~~Chapter 1~~

Note: This is chapter 3,
Accordingly number your
sections and subsections.

Mathematical Backgrounds

1.0 Introduction
A six-axis robot with six revolute joints has been developed as the hardware platform to validate the SLAM solution developed in this thesis. The robot has been named as Richard Mini (R_mini). This chapter presents the features and mathematical model of this robot.

1.1.1 The Mechanical Descriptions

~~Richard Mini is a hardware that had been designed and build to validate the SLAM solution in this thesis.~~ Richard Mini (r_mini) has one fixed link, six moving links, six revolute joints, and six frames attached to the links. The links are named numerical: link1, link2, ..., link6 shown in figure 1.3. The same is true for the joints: joint1, joint2, ..., joint6. All joints are described by the frames attached to the links. Here, I will denote a frame as $\vec{\boldsymbol{p}}$. These joints represent the state of the manipulator which parameterized the control space $\boldsymbol{q}$ where $\boldsymbol{q} \in C^{control}$. $C^{control}$ is a 6-dimensional control space.

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, C^{ee}, refers to the control space of the end-effector where the controlling pipelines would take in $c^{ee} = (\theta_1, \ldots, \theta_6) \in C^{control}$, and the equivalent pose is in the configuration space, C_{ee}.

The figure 1.2 shows the $z - axis$ of the frames aligns with the rotation of the actuators ~~on the robot~~.

$r_mini$follows the condition advised by Pieper (1968) :three last joints are collated and have a shared rotation axis point as prescribed in figure 1.1.

These three joints represent the wrist of ~~r_mini~~.

$r_mini$uses $Z-Y-Z$ Euler convention to present rotation state of the end effector. This convention conforms to the Wigner-D matrix parameterization where this research will use to estimate the orientation of the end-effector.

Note:
Always mention figures/tables maintaining sequences, do not jump forward.
# use R_mini consistantly instead of r_mini (Also avoid using Rimini)
Avoid merging words. Do not use "paper" instead of "thesis".

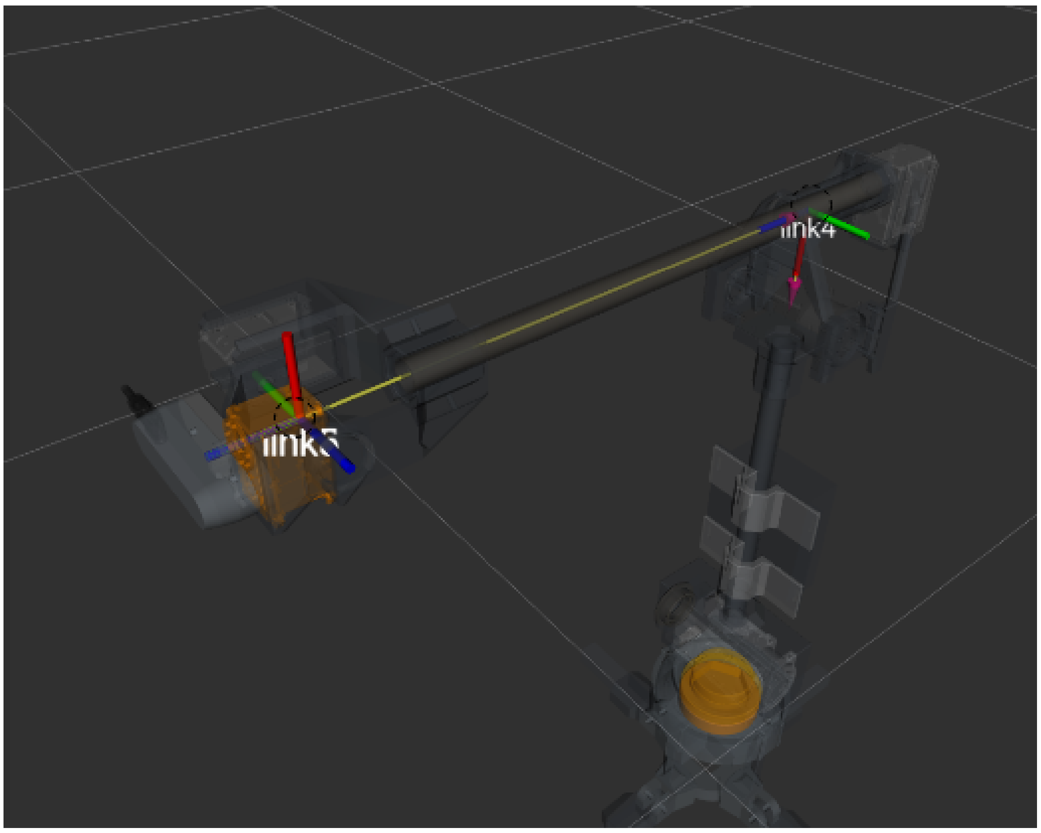
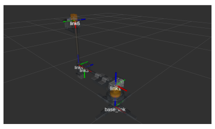
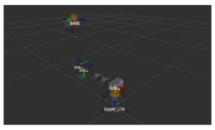
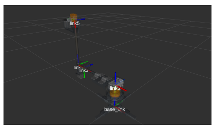
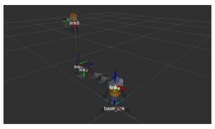


