CHAPTER 1

EXPERIMENTATION, RESULT AND DISCUSSION

1.1 BENCHMARKING EXPERIMENT DESIGN AND RESULT ON THE SAMPLING-BASED PLANNER IN STATIC ENVIRONMENT

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 represented in the form of *equation* **??**. The following vectors explain the numerical value of these poses with respect to the frame attached to the base of *r_mini*.

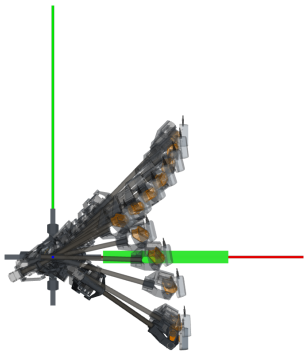
$$c_{initial} = \begin{bmatrix} 0.0 \\ 0.71 \\ 0.0 \\ 0.71 \\ \hdashline 0.43 \\ 0.25 \\ 0.42 \end{bmatrix} \quad c_{goal} = \begin{bmatrix} 0.0 \\ 0.71 \\ 0.0 \\ 0.71 \\ \hdashline 0.46 \\ 0.29 \\ 0.43 \end{bmatrix} \tag{1.1}$$

A box, with dimension, 0.5 m $\times$ 0.05 m $\times$ 0.575 m, are place in front of the robot, it's pose is described by the vector,

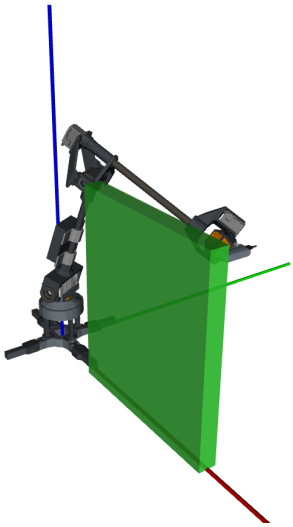
$$c_{box} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \hdashline 0.45 \\ 0 \\ 0.2874 \end{bmatrix}$$

Figure 1.1 shows the simulation setup and the planning motion in action. The simulation ran 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 thesis uses these configurations and the default configuration of each planners in the MoveIt to start the benchmarking.

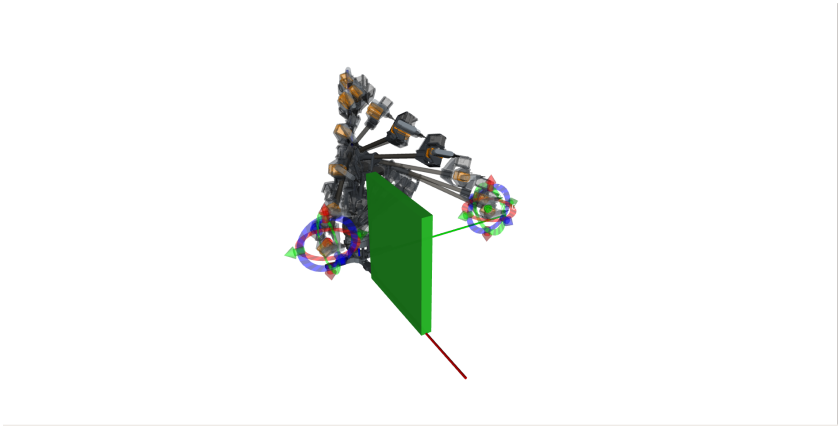
Figure 1.2 shows the compiled statistics of the time the solution were passed to the controller (in this case a virtual controller for simulation of *r_mini* in the simulated environment). RRT requires on average, 0.031 planning time while PRM requires 0.035 planning time from the initial pose to the goal pose when subjected to an obstacle very close to the robot. **Wei2018** explained the improved RRT algorithms, such as the bi-RRT, and the RRT-connect, solve a query faster. However, based on our benchmarking,



(a)



(b)



(c)

Figure 1.1: The top view of the simulation shown in ,(a) , and the isometric view of the benchmark setup in (b). In (c) *r_mini* attempts to move around the static obstacle placed in it's immediate configuration workspace.

