Algorithms for computing Safe Working Zone of a Parallel Manipulator

A Project Report

submitted by

DHRUVESH PATEL

in partial fulfillment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
(Engineering Design)

and

MASTER OF TECHNOLOGY
(Automotive Engineering)

DEPARTMENT OF ENGINEERING DESIGN
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2016

THESIS CERTIFICATE

This is to certify that the report titled "**Algorithms for computing the Safe Working Zone of a Parallel Manipulator**", submitted by **Patel Dhruvesh (ED11B026)**, to the Indian Institute of Technology, Madras, in partial fulfillment of the award of the degrees of **MASTER OF TECHNOLOGY** and **BACHELOR OF TECHNOLOGY**, is a *bona fide* record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Sandipan Bandyopadhyay
Project Guide
Department of Engineering Design,
Indian Institute of Technology Madras

Prof. Nilesh Jayantilal Vasa
Head of the Department
Department of Engineering Design,
Indian Institute of Technology Madras

Place: Chennai

Date: June 14, 2016

ACKNOWLEDGEMENTS

I would like to thank my guide Dr. Sandipan Bandyopadhyay for his invaluable guidance, constant encouragement and for molding me from a student to a professional. He not only guided but helped me conceive the ideas presented in this work. He was supportive of my decision to change the topic of my project when I was unable to produce results. I am greatly thankful to my parents who have been a constant support and source of motivation in my life and without whom I would not be what I am today. I would also want to thank all my colleagues in the Robotics Lab., IIT Madras, who shared their valuable knowledge and experience when I was in need. Lastly, I want to thank my friends who have always encouraged and helped me in my work.

ABSTRACT

KEYWORDS: Safe Working Zone ; Parallel Manipulators; Kinematics; 3-RRS; Five-bar ; Singularities; Kinematic node

This report presents new algorithms for computing the Safe Working Zone (SWZ) of parallel manipulators. The workspace analysis of parallel manipulators is more challenging than that of the serial manipulators, as the former has additional singularities, called gain-type singularities inside the workspace boundary marked by the loss-type or serial-type singularity. The motivation for the definition and computation of the SWZ comes partly from the problem of path-planning for parallel manipulators, where the paths need to be free from all types of singularities as well as physical limits such as link intereference.

SWZ of a parallel manipulator, defined later in the report, is essentially a subset of the workspace of the manipulator which is free from all singularities and physical limits. Computing such a subset of the workspace, a priori, eliminates the task of path validation during path planning.

Kilaru *et al.* (2015) makes use of the SWZ as a design constraint and proposes a framework for designing parallel manipulators considering its kinematic as well as dynamic performance. This kind of framework for iterative design requires the algorithms for finding the SWZ to be extremely fast and efficient as the entire design process goes through thousands of iterations.

First the general procedure to find the singularity functions, essential for computing the SWZ, is outlined followed by the proposed algorithms for computing the SWZ. To benchmark the performance of the new algorithms, they are compared to an existing algorithm by using them to find the SWZ of a five-bar and a 3-RRS parallel manipulator.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

SWZ	Safe Working Zone
MSoR	Maximal Surface of Revolution

CHAPTER 1

Introduction

1.1 Overview

Computing the workspace of a robotic manipulator is useful in analysis of an existing manipulator as well as in designing a new manipulator iteratively (Kilaru *et al.* (2015)). Also, the knowledge of singularity-free regions in the workspace and their topologies can be very useful in real-time path-planning. Once this information is obtained offline, for a particular design of the manipulator, the issue of checking paths for singularities can be completely omitted from the problem of path-planning, bringing down its complexity considerably.

A parallel manipulator can be thought of as a set of serial manipulators combined at the end effector, which also means that analyzing the workspace of a parallel manipulator is more complicated than that of a serial manipulator. Owing to the closed loops of links present in a parallel manipulator, its workspace is plagued by another type of singularity, called the *gain-type* singularity apart from the usual loss-type singularity inherited from the constituent serial chains, which complicates the problem. These singularities are also referred to as type-I and type-II singularities in Gosselin and Angeles (1990). In order to perform a realistic analysis of a manipulator design, the physical limitations such as link interference and joint limits have to be considered along with the kinematic singularities. Checking for link interference is in general computationally intensive, and the number of times it is performed should be kept to a minimum to restrict the total computation time within realistic limits.

Keeping all these issues in mind, this work presents and compares the performance of a set of algorithms to find the Safe Working Zone (referred to as SWZ hereafter) of parallel manipulators with two and three degrees-of-freedom, defined in Srivatsan and Bandyopadhyay (2014) and explained in Chapter 2. The algorithms presented in this work are implemented in `C++` language.

1.2 Literature survey

Avoiding singularities is a major task while planning paths in the workspace of a parallel manipulator. Various approaches have been resorted to to accomplish this task. For example, Bohigas *et al.* (2013) numerically identifies the path between two non-singular configurations in the workspace of a parallel manipulator, while the same is achieved by Dash *et al.* (2005) by identifying and grouping clusters of singularity points and modeling them as obstacles. Another approach as mentioned by Leguay-Durand and Reboulet (1997) is used by Nasa and Bandyopadhyay (2011) to avoid singularities on a path, in a lower dimensional space than the degree-of-freedom of the manipulator, by using the freed degree-of-freedom to avoid singularities. The main limitation of these approaches is that the solutions obtained are specific to the path under consideration and have to be recomputed afresh for every new path, causing a severe computational overhead in real-time applications.

Another approach to handle the problem is to identify singularity-free zones in the workspace of a manipulator and plan paths inside these zones, which would guarantee them being singularity-free. Li *et al.* (2006) and Jiang and Gosselin (2009) apply constrained optimization to identify maximal singularity-free zones centered around a given point in the workspace of 3-RPR manipulator and Stewart-Gough Platform, respectively. The limitation of these methods is that they use closed-form expressions for the singularities which are not yet found for all types of parallel manipulators and they also do not take into account all the other relevant issues, i.e., link interference and joint limits, that arise in the workspace of a parallel manipulator along with the kinematic singularities.

Karnam *et al.* (2016) attempts to address all the above mentioned issues but loses out on the continuity aspect of the scanned 3D region. This work attempts to resolve that issue, as well as improve the performance compared to the existing algorithms.

1.3 Objectives

Following are the major objectives which were kept in mind while formulating the algorithms presented in the subsequent chapters:

- **Methodology**: Scanning the workspace in a manner which is continuous in all three directions.
- **Performance**: Eliminating redundant computations and implementing the approach of *growing from the inside* to improve the running time when compared to the existing algorithms.
- **Code re-usability**: Making the implementation generic enough to be used for any two and three degree-of-freedom manipulator.

1.4 Organization of the report

The Chapter 2 explains the singularities in the workspace of a parallel manipulator and the concept of SWZ and Maximal Surface of Revolution (referred to as MSoR hereafter). This is followed by the algorithms developed to compute the SWZ and MSoR for parallel manipulators described in Chapter 3. Chapter 4 presents the results and compares the performance of the algorithms with the algorithm presented in Karnam *et al.* (2016). The report concludes with Chapter 5 which presents the summary of the work done and scope for future work.

1.5 Conclusion

This chapter highlighted various approaches taken to avoid singularities and to compute zones in the workspace, of a parallel manipulator, which are free of singularities. The objectives for this work were presented at the end of the chapter.

CHAPTER 2

Safe Working Zone of a parallel manipulator

2.1 Introduction

The algorithms presented in this work find regions which are subsets of the workspace of a parallel manipulator satisfying certain conditions. Following are the definitions of two such regions:

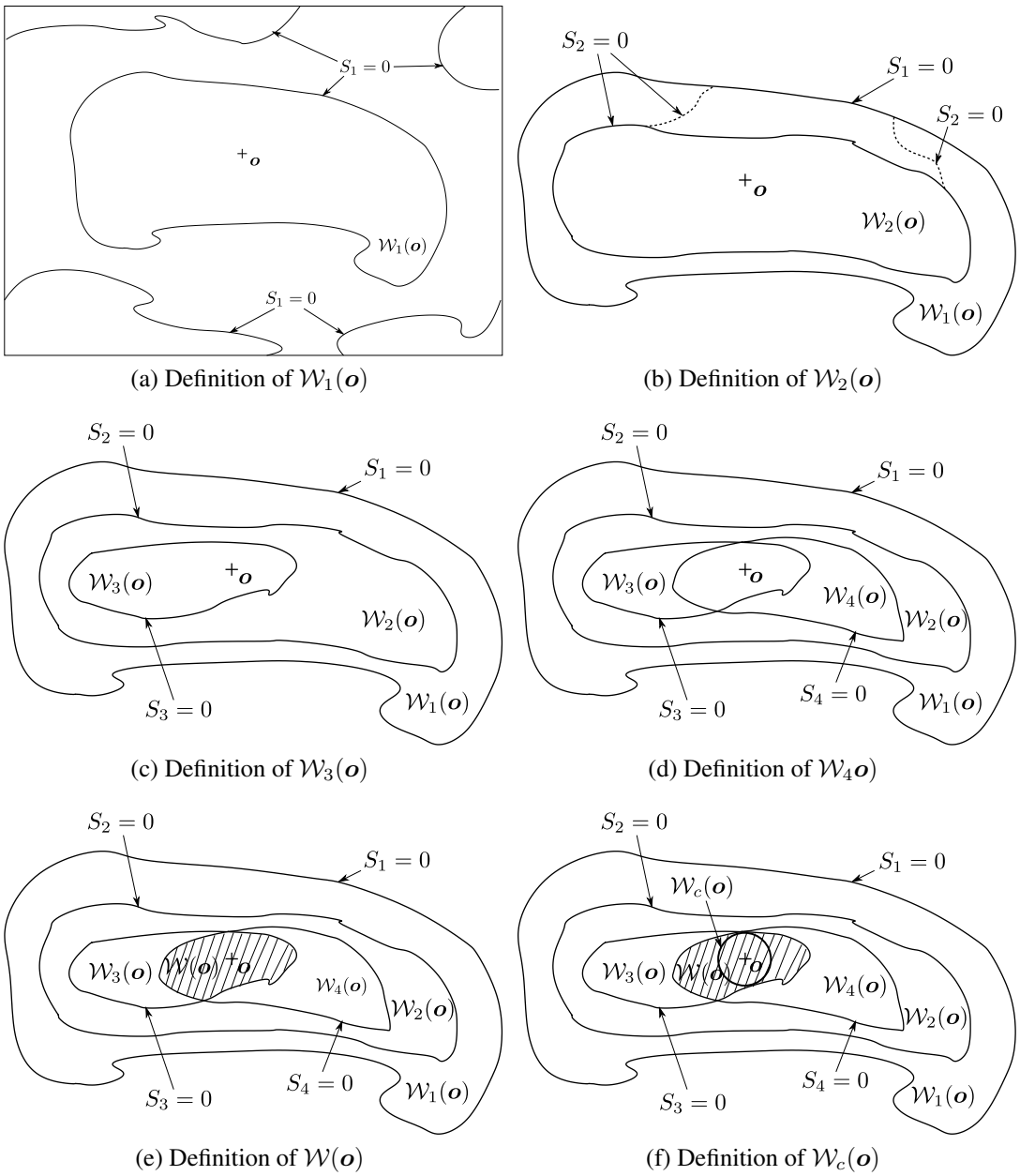
Definition 1 *The SWZ, $\mathcal{W}(\boldsymbol{o})$ of a parallel manipulator is defined as a subset of the workspace of the manipulator satisfying the following:*

1. *It is a* connected *set and contains the given point of interest $\boldsymbol{o}$;*
2. *It is contained entirely* inside *the workspace of the manipulator, i.e., it is free from loss-type singularities;*
3. *It is free from gain-type singularities;*
4. *There is no interference between the links at any point inside $\mathcal{W}(\boldsymbol{o})$;*
5. *At no point in $\mathcal{W}(\boldsymbol{o})$ does any joint violate its physical joint limits.*

This definition for the SWZ is proposed by Srivatsan and Bandyopadhyay (2014). In addition to the above mentioned definition, the SWZs considered in this work are convex and are hence denoted by $\mathcal{W}_c(\boldsymbol{o})$.