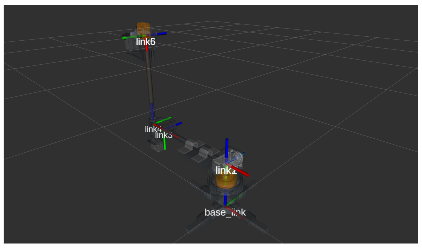
Figure 1.1: *r_mini* wrist conforms to Pieper condition where axis of rotation for joint4, joint5, and joint6 share points of intercept. The dash circle in the diagram a possible point of intercepts. Both point are valid for a Pieper condition


(a) Frame 1


(b) Frame 2 sharing origin with Frame1


(c) Frame 5


(d) Frame 6 sharing the origin with frame 5



(e) Rimini build completion.

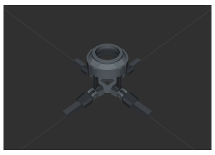
Figure 1.2: The location and orientation of ~~Rimini~~. The choice of the orientation for each frames are based on Denavit-Hartenberg. The joints are values represented by the angle between two $x-axes$ around the $z-axis$ or rotation axis of each actuators

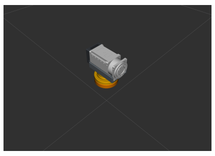
1.1.2 Forward Kinematics of ~~*r_mini*~~

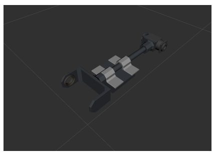
The kinematics of *r_mini* follows the Denavit-Hartenberg (DH) convention. In this convention, the homogenous transformation of one frame to the next frame in the kinematic chain of the robot does not entails any rotation about the $y-axes$. In so doing, the homogenous transformation between two frames are simplified. By disregarding the rotation about the $y-axis$, matrix-to-matrix multiplication between two frames required for homogenous transformation is computationally relax. To help satisfy this constraint about the $y-axis$, two condition should be followed:

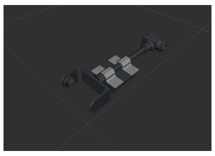
- The axis x_i belonging to $frame_i$ should be perpendicular to the axis z_{i-1} belonging to the $frame_{i-1}$.
- The axis x_i intersects the axis x_{i-1}

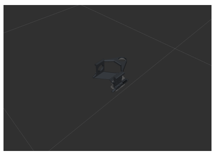
Here, the subscript i represent the frame number, hence, the link and joint number. Based on these contraints, the DH parameters are summarized in table 1.1, where a_i is the offset of the $frame_i$'s $z-axis$ along the x_i, α_i is the rotation of $frame_i$ about the x_i, d_i is the offset of the $frame_i$'s $x-axis$ along the z_i, and θ is the $joint_i$ angle of rotation. DH-convention states

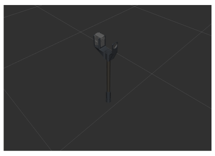

(a) base link

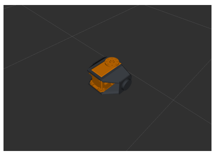

(b) link1

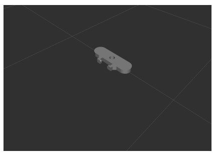

(c) link2

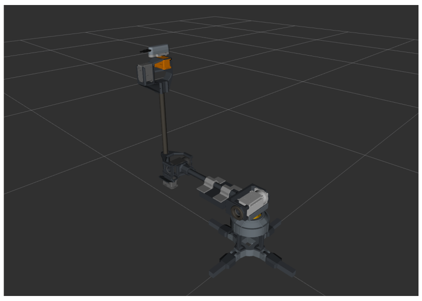

(d) link2


(e) link3


(f) link4


(g) link5


(h) link6.