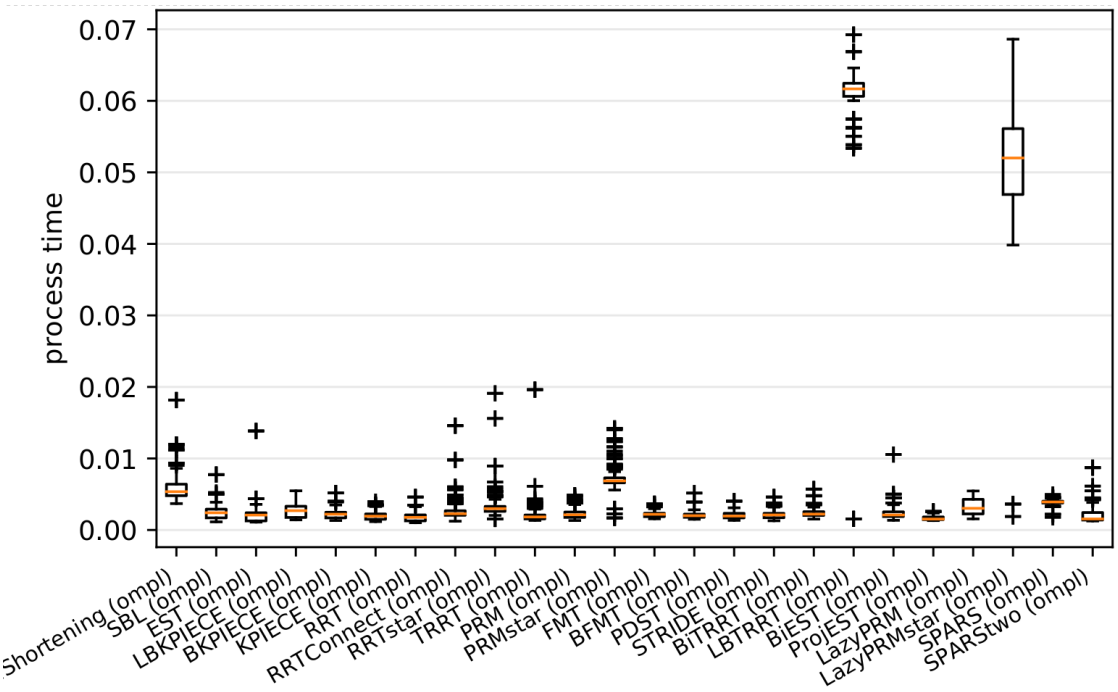


Figure 1.2: The benchmark result when two configurations are defined and pass to the OMPL planner pipeline. All planners completed a 50-cycle query from an initial pose to a goal pose. RRT requires the least amount of processing time at finding the motion planning solution, followed by the PRM.

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 in selecting the motion planner for the dynamic obstacle avoidance.

1.2 EXPERIMENT DESIGN FOR UNPREDICTABLE OBSTACLES USING RRT AND PRM

The cyclical space is populated by the RRT-Newton-Raphson and the PRM-Newton-Raphson pipelines 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 is described by the vectors in *equation 1.2* and the *figure 1.3*

$$c_{initial} = \begin{bmatrix} 0.60 \\ 0.36 \\ -0.60 \\ 0.37 \\ \hdashline 0.39 \\ 0.04 \\ 0.22 \end{bmatrix} \quad c_{goal} = \begin{bmatrix} -0.18 \\ 0.77 \\ 0.12 \\ 0.59 \\ \hdashline -0.09 \\ -0.34 \\ 0.34 \end{bmatrix} \tag{1.2}$$

In this expermiment, the robot will complete a set of ten cyclical space. A static