Definition 2 *The MSoR, $\mathcal{M}(\boldsymbol{o})$ for a parallel manipulator is defined as a subset of the workspace of the manipulator satisfying all the points mentioned in* Definition 1 *and in addition maintains the following:*

6. *It is bounded by the surface of revolution about a given central axis. This surface touches one or more of the boundaries of the region in the workspace, marked by any of the loss-type singularities, gain-type singularities, link interference and joint limits.*



(a) Definition of $\mathcal{W}_1(\boldsymbol{o})$

(b) Definition of $\mathcal{W}_2(\boldsymbol{o})$

(c) Definition of $\mathcal{W}_3(\boldsymbol{o})$

(d) Definition of $\mathcal{W}_4\boldsymbol{o})$

(e) Definition of $\mathcal{W}(\boldsymbol{o})$

(f) Definition of $\mathcal{W}_c(\boldsymbol{o})$

Figure 2.1: Definition of $\mathcal{W}(\boldsymbol{o})$ and $\mathcal{W}_c(\boldsymbol{o})$.
Source for the images: Karnam *et al.* (2016)

The requirements 2-5 in the Definition 1 and Definition 2 have a common characteristic: each define a subset of the workspace, which is bounded by the zero level-set of a corresponding boundary function (denoted henceforth by S_i). The following define the terminology associated with these boundary functions.

1. The zero level-set of the condition for loss-type singularity, defined by $S_1 = 0$, bounds the workspace, as seen in Fig. 2.1a. The subset of this workspace which contains $\boldsymbol{o}$ is denoted by $\mathcal{W}_1(\boldsymbol{o})$.

2. The gain-type singularity manifold is obtained by evaluating the condition for gain-type singularity, which is given by the solution set of $S_2 = 0$. The region $\mathcal{W}_2(\boldsymbol{o})$, which is inside $\mathcal{W}_1(\boldsymbol{o})$, and contains $\boldsymbol{o}$, and is free of gain-type singularities, is

bounded by the set of points defining the gain-type singularity manifold, as seen in Fig. 2.1b.

3. The region inside $\mathcal{W}_2(\boldsymbol{o})$ that includes $\boldsymbol{o}$, and is free of link interference as well, is denoted by $\mathcal{W}_3(\boldsymbol{o})$, and is bounded by the set satisfying $S_3 = 0$, as seen in Fig. 2.1c.

4. The solution set of $S_4 = 0$ which falls inside the region $\mathcal{W}_2(\boldsymbol{o})$, delimits $\mathcal{W}_4(\boldsymbol{o})$, i.e., the space containing $\boldsymbol{o}$ and free of joint-limit violations (see Fig. 2.1d).

As seen in Fig. 2.1e, the SWZ of the manipulator can be found as $\mathcal{W}(\boldsymbol{o}) = \bigcap_{i=1}^{4} \mathcal{W}_i(\boldsymbol{o})$.

The following sections give an outline of general procedure for finding the singularity functions S_1, S_2, S_3, and S_4 which describe the singularity manifolds of loss-type and gain-type and regions of link interference and joint limit violation respectively. From here on, these functions will together be called as singularity functions.

2.2 Loop closure equations

The loop-closure equations for a non-redundant parallel manipulator with n degrees-of-freedom can always be written in the form

$$\boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{x}) = \boldsymbol{0},\ \boldsymbol{g} \in \mathbb{R}^{m+n},\ \boldsymbol{\theta}, \boldsymbol{x} \in \mathbb{R}^n,\ \text{and } \boldsymbol{\phi} \in \mathbb{R}^m, \tag{2.1}$$

where, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{x}$ represent the actuated/active joint variables, unactuated/passive joint variables and end-effector pose variables respectively.

This set of equations, after eliminating appropriate variables, is used to obtain the equations for loss-type and gain-type singularities as shown in the following sections.

2.3 Loss-type singularity

The passive variables $\boldsymbol{\phi}$ can be eliminated from Eq. (2.1) to get n equations relating $\boldsymbol{x}$ and $\boldsymbol{\theta}$, which solve the inverse kinematics

$$\boldsymbol{h}(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{0}, \boldsymbol{h}, \boldsymbol{\theta}, \boldsymbol{x} \in \mathbb{R}^n. \tag{2.2}$$

Using the Implicit Function Theorem (IFT), the condition for loss-type singularity can be obtained from Eq. (2.2) as $S_1 = 0$, where

$$S_1 = \det\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta}}\right). \tag{2.3}$$

According to IFT, wherever $S_1 = 0$, the branches of inverse kinematic solutions meet, which signifies loss-type singularity (Gosselin and Angeles (1990)).

It has to be noted that the expression for S_1 involves $\boldsymbol{\theta}$, and hence evaluating it given the task-space variables $\boldsymbol{x}$ involves solving the inverse kinematics.

2.4 Gain-type singularity

The end-effector pose variables can be eliminated from Eq. (2.1) to obtain a set of n equations relating $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ as

$$\boldsymbol{\eta}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbf{0},\ \boldsymbol{\eta}, \boldsymbol{\phi} \in \mathbb{R}^m, \boldsymbol{\theta} \in \mathbb{R}^n \tag{2.4}$$

Invoking the IFT again, the condition for gain-type singularity can be stated as $S_2 = 0$ (Ghosal (2006)), where

$$S_2 = \det\left(\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\phi}}\right). \tag{2.5}$$

Again, the inverse kinematics along with Eq. (2.4) have to be solved to compute S_2 from $\boldsymbol{x}$, for the appropriate *working mode* (Macho *et al.* (2008)).

2.5 Physical limitations

The interference of the links of the manipulator is captured by the function S_3 as follows

$$S_3 = \begin{cases} 1 & \text{if two or more links interefere,} \\ 0 & \text{otherwise.} \end{cases}$$

The joint limits are modeled as a function S_4 , which is equal to 0 for a particular pose of the manipulator, if all the links are within their corresponding prescribed joint limits, and equal to 1 otherwise.

2.6 Numerical evaluation of singularity functions

The complete configuration space of the manipulator is $\boldsymbol{q} = \{\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{x}\}^\top$. While the evaluation of S_1 requires $\boldsymbol{\theta}$ and $\boldsymbol{x}$, and S_2 requires $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, the evaluation of S_3 and S_4 requires all the configuration space variables. We define a function to perform

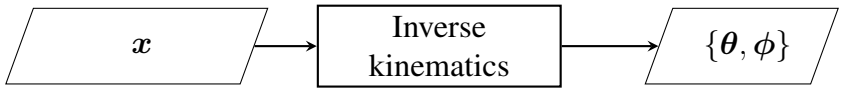


Figure 2.2: Inverse kinematics

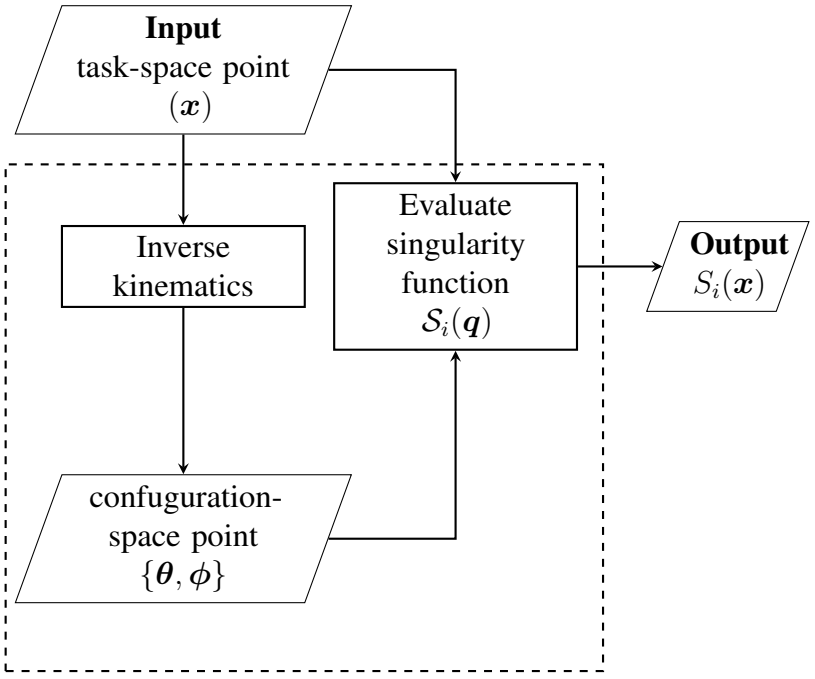


Figure 2.3: Computation of singularity functions

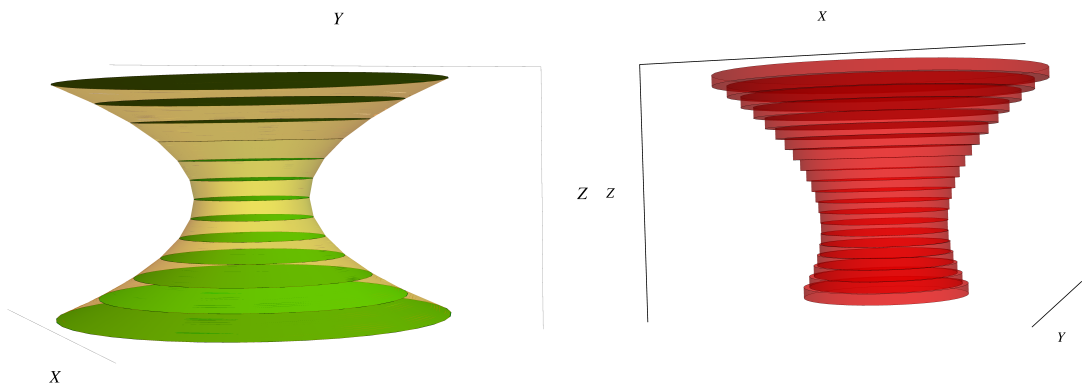
inverse kinematics as shown in Fig. 2.2 which gives the state of all the configuration variables. The functions $\mathcal{S}_i(q)$, $i = 1, 2, 3, 4$, take in the entire configuration of the

manipulator and evaluate the singularity functions $S_i(\boldsymbol{x})$. Fig. 2.3 shows the entire procedure of computing the S_i functions from the task-space variables $\boldsymbol{x}$.

2.7 SWZ and MSoR presented in this work

As mentioned earlier, the SWZ should be convex, but it can have any shape. For manipulators with three degrees-of-freedom with two of them being of same type (i.e., translational or rotational) and the third of a different type, a cylinder with its axis along the dimension representing the unique degree-of-freedom is a natural choice for the shape of the SWZ. For example, in a manipulator with two rotational and one translational degrees-of-freedom denoted by (α, β, z), the *natural* SWZ cylinder will have its axis along z. Likewise, in the case of one rotational and two translational degrees-of-freedom denoted by (x, y, α) , the SWZ cylinder will have its axis along α. All the algorithms presented here find the SWZ as a cylinder for the case of three degree-of-freedom manipulators, and as a circle for two degree-of-freedom manipulators.

The `rectangular-grid-scan` discretises the search domain along the Z-direction into 2D slices and fits the circle of maximum radius in every slice. The MSoR joining these circles, as shown in the Fig. 2.4a, is the output of the `rectangular-grid-scan` in the case of a three degree-of-freedom manipulator.



(a) MSoR obtained by `rectangular-grid-scan` (b) MSoR obtained by `maximal-fit`

Figure 2.4: MSoR

On the other hand, `maximal-fit` fits disks with small but finite length along the Z-direction instead of circles in planes. The MSoR given by `maximal-fit` is shown in Fig. 2.4b.

2.8 Conclusion

This chapter described the manifolds and regions in the workspace of a parallel manipulator which limit its size and which divide it into smaller regions. It outlines a general procedure to compute these functions for any parallel manipulator. The algorithms presented in the following chapter use these functions to compute the SWZ and MSoR of a parallel manipulator.