(i) Rimini in simulation

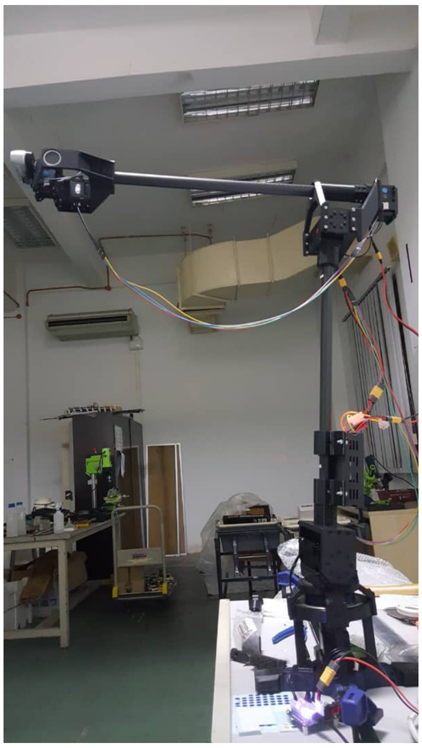

(j) Rimini build completion.

Figure 1.3: *r_mini* and its links

Table 1.1: DH-parameter table

Link (i)	a_i	α_i	d_i	θ_i
1	0	0	0.196	θ_1
2	0	$-90°$	0	θ_2
3	-0.373	0	0	θ_3
4	-0.08	$-90°$	0	θ_4
5	0	$-90°$	0.391	θ_5
6	0	$-90°$	0	θ_6

that each joint values are defined by the angle between two $x - axes$. As an example joint1 is represented by the angle between frames attached to link1 and the base frame. Similarly joint2 is represented by the angle between x_1 and x_2.

Each row in table 1.1 describe the homogenous transformation, A_i from the $frame_{i-1}$ to $frame_i$. The transformation is repesented by four matrix operation:

$$A_i = Rot_{z,\theta_i} T_{z,d_i} T_{x,a_i} Rot_{x,\alpha_i} \tag{1.1}$$

where Rot, T and a is the representation of rotation and translation in special Euclidean group $\boldsymbol{SE(3)}$. Here $A \in \boldsymbol{SE(3)}$. The homogenous transformation is represented by the homogenouns transformation matrix defined as:

$$A = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \tag{1.2}$$

where $\boldsymbol{R} \in \boldsymbol{SO(3)}$ and $\boldsymbol{t} \in \Re^3$. $\boldsymbol{SO(3)}$ is the special orthogonal group that represent orientation of the frames in the Euclidean space $\boldsymbol{R^3}$.

With each row of the DH-parameter table 1.1 represented by a homogenous transformation matrix A, we can map the r_mini configuration space $\boldsymbol{q}$ into $\Re^3$:

$$\begin{aligned} \boldsymbol{p_6} &= f(\boldsymbol{q}) \\ &= A_1^0(\theta_1)A_2^1(\theta_2)A_3^2(\theta_3)A_4^3(\theta_4)A_5^4(\theta_5)A_6^5(\theta_6)\boldsymbol{p_0} \end{aligned} \tag{1.3}$$

.

Define p, p0, p6 etc.

1.1.3 Inverse Kinematics of *r_mini*

The inverse kinematics solution for *r_mini* follows the analytical approach derived by Pires (2007), since both the IRB1600 and *r_mini*has the Pieper condition addressed in section 1.1.1. However, given that the analytical approach introduce multiple solution, accordingly, subjected the motion planning to singularities, this ~~report~~ adopts ~~a~~ numerical method approach ~~which are used~~ during the simulation and experimentation.

