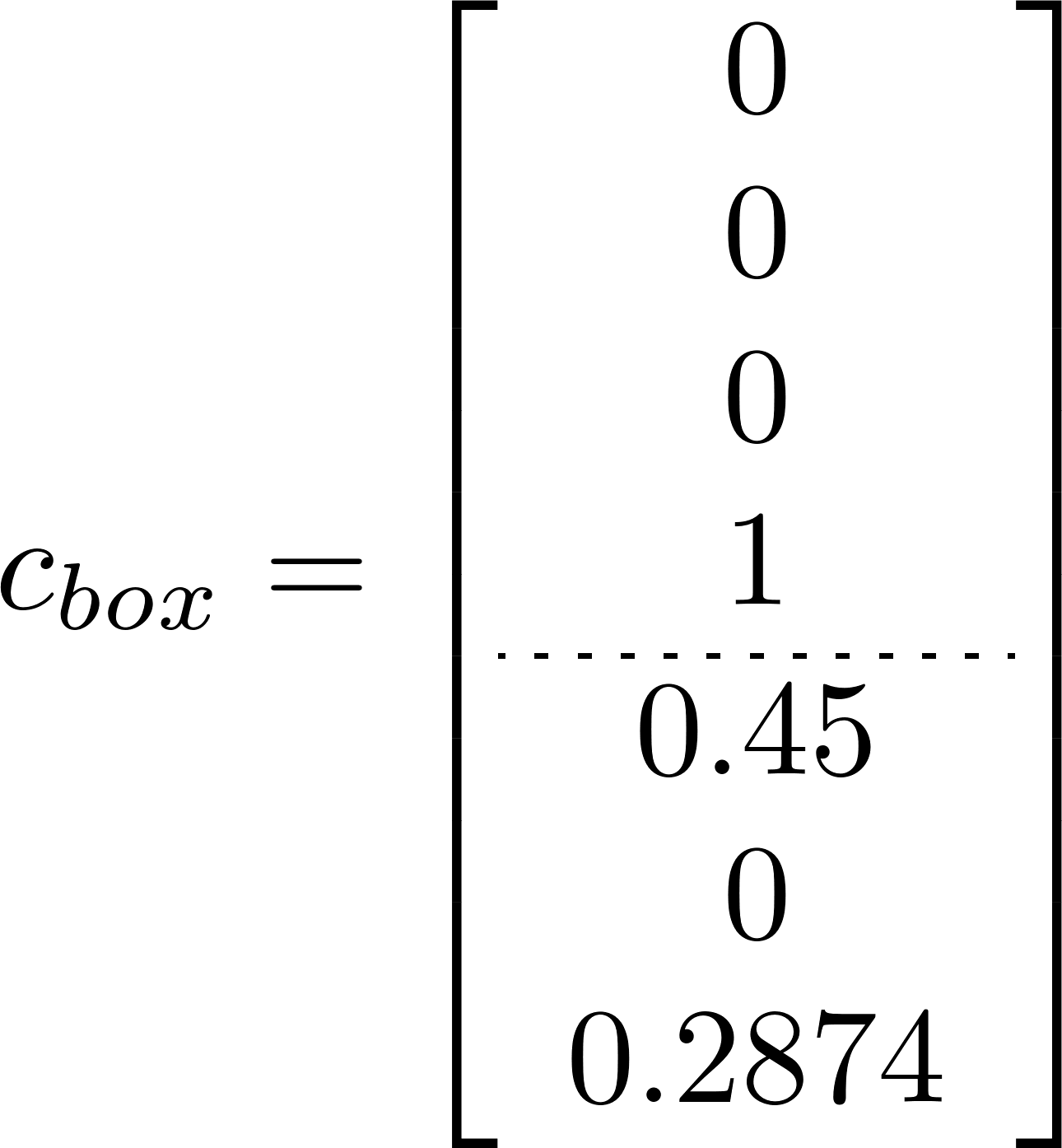


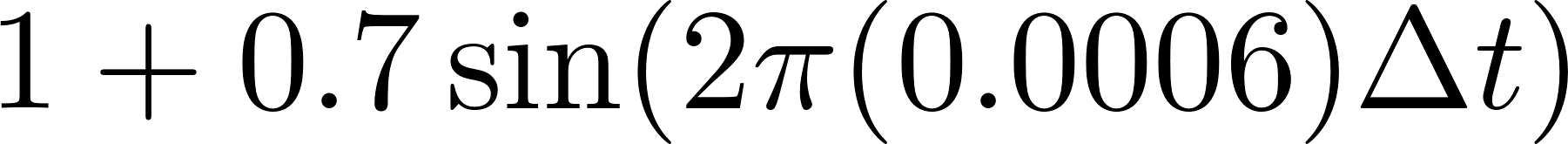
Fig. (4) shows the simulation setup and the planning motion in action. The simulation is run for 50 request from the initial pose to the goal pose. Time processing is given a 10 s limit. The memory limit is set to 1 Mb. The time limit for a request, including the motion and the processing time is set to 3637 s. This paper use these configurations and the default configuration of each planners in the MoveIt to start the benchmarking.

|  |
| --- |
|  |
|  |
|  |

Fig. 4: The top view of the simulation shown in (top), and the isometric view of the benchmark setup (middle). *r\_mini* attempts to move around the static obstacle placed in it’s immediate configuration workspace (bottom).