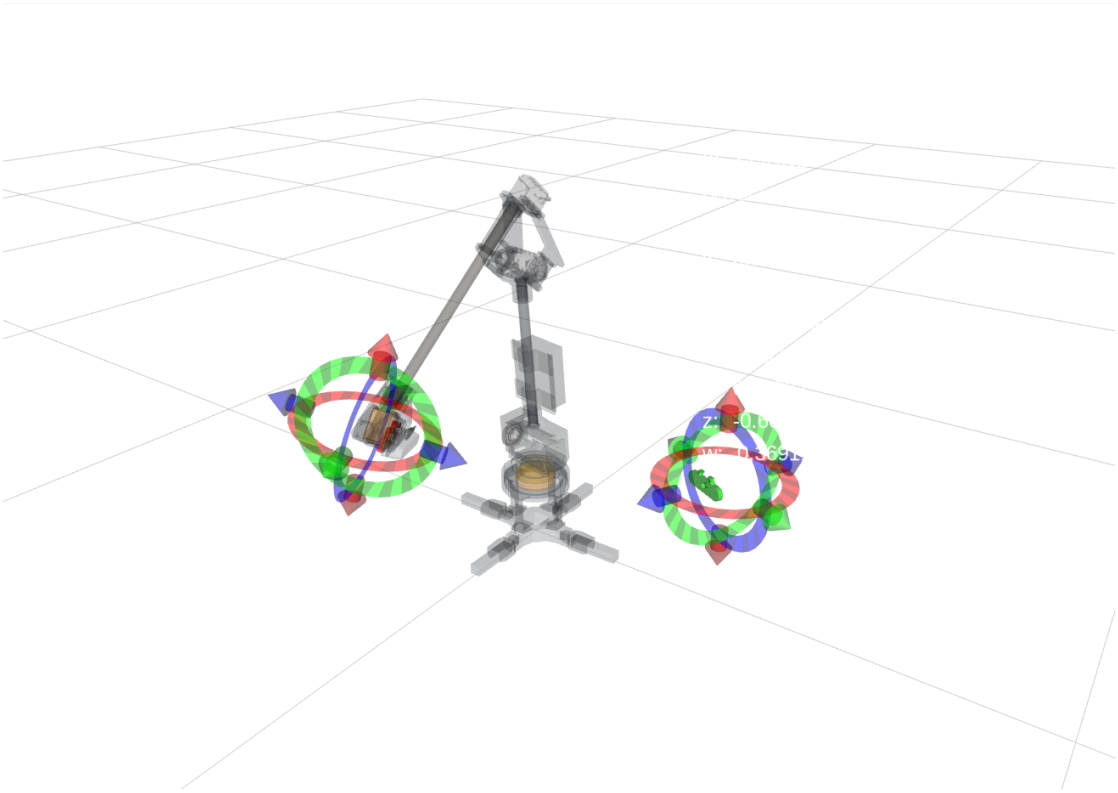


Figure 1.3: The experimentation setup for unpredictable static obstacles where the initial and the goal poses are defined in *equation 1.2*

object is introduced between a straight line that connects between the initial and the goal pose. The obstacle is introduces in the fourth cycle space. It is then removed at the end of the seventh cycle. The obstacle can take any shape and sizes but for this experimentation, the obstacle for the simulation is a cyclinder with height 0.25 m and radius 0.1 m. For the hardware validation, a book with height 0.23 m, length 0.18 m, and 0.02 m.

1.3 EXPERIMENT DESIGN FOR MOVING PLANNING IN DYNAMIC ENVIRONMENT

The cycliccal space is populated only by the RRT-Newton-Raphson pipline. Two poses are defined in this experimentation which has been described in *equation 1.1*. 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 a $x = 1 + 0.7\sin(2\pi(0.0006)\Delta t)$. Two velocities values were used: 50% scale and 10% scale of end-effector's maximum velocity.

The planner is invoked 5 s before the obstacle is placed into the planning scene. The cylinder is directly place into the planning scene such that no motion tracking is necessary for this research. The planner are requested to provide solution for the motion described by the cycle space. Five 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 experimentatation

Table 1.1: The simulated and hardware-connected result of the performance of RRT in a dynamic space. NC stands for No Collision after 5-minute runtime

	time to 1^{st} collision,(s)				
condition\iteration	1^{st}	2^{nd}	3^{th}	4^{th}	5^{th}
simulation$_{0.5v}$	64	NC	NC	133	18
hardware$_{0.1v}$	205	16	17	134	13
simulation$_{0.5v}$	13	23	11	9	13
hardware$_{0.1v}$	17	4	15	7	10

is done, both, in simulation, and with the real robot hardware. However, for both the simulation and the hardware validation, the obstacle are augmented in simulated environment.

1.4 RESULT ON RRT PLANNING

There are two hardware validations performed. One regarding the planning under dynamic environment where a synthetic obstacle, moving periodically, cuts through the line of planning in the cyclical space, C_{cycle}. The second experimentation involve the use of RTAB-Map impregnated with the PHASER implementation where the state estimation pipeline was swapped with the spherical harmonics approach.

1.4.1 Result on Path Planning Under Dynamic Environment

Table 1.1, shows the recorder 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 poor performance is subjected to the algorithm **??**, specifically in line **??** and line **??**, RRT is called. Within this call (algorithm **??**, line **??** consider an obstacle map that is outdated given the cyclinder has moving further towards the manipulator when RRT: line **??**. Within the RRT algorithm, there are no mechanism for the robot to stop or move at a lower rate to avoid the cylinder. *Figure 1.4* 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 $x-axes$ of the $c_{initial}$ and c_{goal}, the planner successfully avoid the obstacles when the lines **??** and **??** in algorithm **??** is invoked.