CHAPTER 3

Proposed algorithms for finding the SWZ

3.1 Introduction

This chapter describes the proposed algorithms `quick-fit` and `maximal-fit` to find the SWZ and MSoR of a parallel manipulator. It starts by an overview of the existing algorithm, `rectangular-grid-scan`, to find the MSoR of a parallel manipulator, followed by mentioning its limitations. Overcoming the limitations of the `rectangular-grid-scan` and improving the running time serve as the major objectives for the development of `quick-fit` and `maximal-fit`, details of which are presented later in this chapter.

3.2 Overview of the existing algorithm

The existing algorithm, `rectangular-grid-scan`, proposed by Karnam *et al.* (2016), discretises the space into a number of two dimensional slices parallel to the XY- plane. It uses a rectangular two-dimensional grid on each slice to find the contours of the singularity functions, and subsequently finds the maximal circle, centered at a given point, and free of all the singularities and physical issues. The central point for every two dimensional slice is along a single line and hence by joining all the circles on consecutive slices one gets a surface of revolution, which is the MSoR for the manipulator.

There is a natural hierarchy in the singularity functions, i.e., loss-type singularity generally bounds the workspace of the manipulator while the presence of gain-type singularity inside the workspace divides it into singularity-free zones. Only when one finds a zone free from the kinematic singularities, i.e., loss- and gain-type, is one interested in investigating the physical issues of link interference and joint limits inside that zone. Also, is it more efficient to check for physical issues in a smaller region

than in the entire search space, as these are more expensive to compute compared to the kinematic singularities. The `rectangular-grid-scan` makes use of these facts and computes the contours of the singularity functions hierarchically, as show in Fig. 3.2, starting from finding $S_1 = 0$, followed by computing $\mathcal{W}_{c_1}(\boldsymbol{o})$ and so on, going down to $S_4 = 0$ as the scan region is shrunk at every stage. The details of `rectangular-grid-scan` can be found in Karnam *et al.* (2016).

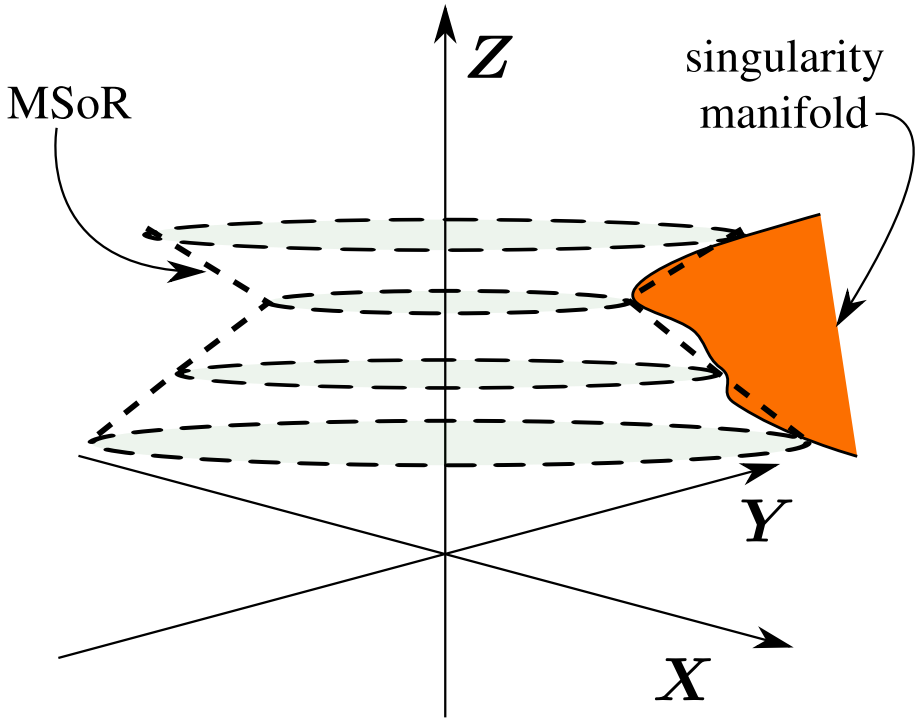
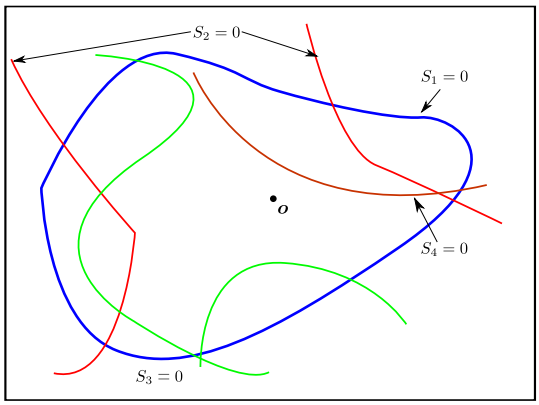


Figure 3.1: Error due to discontinuous scan in z-direction. The 3-dimensional object obtained by joining the 2-dimensional slices can have intersections with the singularity manifold.

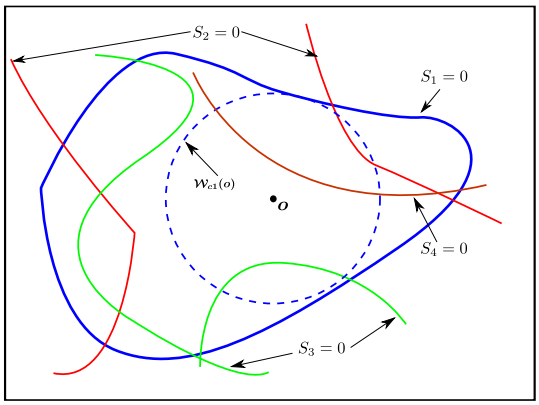
3.2.1 Limitations

The two major limitations of `rectangular-grid-scan` are as follows:

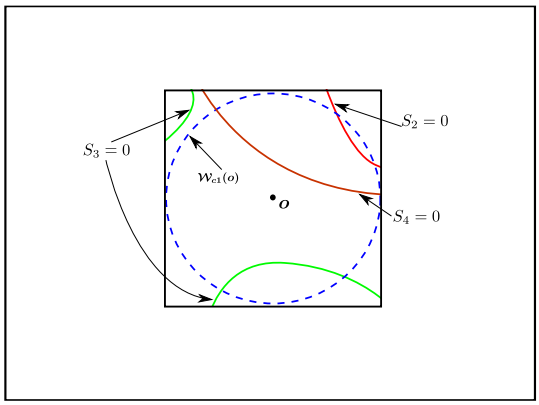
- Redundant computation of inverse kinematics, once for computing every singularity function S_i
- For manipulators with three degrees of freedom, the scan in the third dimension, i.e., in the Z-direction is discontinuous and the MSoR obtained by connecting the 2D slices can be erroneous, as shown in Fig. 3.1



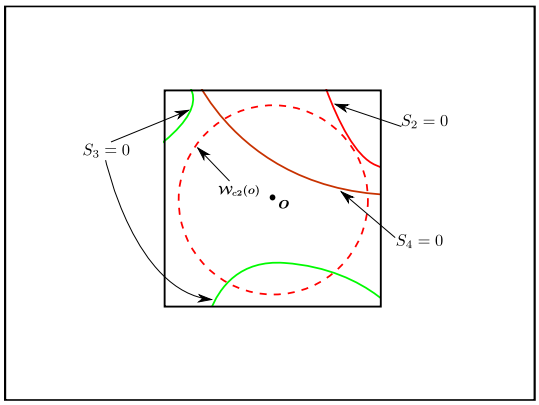
(a) Workspace with singularities



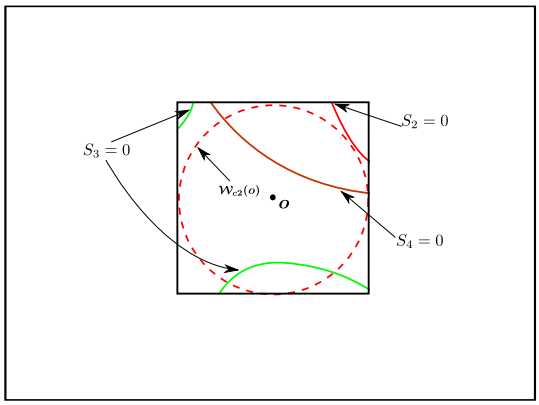
(b) $\mathcal{W}_{c_1}(\boldsymbol{o})$



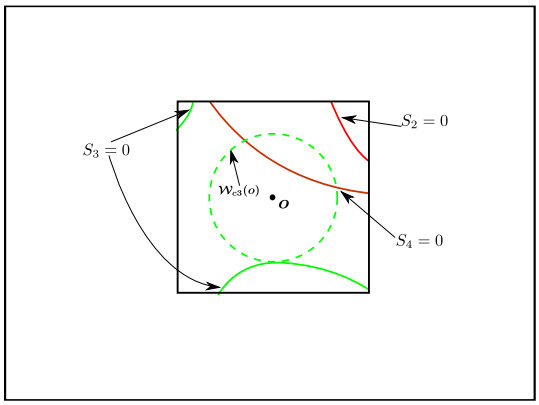
(c) Reduced search space



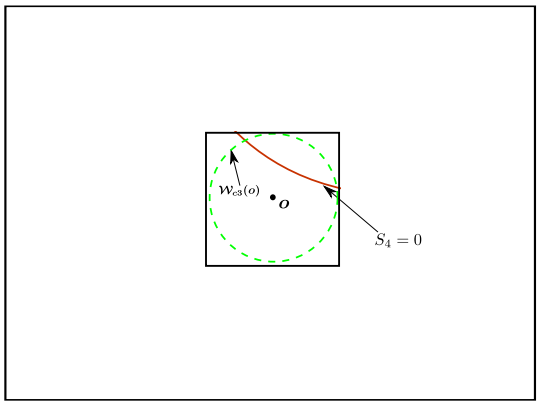
(d) $\mathcal{W}_{c_2}(\boldsymbol{o})$



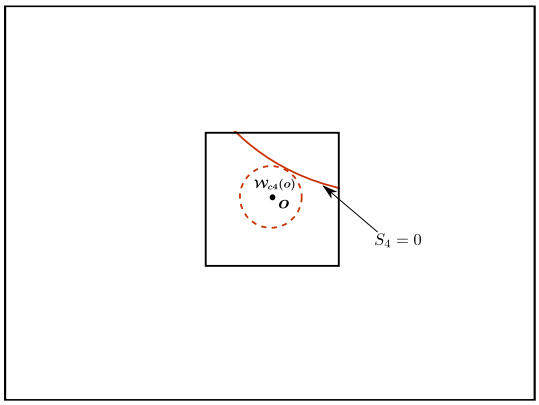
(e) Reduced search space



(f) $\mathcal{W}_{c_3}(\boldsymbol{o})$



(g) Reduced search space



(h) The SWZ, $\mathcal{W}_c(\boldsymbol{o}) = \mathcal{W}_{c_4}(\boldsymbol{o})$

Figure 3.2: Geometric explanation of `rectangular-grid-scan`

3.3 Central approach used by the proposed algorithms

The idea behind these new algorithms for finding the $\mathcal{W}_c(\boldsymbol{o})$ is to grow from inside, with a predefined convex shape, instead of scanning the entire search space to find $\mathcal{W}_{ci}(\boldsymbol{o})$ with $i = 1, \cdots, 4$ and getting $\mathcal{W}_c(\boldsymbol{o})$ subsequently.

The next section describes in detail, the key concept of *kinematic node*, introduced to reduce the number of redundant computations, and the subsequent sections describe the two algorithms which use this concept.

3.3.1 Kinematic node

Finding the boundaries of SWZ involves computing the singularity functions, $S_i(\boldsymbol{x})$, at various points $\boldsymbol{x}$ in the task-space. As shown in the Fig. 2.3, computing the singularity functions involves finding the configuration-space coordinates from the task-space coordinates using inverse kinematic computation. Treating each of the singularity functions separately results in the redundant inverse kinematics computation, which can be cut down if all the singularity functions are tied together with inverse kinematic data.