* 1. **Experiment Design**

The cyclical space is populated by the RRT-Newton-Raphson pipeline where the generated trajectories are then pass to the control pipeline where the controller will spline the sparse trajectory waypoints. Two poses are defined in this experimentation which has been described in Eq. (8). A moving obstacle are placed in front-view of the robot. The obstacle is a cylinder with 0.1 m radius base at 1 m height. The obstacle moves from 0.3 m to 1.7 m away from the robot in oscillation. The period of motion is harmonic, such that, the robot follows along the . Two velocities () values were used: 50% and 10% scale from the maximum velocity of the end-effector.



The planner is invoked five seconds before the obstacle is placed into the planning scene. As described previously, the cylinder are directly place into the planning scene (i.e. collision space) such that no motion tracking via mapping sensor feedback is necessary for this research. The planner are requested to provide solution for the motion described by the cycle space. Twenty iterations are done with each given a five minutes runtime. The metric use for this experiment is the time on first collision where,when the prototype touches the cylinder, the iteration is terminated. This experimentation is done, both, in simulation, and with the real robot hardware coupled with the simulated environment. To reiterate, for both the simulation and the hardware validation, the obstacle is augmented in simulated environment.

1. **RESULT AND DISCUSSION**

There are two part of the result on this paper, the first dealing with the benchmarking result to ascertain the best sampling-based planner. The second part delve into the performance of the selected planner from the benchmarking on a moving obstacle.

* 1. **Benchmark Result**

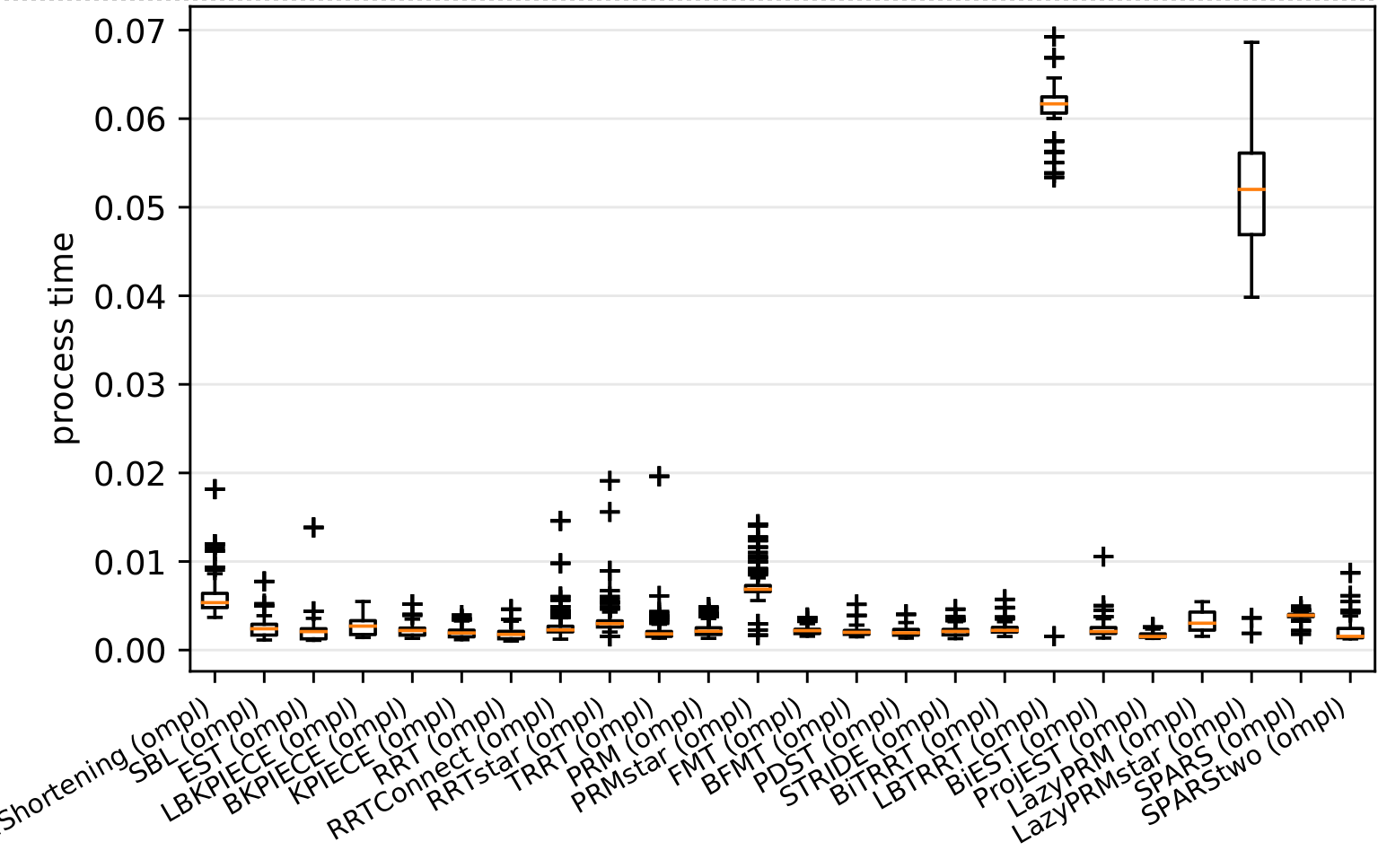
Fig. 5: The benchmark result when two configurations are defined and pass to the OMPL planner pipeline. All planners completed a 50-cycle query from an initial pose to a goal pose. RRT requred the least amount of processing time at finding the motion planning solution, followed by the PRM.