The planner shows reactive behavior when the cyclical space is initialize, via algorithm **??**. .*Figure 1.5* illustrates such behavior in the simulated environment, and *figure 1.6* shows the same behavior in the hardware reiteration of the experimentation. This is illustrated in *figure 1.7*, where the $joint_2$ and $joint_3$ changes the range of their movement while $joint_3$ changes the rate of its movement.

‘

No significant changes are observed for $joint_4, joint_5$ and $joint_6$. This is the implication of the Pieper-condition manipulator design where, none of the $z-axis$ from the first three joints shares the same crossing point, which suggest the rotation acting by these joints are not a linear transformation. Due to the offset (affine transformation) of the joints' axis of rotation, these joints' there is a bijection mapping of these joints to

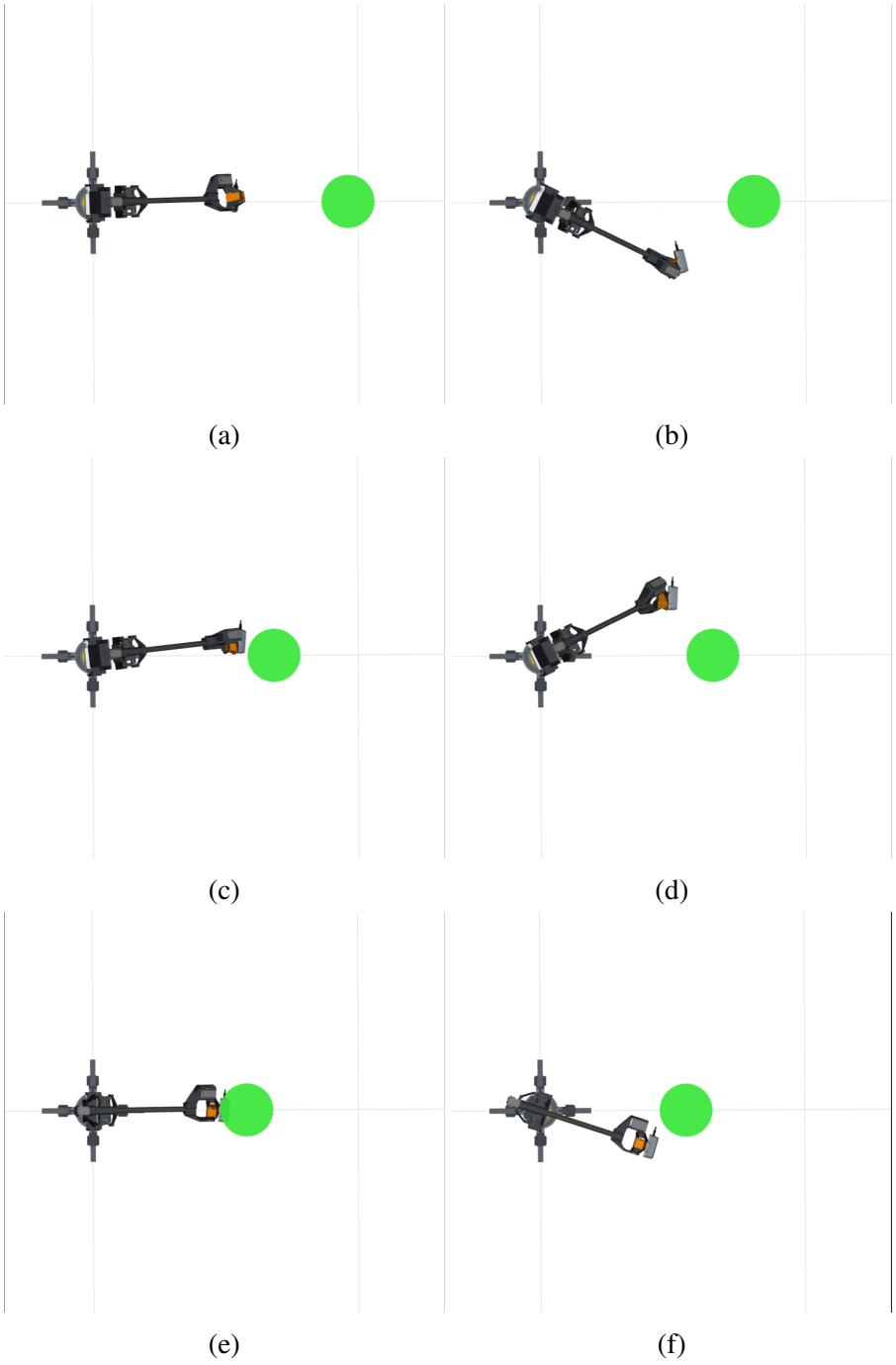
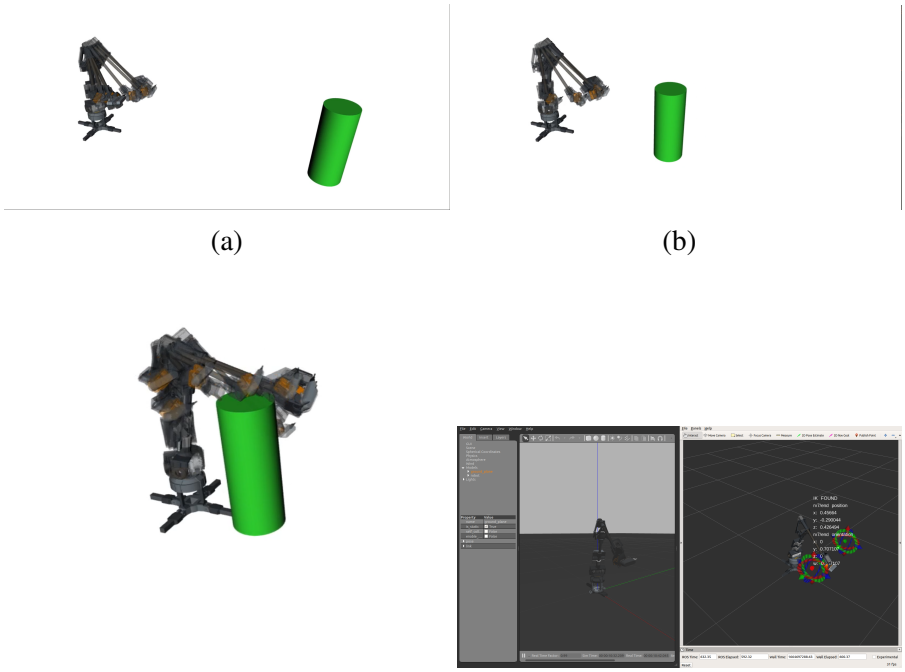