This has been achieved in the `C++` implementation by composing the task-space coordinates (`TtaskSpace-Coordinate`), manipulator pose (`TPose`) and singularities (`TSingularities`) into a single entity called *kinematic node* (`SNode`) as shown in the Fig. 3.3. Here, `TtaskSpaceCoordinate` contains the coordinates of the point in the task-space, `TPose` contains inverse kinematic data and the physical dimensions of the manipulator and `TSingularities` contains the singularity functions. This composition allows the algorithm to be used for any manipulator with two or three degrees of freedom, achieving a balance between adaptability and efficiency.

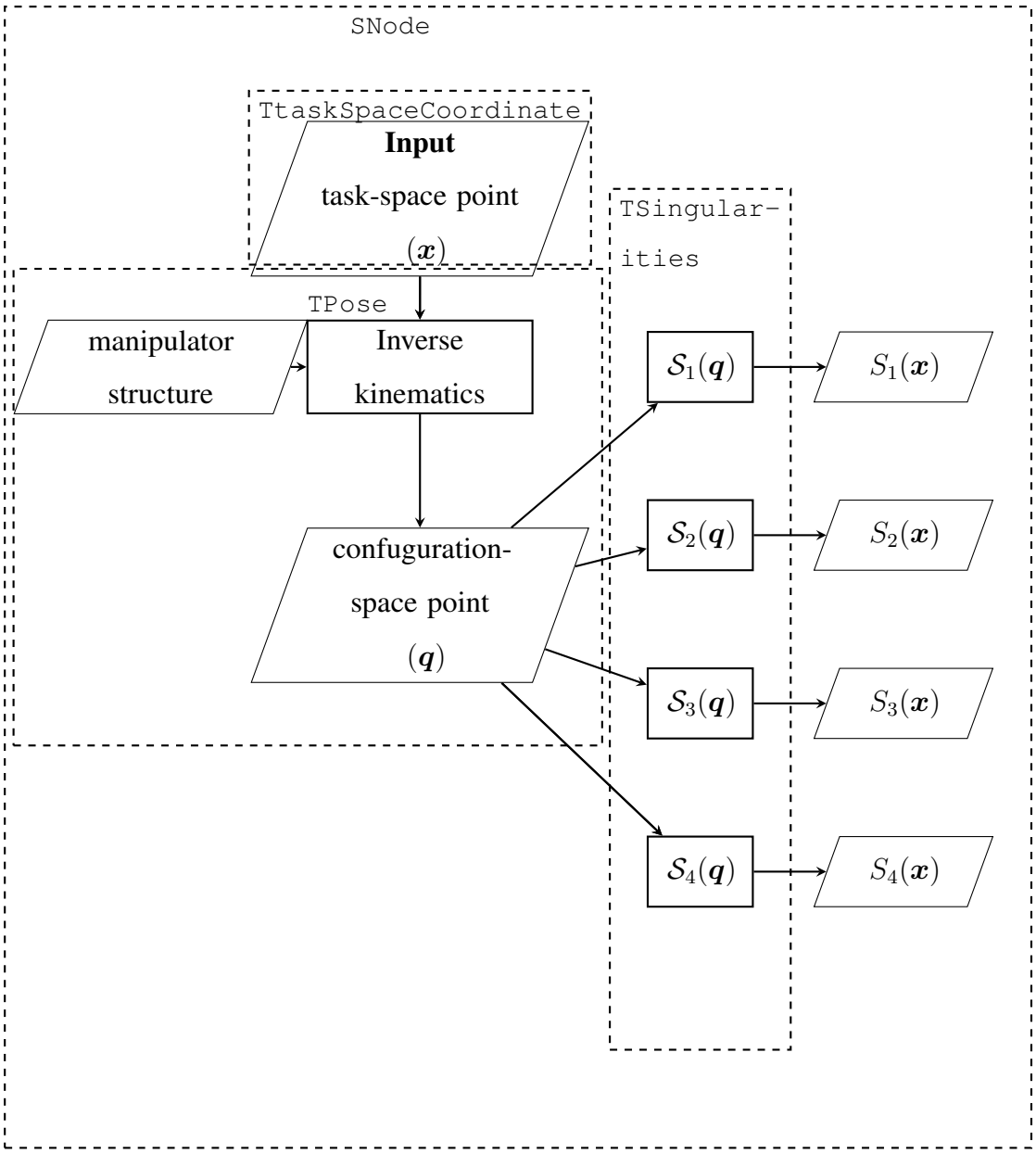


Figure 3.3: Computation of singularity functions - C++ object model

The values of $S_i(\boldsymbol{x})$ at a node n with task-space coordinate $\boldsymbol{x}$ are assigned corresponding state s_i as

$$s_1(\boldsymbol{x}) = \begin{cases} \texttt{white} & \text{if } S_1(\boldsymbol{x}) \leq 0 \\ \texttt{black} & \text{if } S_1(\boldsymbol{x}) > 0 \end{cases} \qquad s_2(\boldsymbol{x}) = \begin{cases} \texttt{white} & \text{if } S_2(\boldsymbol{x}) \leq 0 \\ \texttt{black} & \text{if } S_2(\boldsymbol{x}) > 0 \end{cases}$$

$$s_3(\boldsymbol{x}) = \begin{cases} \texttt{white} & \text{if } S_3(\boldsymbol{x}) = 0 \\ \texttt{black} & \text{if } S_3(\boldsymbol{x}) = 1 \end{cases} \qquad s_4(\boldsymbol{x}) = \begin{cases} \texttt{white} & \text{if } S_4(\boldsymbol{x}) = 0 \\ \texttt{black} & \text{if } S_4(\boldsymbol{x}) = 1 \end{cases}$$

Definition 3 *Any two* kinematic nodes n_1 *and* n_2 *are said to be* similar *if all the singularity states* s_i *of all the singularity functions at* n_1 *are same as the corresponding singularity states of that at* n_2 .

The following sections elaborate the algorithms which make use of *kinematic nodes* and their *similarity* to find the SWZ.

3.4 Algorithm 1 - `quick-fit`

Given the range (z_{min}, z_{max}) in Z-direction and a reference *kinematic node* n_o with task-space point $\boldsymbol{o}$, which is assumed to be inside the SWZ and is usually the geometric center of the workspace of the manipulator, the `quick-fit` algorithm fits a cylinder with maximum radius inside the singularity free workspace of the manipulator, with its end points as z_{min} and z_{max} in the axial direction. This cylinder is the SWZ of the manipulator under consideration.

The next subsection gives the geometric overview of the algorithm which is then followed by the details of the algorithm and its features.

3.4.1 Geometric overview of the algorithm

As shown in Fig. 3.4, a cylindrical shell, initialized between z_{min} and z_{max}, with zero radius, expands in the radial direction till it intersects any of the singularity surfaces.

Every point on the cylinder is a *kinematic node* which is compared to the reference node n_o. The cylinder stops expanding when it finds that the current node is not *similar* to the reference node, which in turn means that a boundary defined by one of the singularity functions has been reached at the current point.

3.4.2 Details

The algorithm consists of three basic steps, namely *circumferential scan*, *axial sweep* and *radial expand*, performed in a loop till a *kinematic node* finds a singularity.

Fig. 3.5 shows a circumferential scan in progress. The circle is traversed in the circumferential direction, adding a *kinematic node* at every step and checking if it is *similar* to the reference node n_o, as outlined in Algorithm 1.

Algorithm 1 circumferential scan

```
1: function CIRCUMFERENTIALSCAN
2:     δθ ← δs/r        ▷ keeps the arc length between two nodes constant as the radius expands
3:     θ ← 0
4:     scanComplete ← true
5:     for θ < 2π do
6:         if n(r, θ, z) not similar to n_o then
7:             scanComplete ← false
8:             break
9:         θ ← θ + δθ
10:    return scanComplete
```

The axial sweep, outlined in Algorithm 2, uses *circumferential scan* at every step and incrementally adds new shells to the scanned region, as shown in Fig. 3.6.

Algorithm 2 axial sweep

```
1: function AXIALSWEEP
2:     z ← z_min
3:     sweepComplete ← true
4:     for z < z_max do
5:         if CIRCUMFERENTIALSCAN(·) is not true then
6:             sweepComplete ← false
7:             break
8:         z ← z + δz
9:     return sweepComplete
```

Finally, `quick-fit`, outlined in Algorithm 3, uses *axial sweep* and *radial expand* (line 7 in Algorithm 3), keeping constant arc length between two consecutive nodes as the radius expands, and finds the SWZ.

Algorithm 3 `quick-fit`

```
1: function QUICKFIT
2:     z_min, z_max, r_max, δz, δr, δs, refNode, ManipulatorStructureData ← inputs
3:     r ← 0
4:     for r ≤ r_max do
5:         if AXIALSWEEP(.) is not true then
6:             return r − δr
7:         r ← r + δr                                   ▷ Radial Expand
8:     return r
```