Newton-Raphson method to find the inverse kinematic solution of *r_mini*, $\hat{c}^{ee}$. The generalization of the method uses the current value of the robot's encoder, $c^{current}$, and the termination value, $\epsilon = 0.005$, to end the iteration. Algorithm 1 delineate the the method.

```
Algorithm 1: getInverseKinematics
  Input: $c_{goal}$, $c^{current}$,$\epsilon$
  Output: $\hat{c}_{ee}$
1 $e \leftarrow$ getForwardKinematic($c^{current}$)
2 while $\|e\| \neq \epsilon$ do
3     $\hat{c}^{estimate} \leftarrow c^{current}+$ getInverseJacobian( $c^{current}$ )$e$
4     $e \leftarrow C_{goal}-$ getForwardKinematic($c^{estimate}$)
5     $c^{current} \leftarrow c^{estimate}$
6 $\hat{c}^{ee} = c^{current}$
7 return $\hat{c}$
```

1.2 *r_mini*'s Path Planner

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*:

where k represent the, number of node in the tree generated by the RRT, $\mathcal{M}$, represent the collision map which is part of the planning scene where all RRT sampling takes place and $\mathcal{T}$ 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 C_{goal}. Line 4 invokes the k-nearest neighbor to find a selection of nodes that is closes to the state configuration, c_{random}. 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

Algorithm 2: generateRRT

```
  Input: $C_{initial}$,$C_{goal}$,$\Delta t$,$k$,$\mathcal{M}$
  Output: $\mathcal{T}$
1 $\mathcal{T}$.initialize($C_{initial}$,$C_{goal}$,$\mathcal{M}$)
2 while $c_{new} \neq C_{goal}$ do
3     $c_{random} \leftarrow$ randomState()
4     $c_{near} \leftarrow$
        kNearestNeighbor($k$,$c_{random}$,$\mathcal{T}$)
5     $\boldsymbol{u} \leftarrow$ selectInput($c_{random}$,$c_{near}$)
6     $c_{new} \leftarrow$ newConfiguration($c_{near}$,$\Delta t$)
7     $\mathcal{T}$.append($c_{new}$,$c_{near}$,$\boldsymbol{u}$)
8 return $\mathcal{T}$
```

space via the kinematic Jacobian which requires the information on the Δt. 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, C_{limit}, containing configuration that abides the joint-configuration space range and angular velocity limit.

The configuration space where the RRT-sampling is concerned is modified in this ~~paper~~ where, the rotation representation and its sampling is in $R \in \mathbb{H}$, such that the parameterization of the Hamiltonian-space is the quaternions, $\boldsymbol{q} \in \mathbb{R}^4$. Therefore, the representation of the robot poses and also the non-colliding poses, $(\boldsymbol{q}, \boldsymbol{t})$ are explained in *equation 1.4*.

$$c_{pose} = \begin{bmatrix} \boldsymbol{q} \\ \hdashline \boldsymbol{t} \end{bmatrix} \tag{1.4}$$

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.

1.2.1 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, $c_{initial}$ to the goal pose, c_{goal}, 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,

$$\tau = (C^{control}, t) \tag{1.5}$$

which forms a cyclical motion between the initial pose and the goal pose. Here, $C_{cycle} = \overbrace{C_{initial \to goal}}^{\text{~~see~~ algorithm 3 line 4}} + \overbrace{C_{goal \to initial}}^{\text{~~see~~ algorithm 3 line 7}}$. ~~The following~~ algorithm 3 ~~block~~ explains how the C_{cycle} space is constructed.

Algorithm 3: generateCycleSpace

```
   Input: $c_{initial}$,$c_{goal}$,$\Delta t$
   Output: success_status
1  Function generateCycleSpace($c_{goal}$, $c_{initial}$):
2      success_status ← false
3      while within iteration do
4          $\mathcal{T}_{initial \to goal}$ ← generateRRT($c_{initial}$,$c_{goal}$,$\Delta t$)
5          success_status ← moveRobot($\mathcal{T}_{initial \to goal}$,$\Delta t$)
6          if success_status = true then
7              $\mathcal{T}_{goal \to initial}$ ← generateRRT($c_{goal}$,$c_{initial}$,$\Delta t$)
8              success_status ← moveRobot($\mathcal{T}_{goal \to initial}$,$\Delta t$)
9      return success_status
10 Function moveRobot($\mathcal{T}$, $\Delta t$):
11     for all index in $\mathcal{T}$.vertices do
12         $c^{cycle}$(index) ← getIK($\mathcal{T}$.vertex(index))
13         $t$ ← $\mathcal{T}.\boldsymbol{u}$.(index)$\Delta t$
14         $\mathcal{T}$.append($c^{cycle}$, $t$)
15     success_status ← TrajectoryController($\tau$)
```

The control space are represented by the trajectory in the joint-configuration space $C^{control} \subset C^{cycle} \in S$ where S is the 6-hypersphere. In *equation 1.5* the joint-configuration space are equivalent with the configuration space in *equation 1.4*. 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, $\mathcal{T}$, are done within the $\mathbf{SO(3)} \times \mathbb{R}^3$ topology. The free configuration space, or the non-colliding pose, are represented by, $C_{free} = C_{workspace}/C_{obstacle}$. According to LaValle (1998), the $C_{obstacle}$ are 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 n-hyperspace, to map them into the $\boldsymbol{SO(3)} \times \mathbb{R}^3$, we use the kinematic Jacobian.