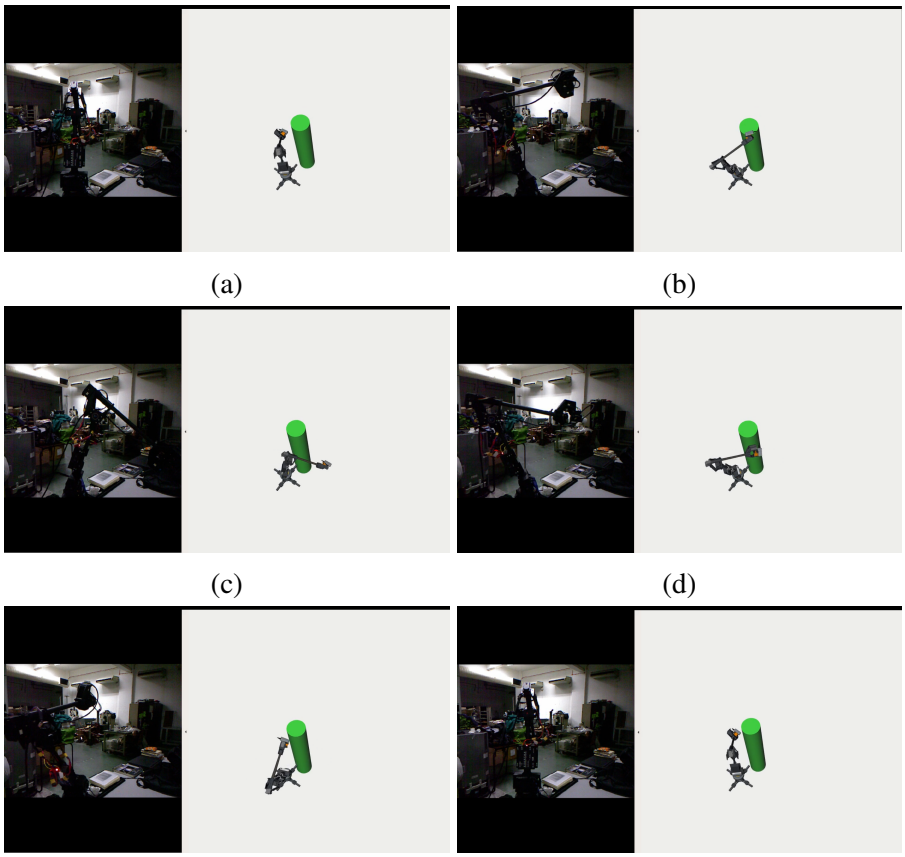
Figure 1.4: This sequence shows the manipulator follows an outdated trajectory and collides with the cylinder despite attempt to move away from the moving cyclinder.