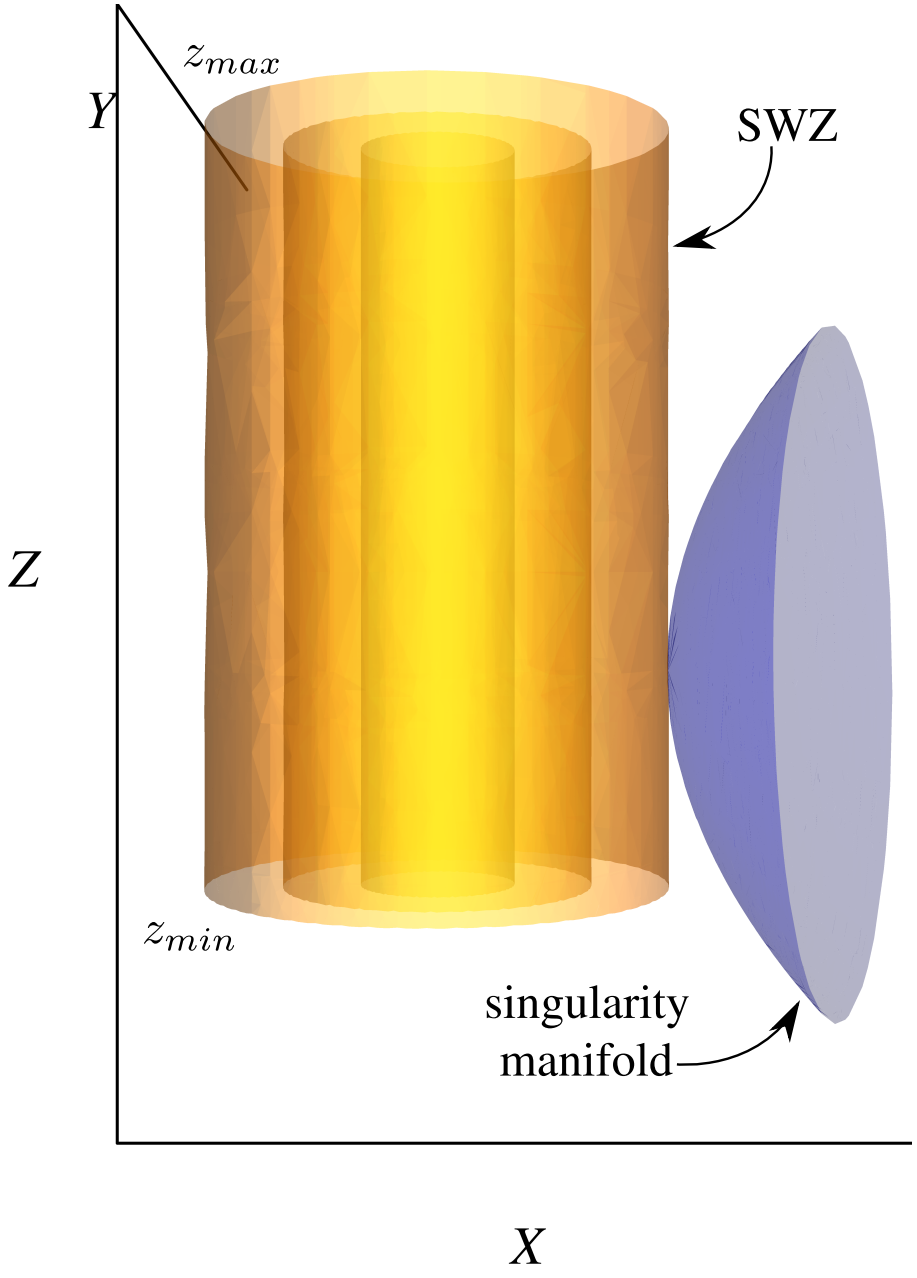


Figure 3.4: Expanding cylinders in `quick-fit` algorithm

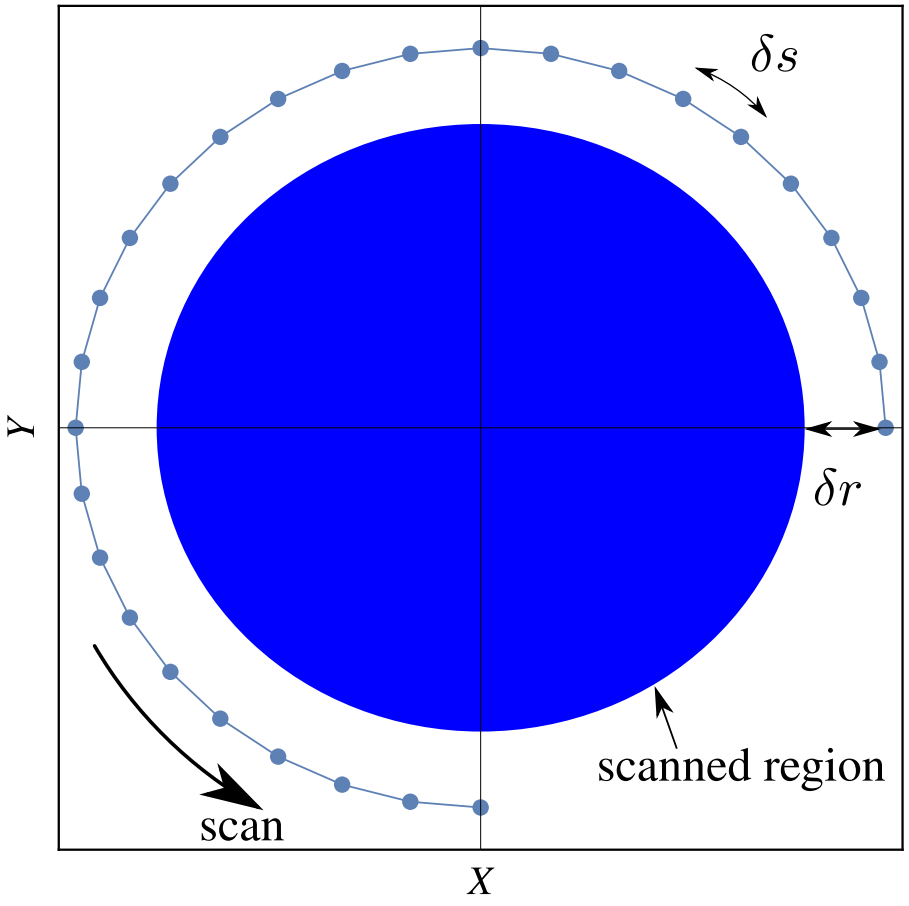


Figure 3.5: Circumferential scan

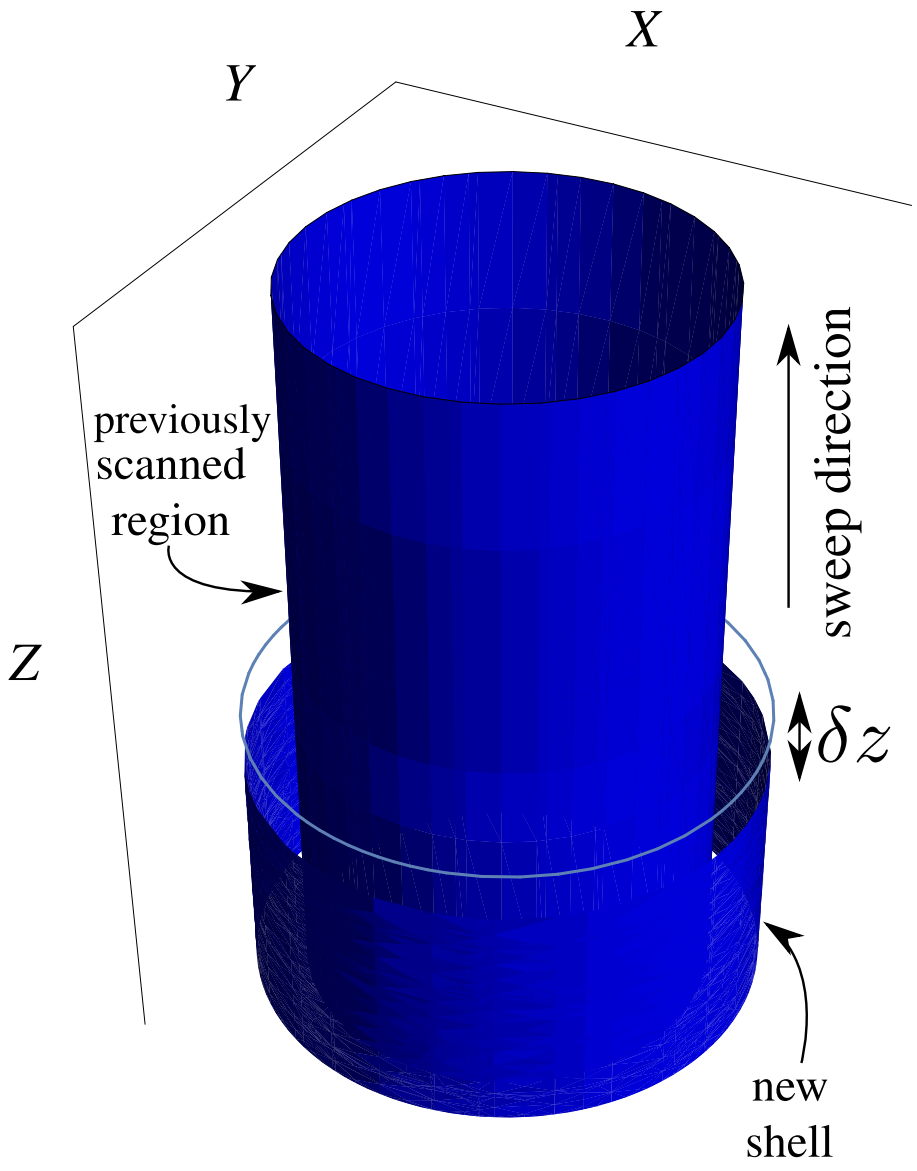


Figure 3.6: Axial sweep

3.4.3 Features of the algorithm

- Extremely fast and ideal for iterative design problems
- No static memory overhead
- No redundant computations
- Continuous in all three dimensions
- Any convex shape other than a circle can also be chosen as the cross-section of the SWZ.

3.5 Algorithm 2 - `maximal-fit`

Given the range (z_{min}, z_{max})in Z-direction and a reference *kinematic node* n_o , which is assumed to be inside the SWZ (usually the geometric center for the manipulator) the `maximal-fit` algorithm finds the MSoR, with a fixed discretisation δz in the Z-direction, inside the singularity-free workspace of the manipulator, which can then be used to obtain an appropriate SWZ as per the design requirement.

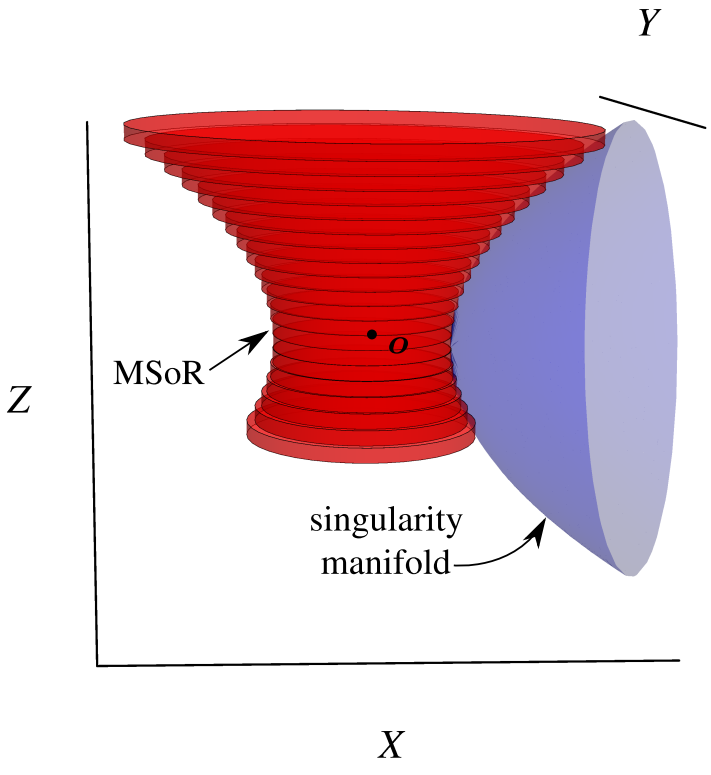


Figure 3.7: Maximal Surface of Revolution

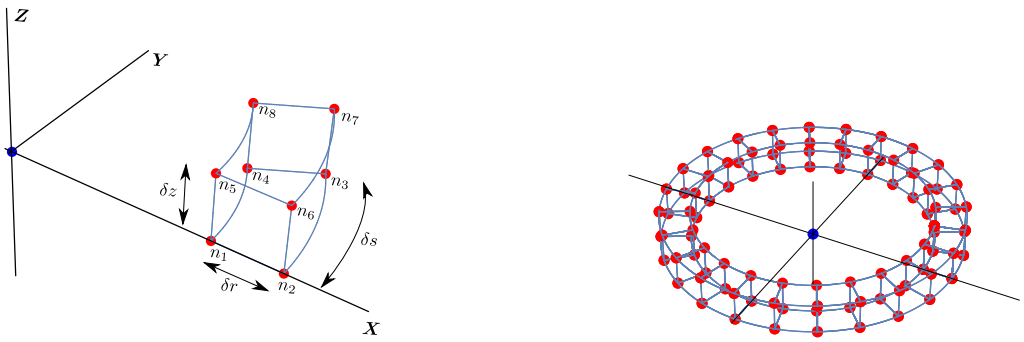
The following subsections give the geometric overview of the algorithm followed by its key features.

3.5.1 Geometric overview of the algorithm

As shown in Fig. 3.7, the algorithm fits disks, with a resolution of δz in the Z-direction, inside the workspace of the manipulator. The disks are scanned one after the other, ensuring that no redundant computation occurs.

3.5.2 Details

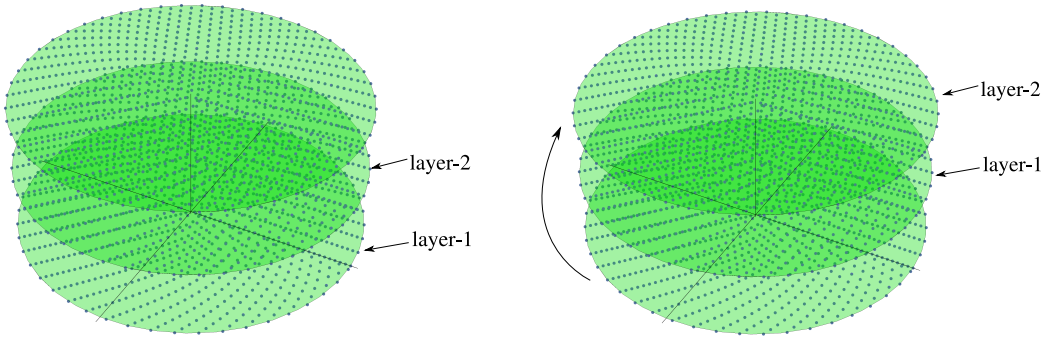
The basic three-dimensional geometric element formed using the *kinematic nodes* is the deformed parallelepiped as shown in the Fig. 3.8a. A set of such deformed parallelepipeds forms a ring as shown in Fig. 3.8b, a set of which, with different radii, forms one disk of the MSoR, as shown in Fig. 3.10.