Fig. (5) shows the compiled statistics of the time the solution were pass to the controller (in this case a fake controller for simulation of *r\_mini* in the simulated environment). RRT requires on average, 0.031 planning time while PRM requires 0.035 planning time from the initial pose to the goal pose when subjected to an obstacle very close to the robot. Wei & Ren (2018) explained the improved RRT algorithms, such as the bi-RRT, and the RRT-connect, solve a query faster [7]⁠. However, based on our benchmarking and in the case of this experimentation setup, vanilla RRT, or base-RRT, and PRM outperform their improved variants when completing the path query between an initial pose and a goal pose. To that end, this research uses vanilla RRT as the scheme for the high-level local planner. This result helps us select the motion planner for the dynamic obstacle avoidance.

* 1. **The Performance of RRT on a Dynamic Environment**

Table 5: The simulated and hardware-connected result of the performance of RRT in a dynamic environment. NC stands for *No Collision* after five minute runtime

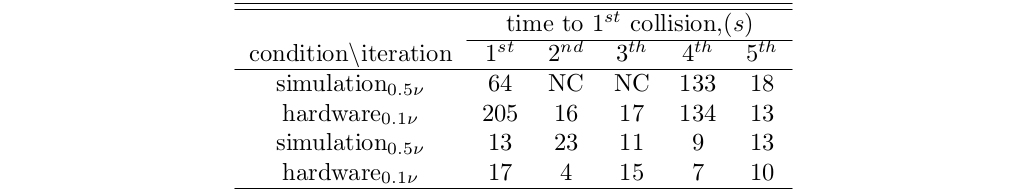


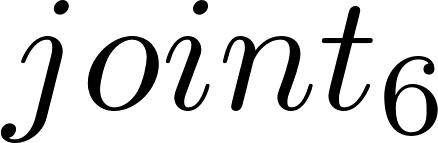
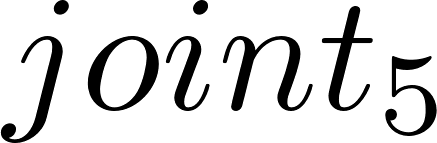
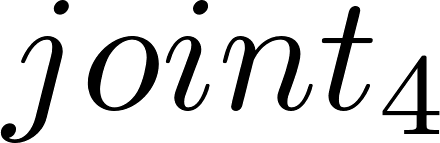
Table 5 shows the recoded time to collision of 20 iterations. The average time to collision is 40 s. There are two iterations where there are no collision recorded. This performance is subjected to the algorithm 3, specifically in line 4 and line 7, when RRT is invoked. Within this call (refer Algorithm 2: line 1) the random tree initilization consider an obstacle map that is outdated given the cyclinder has moving further towards the manipulator when the RRT is executed. Within the RRT algorithm, there are no mechanism for the robot to stop or move at a lower rate to avoid the cylinder. Fig. 6 shows the sequence when the end-effector collide with the cylinder.

Despite the obstacle avoidance fails when the moving cylinder approaches the robot specifically when the centroid of the cylinder is nearing the of the and , the planner successfully avoid the obstacles when the lines 4 and 7 in Algorithm 3 is invoked.