(a) (b)

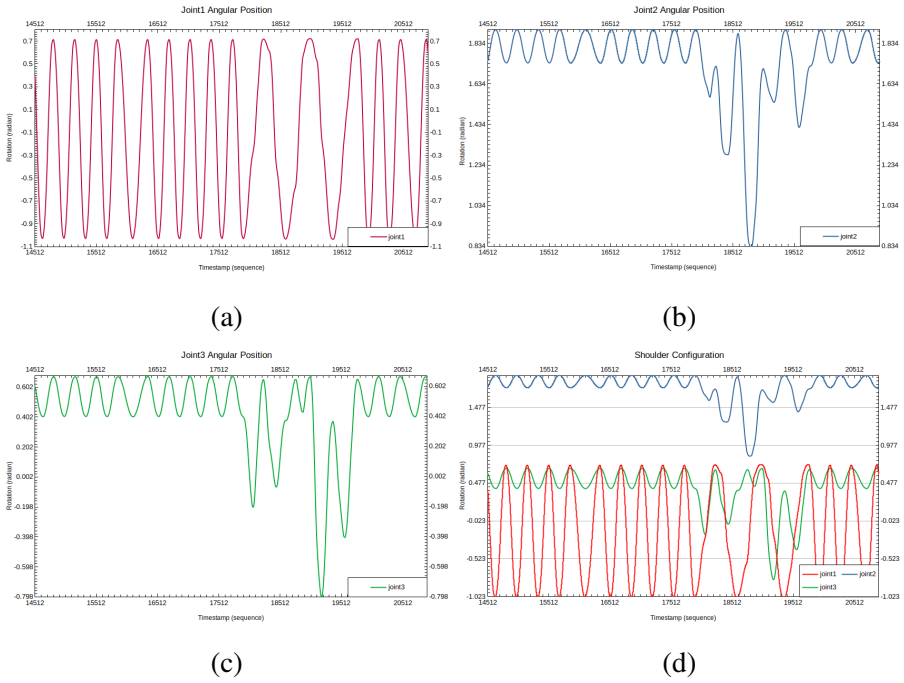
(c) (d)

Figure 1.5: The chronology of attempts at avoiding a moving obstacle when the obstacle approaches the robot. The planning algorithm fails at avoiding the cylinder before it passes the $x-location$ of the poses, c_{intial} and c_{goal}. (c) shows the planner successfully provide a non-colliding solution when the cylinder is moving away from robot. (d) shows the Gazebo as the physic engine to replicate the robot hardware and encoders feedback and the cyclical space initialization.



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

Figure 1.6: The sequence of motion when *r_mini* successfully avoid a moving obstacle when the obstacles at a turning point to move away from the hardware.



(a) (b)

(c) (d)

Figure 1.7: Reaction from joint1, joint2, and joint3 shows that the planner together with the cycle space behave reactively towards the moving object. No rapid movement or rate on the last three joints on *r_mini*

the task-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.

1.4.2 Result on Obstacle-less Planning with SLAM

The result in *figure 1.8* explains the output of the state odometry pipeline. The $\mathbb{R}^3$ part of the task space, C_{ee}, is normalized to the origin $(0,0,0)$.

As the robot arm moves, changing the pose of the task space, velocity change in the movement together with the angular velocity from the rotation change in the end-effector collapses PHASER's state estimation despite the attempt to use the quaternion and the Hilbert's space to discern the correspondence of two 3D point clouds snapshots. In the figure, there are singularities on most of the process. However, estimations are successfully broadcast between 4.3 s to 6.3 s, 16.1 s to 19.4 s, 32.8 s to 35.2 s, 44.2 s to 47.3 s, 56.2 to 64.5 s.

The absence of odometry estimation output, where the pipeline is silence, populates 73.4% of the process. The state estimation for the PHASER implementation is intermittent.

The pose estimation from the SLAM pipeline and also the odometry, where the visual odometry are taken into consideration during data fusion, are only appear between 57.2 s and 61.3 s as shown in *figure 1.9*.

In *figure 1.10*, the outputs of the visual odometry and the SLAM estimation are presented by the light blue lines in the left pane of the screenshot. The right shows, the initial and goal pose of the hardware.