(a) Deformed parallelepiped (b) Ring of nodes

Figure 3.8: Elements of `maximal-fit` algorithm

The algorithm was implemented in two ways to bring out the advantage of utilizing *kinematic node* correctly. Implementation one, called `maximal-fit -1` constructs, checks and discards each parallelepiped along with its nodes as it moves along and checks a ring, while implementation two, called `maximal-fit -2`, constructs two layers of nodes, as shown in Fig. 3.9a, required to form one disk, and then traverses along



(a) Initial layers of nodes (b) Swapping the layers after sacnning the disk

Figure 3.9: Swapping of layers in `maximal-fit -2`

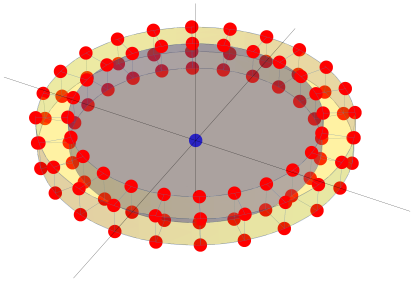


Figure 3.10: Outer ring of nodes expanding the disk

the rings and fits the disk of appropriate radius. Once a disk is scanned, the memory used for its bottom layer is used to construct the top layer of the next disk as shown in Fig. 3.9b. This strategy reduces redundant computations to absolute zero, taking complete advantage of the *kinematic node* concept and gives the benefit of full static memory allocation for a three dimensional grid with the memory cost of allocating just two layers.

3.5.3 Features of the algorithm

Following are the main features of `maximal-fit` which distinguishes it from `rectangular-grid-scan`:

- Achieves the benefit of the static memory allocation of a full three dimensional grid with only two dynamic layers of nodes, which for a 1 m scan in Z-direction with 1 mm resolution reduces the memory usage to 0.2 % of the memory which would have been used if the entire grid were to be allocated statically.
- Explores a large region of the workspace; ideal for initial analysis of a new manipulator.
- Overcomes the major limitation of `rectangular-grid-scan`, i.e., continuity in the Z-direction as illustrated by Fig. 3.11.
- Brings down the unnecessary computations to absolute zero by growing from inside and using the concept of *kinematic node*.
- The memory requirement only depends on the maximum radius of search in XY-plane and is completely independent of the range and resolution of search in Z-direction.

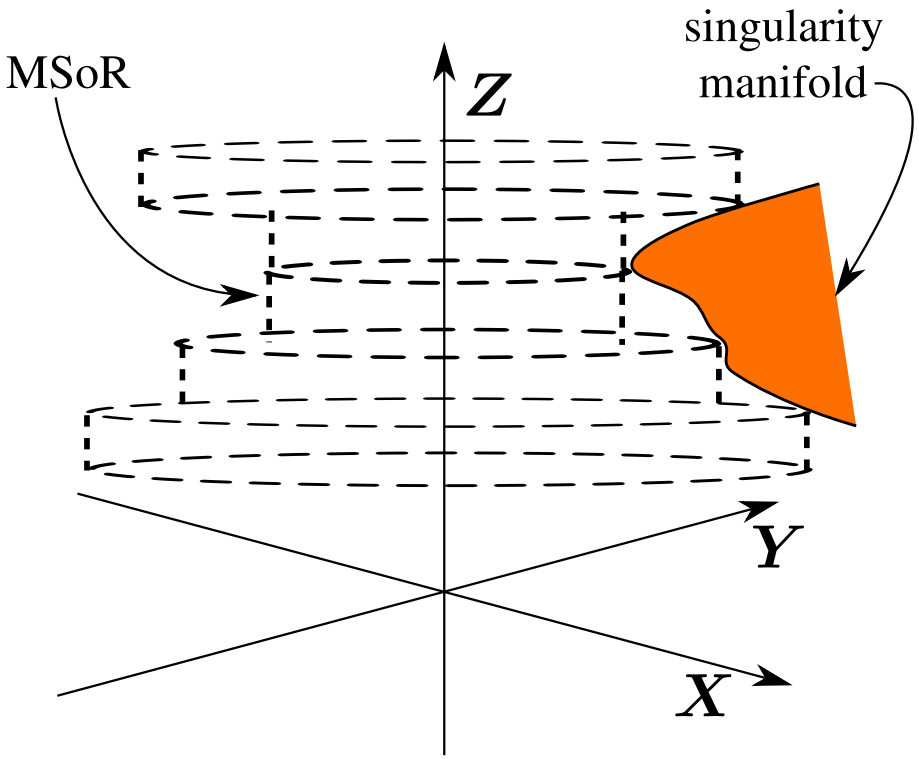


Figure 3.11: Continuity of `maximal-fit` in z-direction

3.6 Conclusion

This chapter presented the `quick-fit` and `maximal-fit` as new algorithms for computing the SWZ and MSoR of a parallel manipulator, along with their key features.

The next chapter compares the performance of the three algorithms, `rectangular-grid-scan`, `quick-fit` and `maximal-fit`, when applied to planar and spatial

parallel manipulators, and quantifies the features of `quick-fit` and `maximal-fit` mentioned in this chapter.

CHAPTER 4

Results and Comparisons

4.1 Introduction

This chapter quantifies the difference in performance achieved by `quick-fit`, `maximal-fit` and `rectangular-grid-scan` when applied to a planar five-bar and a 3-RRS manipulator.

First, the performances of `quick-fit` and `rectangular-grid-scan` are compared for the case of a five-bar. Thereafter, a detailed comparison of both `quick-fit`, `maximal-fit`, and `rectangular-grid-scan` is presented for the case of a 3-RRS manipulator, along with the call graph analysis.

4.2 SWZ for a five-bar mechanism

The Fig. 4.1 shows a symmetric five-bar mechanism, which is a 2-degree-of-freedom, planar parallel manipulator whose task-space coordinates are $\boldsymbol{x} = (x, y)^\top$, active joint coordinates are $\boldsymbol{\theta} = (\theta_1, \theta_4)^\top$, and passive joint coordinates are $\boldsymbol{\phi} = (\phi_2, \phi_3)^\top$. The gain-type and loss-type singularity conditions were derived symbolically in the configuration space $(\boldsymbol{\theta}, \boldsymbol{\phi})$ (refer Karnam *et al.* (2016) for detailed derivation) and used in the numerical routines to compute $S_1(\boldsymbol{x})$ and $S_2(\boldsymbol{x})$.

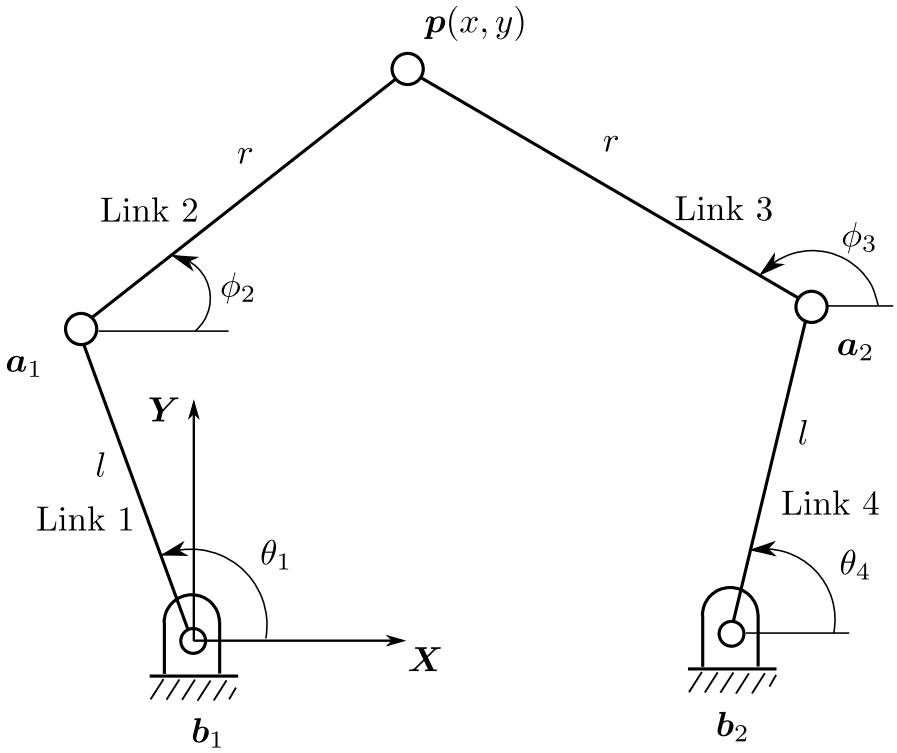


Figure 4.1: A symmetric five-bar mechanism

The SWZ for the five-bar with the dimensions shown in Table 4.1, where l_0 is the distance between the points on the ground, $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$, is taken to be a circle centered at $(1/2, 3/5)$ in the task-space as shown in the Fig. 4.2 . Here, the links have been assumed to be in separate planes and hence S_3 and S_4 need not be considered.

Link	l_0	l	r
Length (m)	1	1/2	3/5

Table 4.1: Physical dimensions of the five-bar

The following section compares the performances of `quick-fit` and `rectangular-grid-scan` algorithms applied to a five-bar. The `maximal-fit` is specific to manipulators with three degrees of freedom and hence cannot be applied to a five-bar.

4.2.1 Results

Both the routines used the same functions for computing S_1 and S_2 and hence the difference in running time can be completely attributed to the change in algorithm. The boundaries for the *rectangular grid scan*, considering the link lengths, were $x \in$

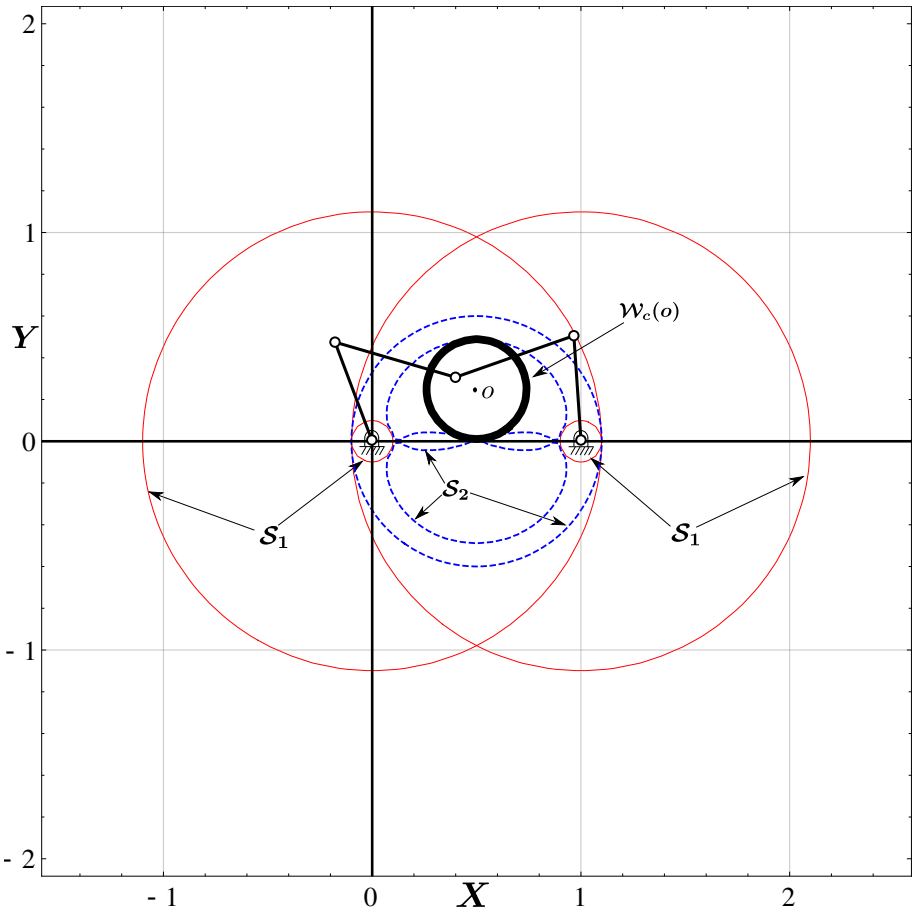


Figure 4.2: SWZ of the five-bar

$(-1/2, 3/2)$ and $y \in (0, 11/10)$, in meters, with the resolution of 1 mm in each direction. The boundaries for `quick-fit` were, $r_{max} = 1$, centered at $(1/2, 1/4)$ in meters, with the resolution of 1 mm in radial as well as circumferential direction.