1.3 The Spherical Harmonics

In this thesis, spherical harmonics are used to discern the estimate of the orientation of *r_mini*'s end effector via the raw data coming from the RGB-D sensor. The spherical analysis derives from the fourier transformation of non-periodic function. Here, we will represent the pointclouds coming from the RGB-D sensor as an unknown functions (Osteen, Owens, and Kessens 2012). With spherical harmonics, the state of the end effector are computed based on two overlapping functions $\tilde{f}$ and $\tilde{g}$ both representing the pointclouds. We project these functions into a unit-hypersphere $D^2 = \{x \in \mathfrak{R}^3 \,|\|x\|_2 = 1\}$. where the I paremeterized the projection of $\tilde{g}$ and $\tilde{f}$ based on the same parameterization in D. M. Healy et al. (2003). Convolution of the two function are represented by inner product, acting on the Hilbert space, $L^2(S^2)$:

$$\langle \tilde{g}, \tilde{f} \rangle = \int_{\boldsymbol{\omega} \in S^2} \tilde{g}(\boldsymbol{\omega}) \overline{\tilde{f}(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega})} \tag{1.6}$$

where $\mathrm{d}\boldsymbol{\omega} = sin(\theta) d\theta d\psi$ and $\tilde{g}$, $\tilde{f}$ are the arbitrary integral functions on S^2 and $\overline{\tilde{f}}$ is the complex conjugate.

The spherical Fourier transform of any function $\tilde{f} \in S^2$ is defined in $\tilde{F^l_m} = \langle \tilde{f}, Y^l_m \rangle$, where Y^l_m are the spherical harmonics ~~gof~~ degree $l \in \mathbb{N}_0$. of order $m \in [0, l] \in \mathbb{N}_0$. This forms the orthonormal basis over $L^2(S^2)$,

$$Y^l_m(\boldsymbol{\omega} = (-1)^m \sqrt{\frac{(2l+1)(l-m)!}{4\pi(l+m)}} P^l_m(cos(\theta)) exp(-jm\psi) \tag{1.7}$$

where P^l_m are the associated Legendre polynomials implemented by D. M. Healy et al. (2003).

The integral over the sphere are evaluated using the sampling theorem formulated by Driscoll and Dennis M. Healy (1994):

$$F^l_m = \frac{\sqrt{2\pi}}{2\tilde{B}} \sum_{j=0}^{2\tilde{B}-1} w_j \tilde{f}(\theta_j, \psi_k) \overline{Y}^l(\theta_j, \psi_k) \tag{1.8}$$

,

where $\tilde{B}$ is the spherical bandwidth, θ and ψ are the samples with the corresponding weight w. Equations 1.8 gives us a way o transform the function $\tilde{f}$ and $\tilde{g}$ which belongs in $\mathfrak{R}^3$ into the Hilbert space, i.e. the frequency domain, via this relationship:

$$\tilde{f}(\boldsymbol{\omega}) = \sum_{l\geqq 0} \sum_{m\leqq} \tilde{F}^l_m Y^l_m(\boldsymbol{\omega}) \tag{1.9}$$

.

The implementation of the equations 1.7 1.8 and 1.9 are coupled with the datastructures ~~of~~ in equations 1.17 1.19 1.20 and 1.21

to form a correlation between the two overlapping functions $\tilde{g} \in L^2 S^2$ and $\tilde{f} \in L^2(S^2)$:

$$\tilde{C} = \sum_{l>0} \sum_{m\leq l} \sum_{m'\leq l} \tilde{F}^l_m \overline{\tilde{G}^l_{m'} D^l_{m'}} \tag{1.10}$$

$$\tilde{C}_r = \sum_{l>0} \sum_{m\leq l} \sum_{m'\leq l} \tilde{F}^l_{m,r} \overline{\tilde{G}^l_{m',r} D^l_{m'}} \tag{1.11}$$