The planner shows reactive behavior when the cyclical space is initialize, via Algorithm 3. Fig. (7) illustrates such behavior in the simulated environment, and Fig. (8) shows the same behavior in the hardware reiteration of the experimentation. The reactive behavior is illustrated in Fig. (9), where and change the range of their movements while changes the rate of its movement.

No significant changes are observed for , and . This is the implication of the Pieper-condition manipulator design where, none of the from the first three joints shares the same crossing point, which suggest the actuation on these joints are not a linear transformation as the case for affine translation. Due to the offset (affine transformation) of the joints' axis of rotation, there is a bijection mapping of these joints to the task-space specifically reserved for translation changes in space. Also changes are also observed on the orientation of the frame attached to the end-effector, however, there are no bijection mapping of the three joints to the task-space's orientation.



|  |  |
| --- | --- |
| (a) | (b) |
| (c) | (d) |
| (e) | (f) |

Fig. 6: These sequences show the manipulator follows an outdated trajectory and collides with the cylinder despite attempt to move away from the moving cylinder.

|  |  |
| --- | --- |
| (a) | (b) |
| (c) | (d) |

Fig. 7: The chronology of the attempt at avoiding a moving obstacle when the obstacle approaches the robot seen in (a) and (b). The planner succesfully provide a non-colliding solution when the cylinder is moving away from the robot (c). This experimentation is defined in a simulated setup using Gazebo with the ODE physic engine to replicate the robot hardware and encoders feedback and the cyclical space initialization (d).

|  |  |
| --- | --- |
| (a) | (b) |
| (c) | (d) |
| (e) | (f) |

Fig. 8: The sequence of motion when *r\_mini* successfully avoid a moving obstacle when the obstacle is at a turning point to move away from the hardware.

|  |  |
| --- | --- |
| (a) | (b) |
| (c) | (d) |

Fig. 9: Reactions from joints 1, 2, and 3, show that the planner, together with the cycle space, behave reactively towards the moving object. No rapid movement or rate on joints 4, 5, 6 are observed.

1. **CONCLUSION AND RECOMMENDATION**

We conclude that, the RRT on a dynamic setup is capable at reacting to an obstacle when the obstacle is moving. The average time-to-collision is 40 s. To increease the time-to-collision, we proposed a obstacle tracking pipeline via mapping sensors such as an RGB-D sensor or a LiDAR. It is also recommended that more than one intermediate poses between the initial pose and the goal pose is defined that has the capability of reacting with the environment, specifically, the obstacle.

**BIBLIOGRAPHY**

[1] M. Hägele, K. Nilsson, J. N. Pires, and R. Bischoff, “Industrial robotics,” in *Springer Handbook of Robotics*, Springer International Publishing, 2016, pp. 1385–1421.

[2] B. Matthias, S. Kock, H. Jerregard, M. Källman, and I. Lundberg, “Safety of collaborative industrial robots: Certification possibilities for a collaborative assembly robot concept,” in *Proceedings - 2011 IEEE International Symposium on Assembly and Manufacturing, ISAM 2011*, 2011, no. 230902, doi: 10.1109/ISAM.2011.5942307.

[3] M. Elbanhawi and M. Simic, “Sampling-based robot motion planning: A review,” *IEEE Access*, vol. 2. Institute of Electrical and Electronics Engineers Inc., pp. 56–77, 2014, doi: 10.1109/ACCESS.2014.2302442.

[4] L. E. Kavraki, M. N. Kolountzakis, and J. C. Latombe, “Analysis of probabilistic roadmaps for path planning,” *IEEE Trans. Robot. Autom.*, vol. 14, no. 1, pp. 166–171, 1998, doi: 10.1109/70.660866.

[5] S. M. LaValle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” 1998. Accessed: Sep. 25, 2022. [Online]. Available: https://www.cs.csustan.edu/~xliang/Courses/CS4710-21S/Papers/06 RRT.pdf.

[6] T. Kunz, U. Reiser, M. Stilman, and A. Verl, “Real-time path planning for a robot arm in changing environments,” *IEEE/RSJ 2010 Int. Conf. Intell. Robot. Syst. IROS 2010 - Conf. Proc.*, pp. 5906–5911, 2010, doi: 10.1109/IROS.2010.5653275.

[7] K. Wei and B. Ren, “A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm,” *Sensors (Switzerland)*, vol. 18, no. 2, 2018, doi: 10.3390/s18020571.

[8] D. L. Pieper, “The Kinematics of Manipulators Under Computer Control,” Stanford University, 1968.

[9] J. Denavit and R. S. Hartenberg, “A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices,” *J. Appl. Mech.*, vol. 22, no. 2, pp. 215–221, 1955, doi: 10.1115/1.4011045.