	Averaged over (runs)	**Average run time (s)**	**SWZ radius (m)**
rectangular grid scan	100	0.430	0.239
quick-fit	100	**0.035**	0.239

Table 4.2: Running times for `quick-fit` and `rectangular-grid-scan` used on a five-bar

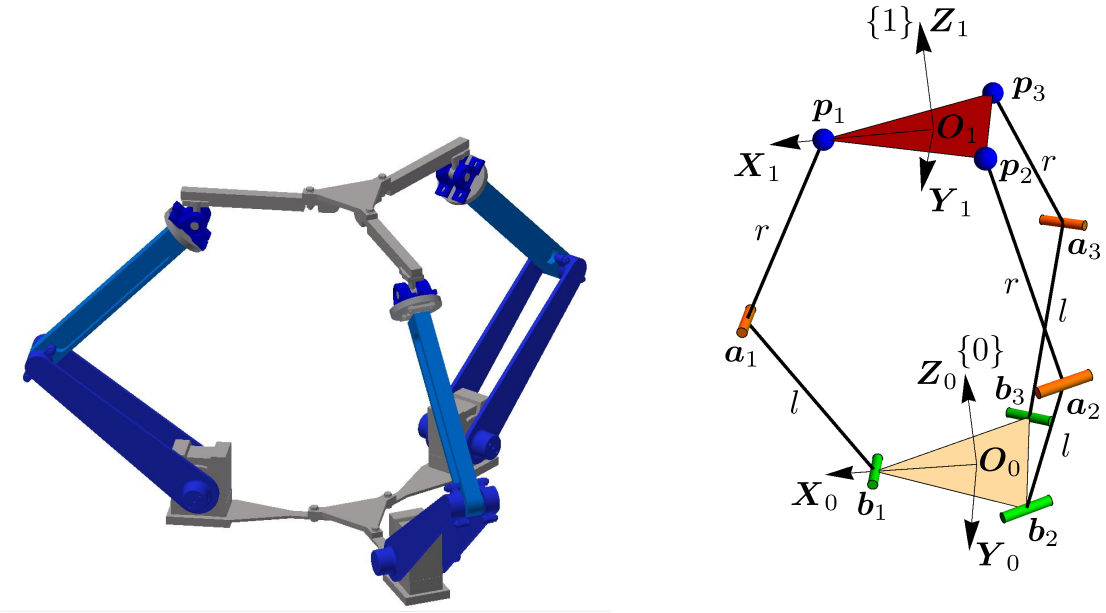
As shown in Table 4.2, the runtime[1] of `quick-fit` is one-tenth that of `rectangular-grid-scan`. This reduction is due the number of evaluations of S_1 and S_2 put together, which is 4406202 for `quick-fit` while it is 182134 for the `quick-fit`, a 24 times

[1]The code was complied using gcc v4.8.4 and was run on a single core of `Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz`

reduction.

4.3 SWZ for a 3-RRS manipulator

The 3-RRS is a 3-degree-of-freedom parallel manipulator consisting of three identical RRS legs, where R, R, and S represent the active/actuated rotary joint, passive rotary joint and passive spherical joint, respectively, as shown in Fig. 4.3a .



(a) CAD model (b) Kinematic sketch

Figure 4.3: The 3-RRS manipulator
Source for the images: Patel *et al.* (2016)

The manipulator consists of two triangular platforms, one fixed at the base and the other as the moving end-effector with circumradii b and a respectively.

The manipulator has three degrees of freedom, which are roll, pitch, and heave. The end-effector pose in the task-space of the manipulator is parameterised as $\boldsymbol{x} = \{\alpha, \beta, z\}$ while the active and passive joint variables are represented as $\boldsymbol{\theta} = \{\theta_1, \theta_2, \theta_3\}$ and $\boldsymbol{\phi} = \{\phi_1, \phi_2, \phi_3\}$ respectively. Here, (α, β) are the Euler angles following the convention $\boldsymbol{R}_{ZXY}(\gamma, \beta, \alpha)$.

4.3.1 Singularity conditions

The gain-type and loss-type singularity conditions for the manipulator are derived from the loop closure equations as discussed in Sections 2.3, and 2.4. These expressions, along with inverse kinematic equation, were used to form the S_1 and S_2 functions. The physical issues of joint limits and link interference have been clubbed together under S_3 . The details of the derivation of S_1 , S_2 and S_3 can be found in Patel *et al.* (2016). It is worthwhile to note that computation of S_3 involves checking for collisions of bounding boxes in three dimensional space, and hence is the most expensive of all the singularity functions, as seen in all the function call graphs in Section 4.3.3.

4.3.2 The SWZ

This subsection presents the SWZ and MSoR of the 3-RRS manipulator shown in the Fig. 4.3 whose dimensions are shown in the Table 4.3.

Link	a	l	r	b
Length (m)	0.550	0.700	0.775	0.550

Table 4.3: Physical dimensions of the 3-RRS manipulator

As mentioned earlier in Section 2.7, the convex shape chosen for the SWZ is a cylinder. Fig. 4.4 shows the radii of the surfaces of revolution $\mathcal{M}_i(\boldsymbol{o})$ w.r.t. S_1 , S_2 and S_3 hierarchically, i.e., surface of revolution $\mathcal{M}_1(\boldsymbol{o})$ is found for S_1 , then inside it, $\mathcal{M}_2(\boldsymbol{o})$ is found for S_2, followed by $\mathcal{M}_3(\boldsymbol{o})$ for S_3 . Hence the region inside $\mathcal{M}_3(\boldsymbol{o})$ is free from all the singularities, but it is not convex. Fig. 4.5 shows the contours of S_1, S_2, and S_3 with the circles, which are the part of the surfaces of revolution shown in Fig. 4.6, for a particular Z-slice.

Cylinders are fit in the desired Z-range to find the SWZ, which, by definition, is convex, as shown in the Fig. 4.4 by C_1 and C_2.