$$\tilde{C}_g = \sum_{l>0} \sum_{m\leq l} \sum_{m'\leq l} \tilde{F}^l_{m,g} \overline{\tilde{G}^l_{m',g} D^l_{m'}} \tag{1.12}$$

$$\tilde{C}_b = \sum_{l>0} \sum_{m\leq l} \sum_{m'\leq l} \tilde{F}^l_{m,b} \overline{\tilde{G}^l_{m',b} D^l_{m'}} \tag{1.13}$$

where $\tilde{C}$ represent the correlation for the range encoding, and $\tilde{C}_r$, $\tilde{C}_g$, and, $\tilde{C}_b$ represent the correlation for the red, blue, green, band images. The function D, the Weigner-D matrix, helps to rotate the spherical harmonics in the Hilbert space. The function is parameterized by the $Z-Y-Z$ Euler-angles rotation. Since, the end effector are parameterized with the same orientation, the orientation estimate of the robot's end effector are given by the argument max of the resultant correlation,

$$\boldsymbol{r}(\theta_4, \theta_5, \theta_6) = \underset{\boldsymbol{r}\in \boldsymbol{SO(3)}}{\operatorname{argmax}}(\tilde{C} \boxplus \tilde{C}_r \boxplus \tilde{C}_g \boxplus \tilde{C}_b) \tag{1.14}$$

where $\boxplus$ represent Gaussian data fusion technique.

1.4 The Use of Kinematics Jacobian and the Singularities or the Manipulator to Relax the Cost Function

Equation 1.15 is defined as,

$$\dot{\boldsymbol{q}} = J^{-1}\boldsymbol{v} \tag{1.15}$$

where the angular velocity in $\mathfrak{C}^3$ changes abruptly. We explain that this abrupt changes is not just because of the zero-determinant of the Jacobian, J, but also because of the duality in the solution explained in section 1.1.3. At a point where the determinant of the Jacobian of the fundamental kinematics equations approaches to zero, the angular velocity of any of the joint state approaches to infinity. This behavior is represented by the mapping of the configuration space to the end effector's state.

The fundamental kinematics equation demonstrate a peculiar characteristic. The equation 1.15 explains, regardless of the infinity velocity when approaching singularity, the task space does not reflect any abrupt or catastrophic movement. This behavior when discern in context of kinematic equation 1.15, shows that the robot configuration changes abruptly when travesing into a subset of the workspace that coincide with zero-determinant Jacobian. Should a closed-form solution to the subset of the function that represent the singularity in the workspace is needed, ~~readers~~ can venture into invoking the fundamental theory of determinants that give constraints and conditions to satisfy non-invertible Jacobian. However, caution ensues as this efforts may not be trivial if solved analytically.

However, since the sampling space for the RRT includes the

Please complete the sentence

1.5 SLAM for *r_mini*

r_mini's end effector is an RGB-D camera: the Intel Realsense 435D. The camera livefeeds point clouds, imagery, and a calibrated pointcloud imagery where each point are encoded with RGB-value; a depth image. The depth images are bound by noise. Hence, the mathematical model of the map are based on Bayesian probability model:

$$P(n|z_{1:t}) = \left[1 + \frac{1-P(n|z_t)}{P(n|z_t)} \frac{1-P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1-P(n)}\right] \tag{1.16}$$

where $P(n)$ are the prior probability distribution, $P(n|z_{1:t-1})$ are the previous estimate of the map and z_t is the measurement posterior distribution, and n, a node in a segmented space; the map model encoding of a octree datastructure. This maps impute the probability value into the segmented space, representing occupied and free space. To relax computation, the continuous values are conditioned to encode binary values by transforming the domain of the random variables n into a logarithmic scale using logit model. The numerator in equation 1.16 are normalizing parameter which can be disregarded without having to change the mean, mode, and the median of the maps distribution (Hornung et al. 2013). Hence, together with the relax logit function, equation 1.16 is reduced to:

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t) \tag{1.17}$$

where

$$L(n) = log\left[\frac{P(n)}{1 - P(n)}\right] \tag{1.18}$$

Equation 1.17 are constructed based on the pointcloud datastructure. The space, in the case of *r_mini*, the manipulator workspace, $\mathcal{W}$ are divided into two values; zero for free space, and one for occupied space. Since, Realsense also produce RGB imageries, the map also valid for the occurance of the color blue, red, and, green in as a subset of $\mathcal{W}$.