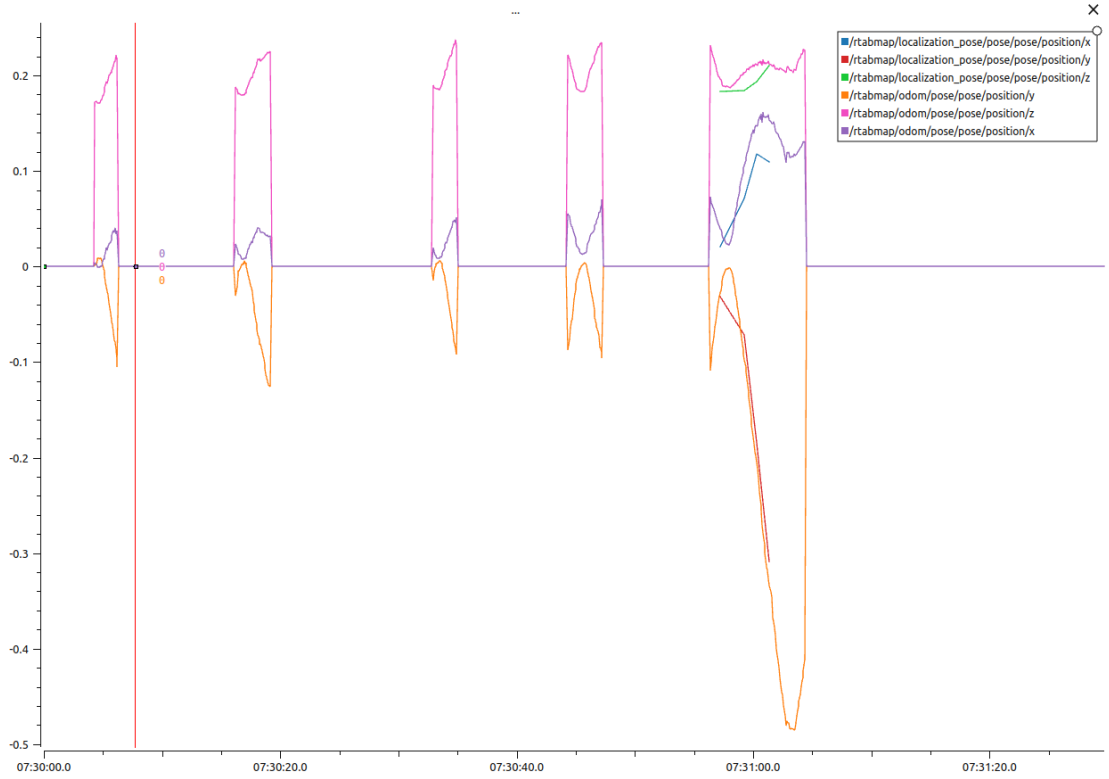


Figure 1.8: The odometry estimate of the end-effector's frame visualize for the $\mathbb{R}^3$ part of the task space, C_{ee}.