$$L_{red}(n) = log\left[\frac{P_{red}(n)}{1 - P_{red}(n)}\right] \tag{1.19}$$

$$L_{green}(n) = log\left[\frac{P_{green}(n)}{1 - P_{green}(n)}\right] \tag{1.20}$$

$$L_{blue}(n) = log\left[\frac{P_{blue}(n)}{1 - P_{blue}(n)}\right] \tag{1.21}$$

With logistic encoding, the datastructure of the map is compact, relaxing computational expenses and thus eliminate bottleneck on the rich streaming from the RGB-D sensor. This map model, also known as octomap encodes the posterior of the SLAM solution in equation. Given the stability of binary values in static environment mapping, a dynamic objects introduced in the workspace are not represented by the octomap datastructure.

The SLAM part of this ~~report~~ implements the RTAB-Map library and its framework to encodes the map model and also the spherical harmonics implementation suggested in the earlier section as the localization engine for the instead the implementation that comes with the libary. The ??

$$\Phi* = \underset{\Phi}{\operatorname{argmax}}\, P(\Phi|Q*, \mu) \tag{1.22}$$

where Φ is the map of the environment from the datastructure infused with the spherical harmonic terms. μ is the the set of waypoints generated by the planner.

$$\begin{aligned} Q* &= \underset{Q}{\operatorname{argmax}}\, P(Q|\Phi, Z) \\ &= \underset{C_{ee}}{\operatorname{argmax}}\, \frac{P(Z|\Phi*, C_{ee})P(C_{ee}|\Phi*)}{P(Z)} \\ &= \underset{C_{ee}}{\operatorname{argmax}}\, logP(Z|\Phi*, C_{ee}) + logP(C_{ee}|\Phi*) \end{aligned} \tag{1.23}$$

where C_{ee} is the prior estimate of the state of the robot's task space on the map model in 1.22.

With these models, the cost function can be defined as:

$$Cost(\boldsymbol{c}) = \gamma\epsilon + \sum_{\boldsymbol{v}\in V} \Phi^{t-1}[T_q Z]^2 \tag{1.24}$$

.

The cost function is differentiated with respect to $\boldsymbol{c}$ to minimized the error ϵ everytime there is an update on the, sensor reading Z, map Φ and command update from V.

To minimize this cost function, equation 1.24 are differentiated.

1.6 Summary

This chapter introduced the mathematical background that will be adopted and implemented into the following chapters. The geometry of *r_mini* configuration space, and its implicit representation, the control space, are explained. Further, this chapter introduce the SLAM solution to estimate the task space of the robot.

Bibliography

Driscoll, James R. and Dennis M. Healy (June 1994). “Computing fourier transforms and convolutions on the 2-sphere”. In: *Advances in Applied Mathematics* 15.2, pp. 202–250. ISSN: 10902074. DOI: 10.1006/aama.1994.1008.

Healy, D. M. et al. (2003). “FFTs for the 2-Sphere-Improvements and Variations”. In: *Journal of Fourier Analysis and Applications* 9.4, pp. 341–385. ISSN: 10695869. DOI: 10.1007/s00041-003-0018-9. URL: https://link.springer.com/article/10.1007/s00041-003-0018-9.

Hornung, Armin et al. (Apr. 2013). “OctoMap: An efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous Robots* 34.3, pp. 189–206. ISSN: 09295593. DOI: 10.1007/s10514-012-9321-0. URL: http://link.springer.com/10.1007/s10514-012-9321-0.

LaValle, Steven M (1998). *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep., pp. 98–11. URL: https://www.cs.csustan.edu/$%5Csim$xliang/Courses/CS4710-21S/Papers/06%20RRT.pdf.

Osteen, Philip R., Jason L. Owens, and Chad C. Kessens (2012). “Online egomotion estimation of RGB-D sensors using spherical harmonics”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., pp. 1679–1684. ISBN: 9781467314039. DOI: 10.1109/ICRA.2012.6225098.

Pieper, Donald Lee (1968). “The Kinematics of Manipulators Under Computer Control”. PhD thesis. Stanford University.

Pires, J.norberto (2007). “INDUSTRIAL ROBOTS PROGRAMMING : INDUSTRIAL ROBOTS PROGRAMMING”. PhD thesis, pp. 198–197. ISBN: 9780387233253.