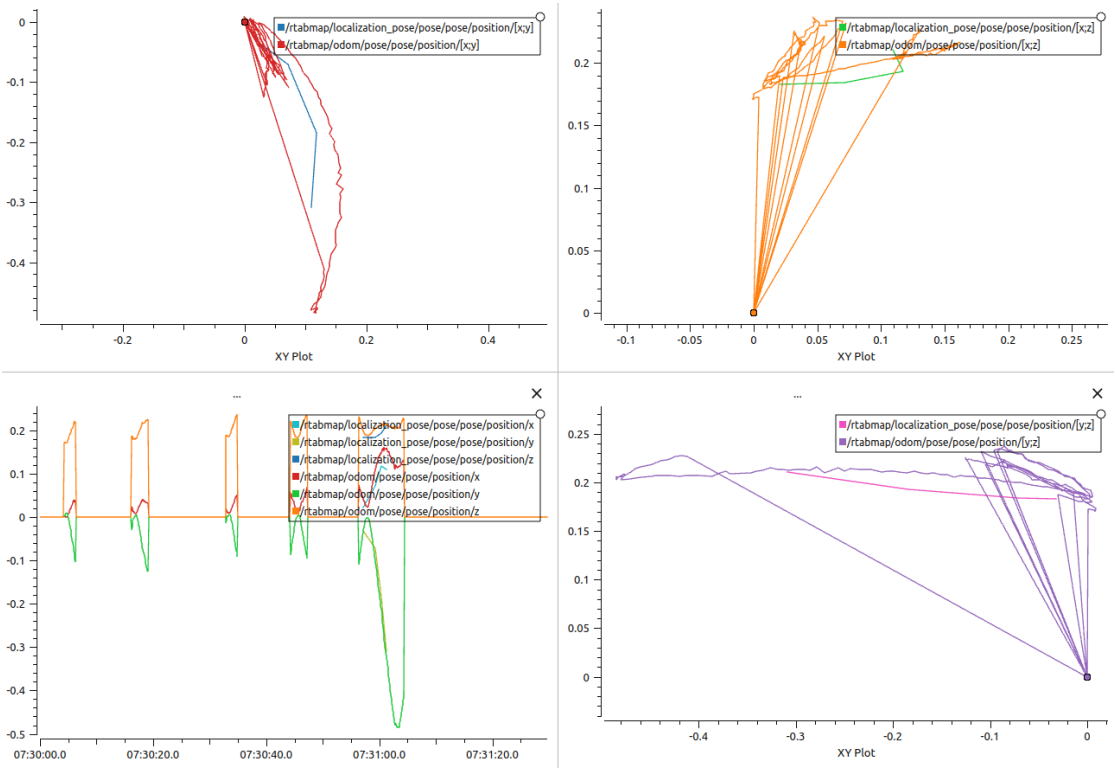


Figure 1.9: The odometry and RTAB-Map state estimation output compared together.

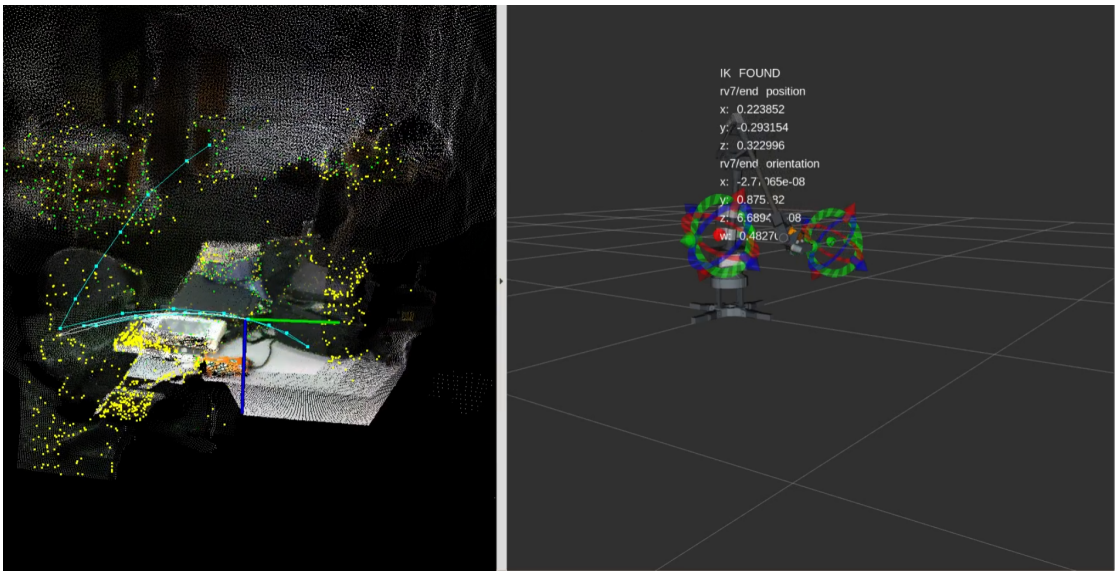


Figure 1.10: This screenshot shows the visual output of the RTAB-Map, where the state estimation and also the visual odometry together with the map of the environment are represented on the left pane of the snapshot. The right pane shows the initial and the goal pose for the C_{cycle}

The SLAM application on the path planning for this research would not be able to give a stream of pseudo-continuous data output from its pipeline. Since, an intermittent nature of these result were shown from the data, it is observed that, when the state estimation is map into the joint-space configuration, and then feeded into the controller's feedback pipeline, the motor could not abide to the trajectory solution populated by the planner. Hence, currently, with PHASER implementation of the state estimation, the SLAM solution and the planning algorithm will not reconcile and will not work without having an intermediate pipeline that can fill up the gap between the one estimation to the next from the RTAB-Map.

1.5 SUMMARY

In this chapter, the result of four experimentations are presented. These results explain the capability of *r_mini* working in a static environment (via the benchmarking) and in a dynamic environment where this research consequently test on the prototype. The later result shows the feasibility of SLAM at estimating the task space of *r_mini*.