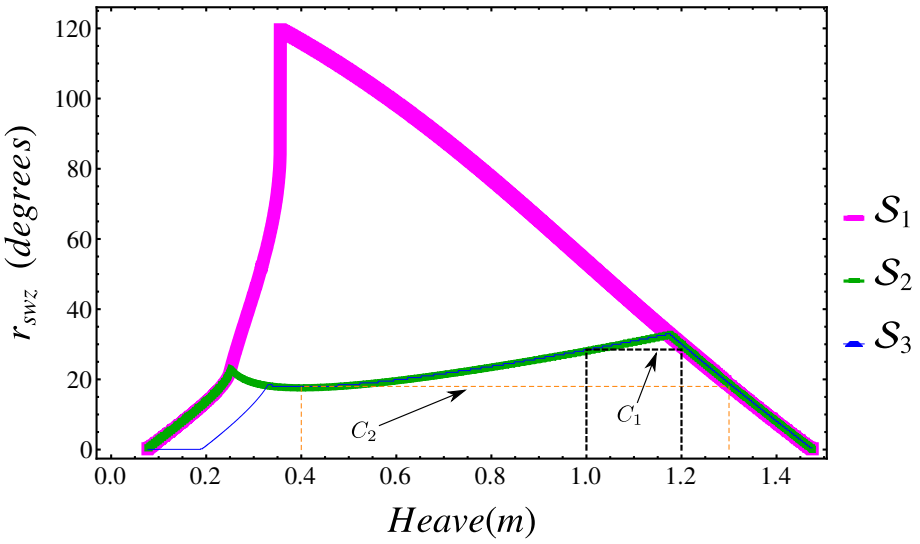


Figure 4.4: SWZ of the 3-RRS manipulator

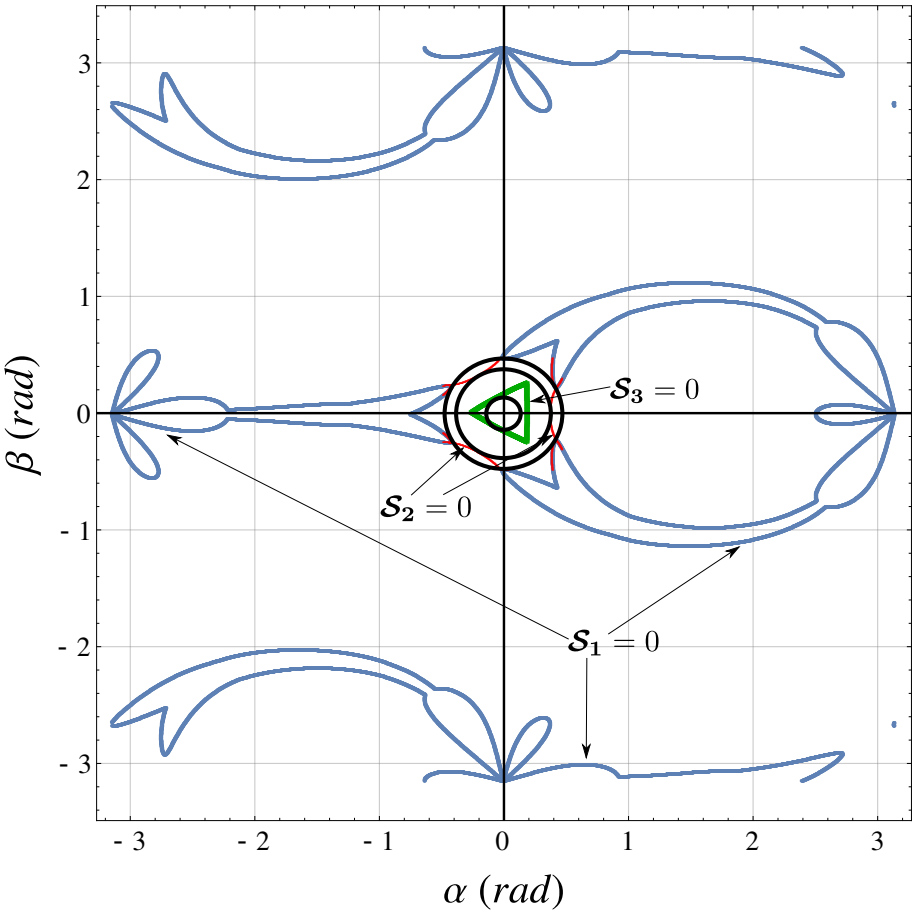


Figure 4.5: SWZ of the 3-RRS manipulator

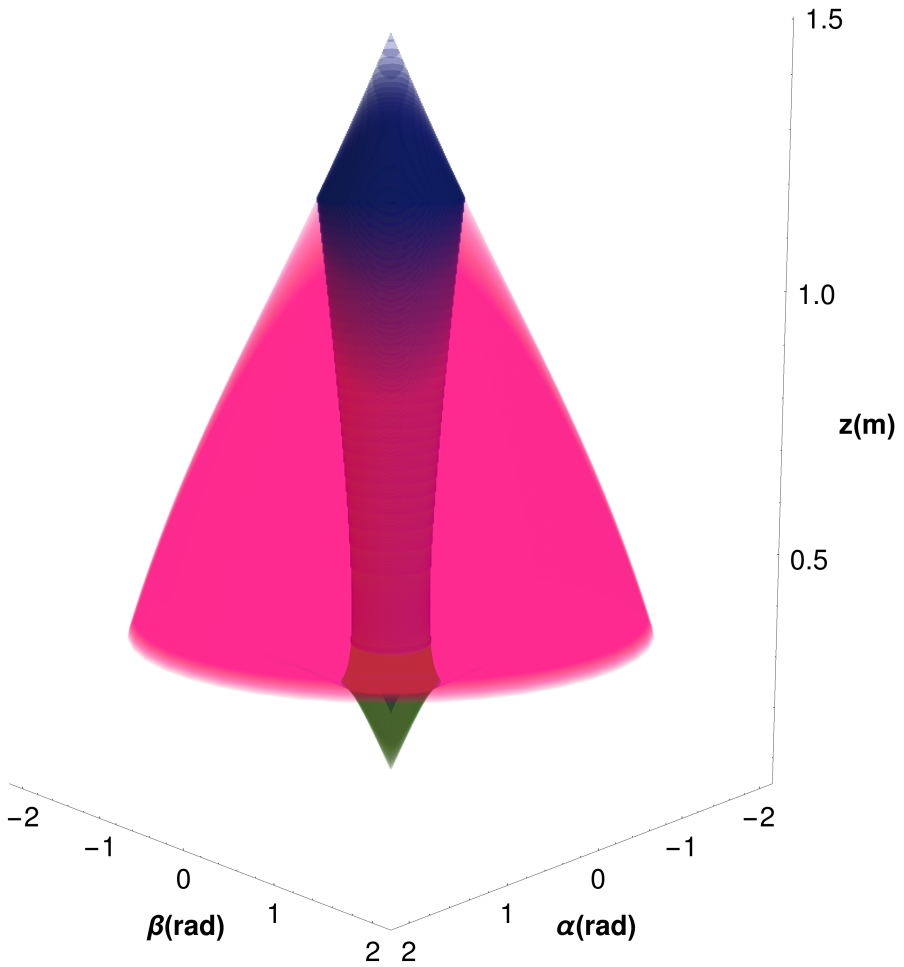


Figure 4.6: Surfaces of revolution w.r.t S_1, S_2, and S_3, for the 3-RRS manipulator

The following subsection presents the analysis and comparison of the `quick-fit` and `maximal-fit` algorithm with the `rectangular-grid-scan`.

4.3.3 Results

It has to be emphasized here that the nature of the output given by `quick-fit` and `maximal-fit` is different. While `maximal-fit` (and also `rectangular-grid-scan`) gives the MSoR of the manipulator, `quick-fit` gives the SWZ itself with the given Z-range, so it is not fair to compare the two mutually, but `maximal-fit` and `rectangular-grid-scan` can be compared in every aspect.

Analysis of quick-fit

Table 4.4 shows the radius of SWZ of the 3-RRS manipulator for various heave ranges, centered around $z = 0.8$. Fig. 4.7 and Fig. 4.8 show the plots of running times[2] for `quick-fit` as well as `rectangular-grid-scan` with respect to the heave range (input) and SWZ radius (output).

Sr. no.	z_{min} (m)	z_{max} (m)	r(rad)	run time for `rectangular-grid-scan` (s)	run time for `quick-fit` (s)
1	0.75	0.85	0.383	29.77	15.46
2	0.70	0.90	0.366	59.14	28.22
3	0.65	0.95	0.349	89.99	38.45
4	0.60	1.00	0.331	119.86	46.64
5	0.55	1.05	0.314	153.04	51.99
6	0.50	1.10	0.314	185.81	61.44
7	0.45	1.15	0.296	220.03	64.87
8	0.40	1.20	0.296	256.6	73.21
9	0.35	1.25	0.296	287.38	85.49
10	0.30	1.30	0.226	313.59	55.73
11	0.25	1.35	0.104	325.69	14.64
12	0.20	1.40	0.0174	337.08	01.03

Table 4.4: Radius of SWZ of 3-RRS manipulator for various heave ranges

[2]The code was run on a single core of `Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz`

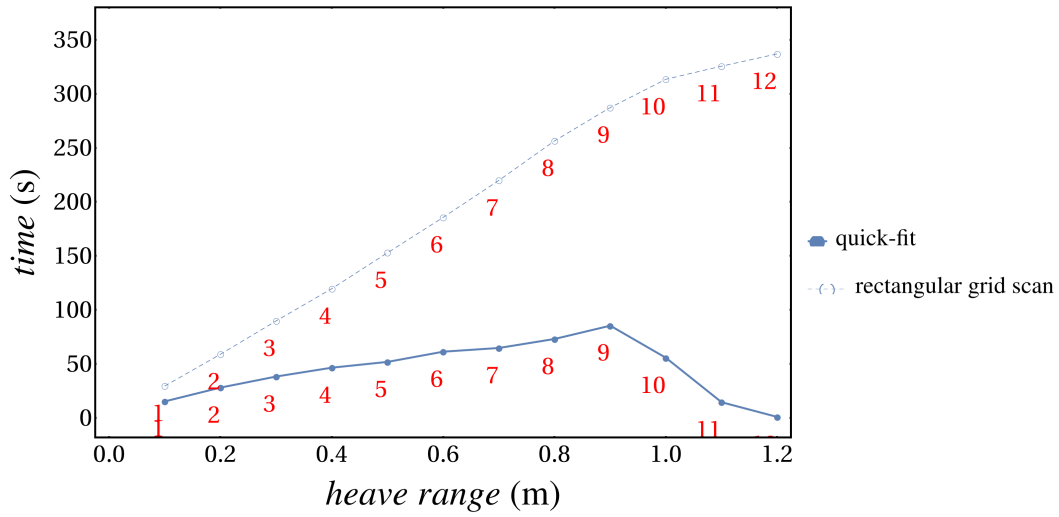


Figure 4.7: Running times for `quick-fit` and `rectangular-grid-scan` for 3-RRS manipulator w.r.t. heave range with labels representing rows of Table 4.4

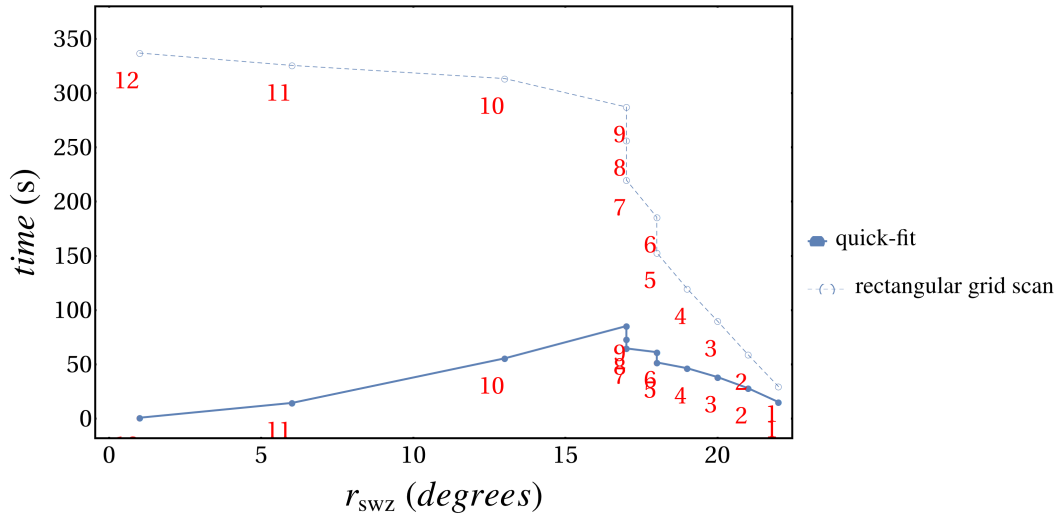


Figure 4.8: Running times for `quick-fit` for 3-RRS manipulator w.r.t. the output (r_{swz}) with labels representing rows of Table 4.4

It can be seen from Fig. 4.9 that in the case of `quick-fit`, the running time is sensitive to input (heave range) as well as the output (radius of the SWZ cylinder) as expected, owing to the approach of growing from inside followed in the algorithm, while the running time of `rectangular-grid-scan` increases with the heave range and is insensitive to the SWZ radius.

The gain in the speed for `quick-fit` comes from the fact that it minimizes the number of points at which the singularity functions are evaluated, as shown in Fig. 4.10, in the workspace when compared to the number of evaluations done by `rectangular-grid-scan`, shown in Fig. 4.11.

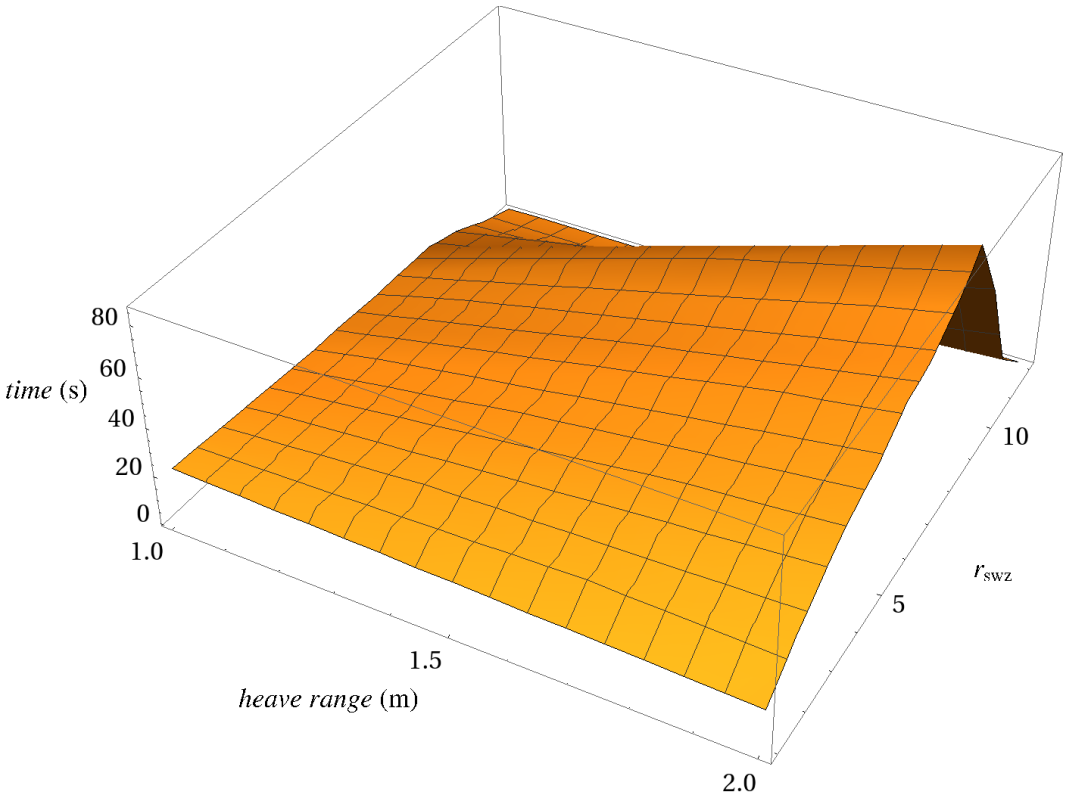


Figure 4.9: Running time of `quick-fit` as a function of input and output

The following points can be noted from the call graph analysis[3] from Figs. 4.10, 4.11:

- The most expensive of the singularity functions (S_3), which takes around 98% of the total computation time of all the singularity functions put together, is evaluated 1140670 times in `quick-fit` compared to 1180251 times in `rectangular-grid-scan` signifying a reduction of 3.3%.
- Every singularity function in `rectangular-grid-scan` makes individual calls to the inverse kinematics functions which makes the ratio of number of S_i computations to that of inverse kinematics computation as 1:1 while the same ratio in `quick-fit` is 3:1.

[3]GNU profiler, `gprof` has been used to generate the profile information.

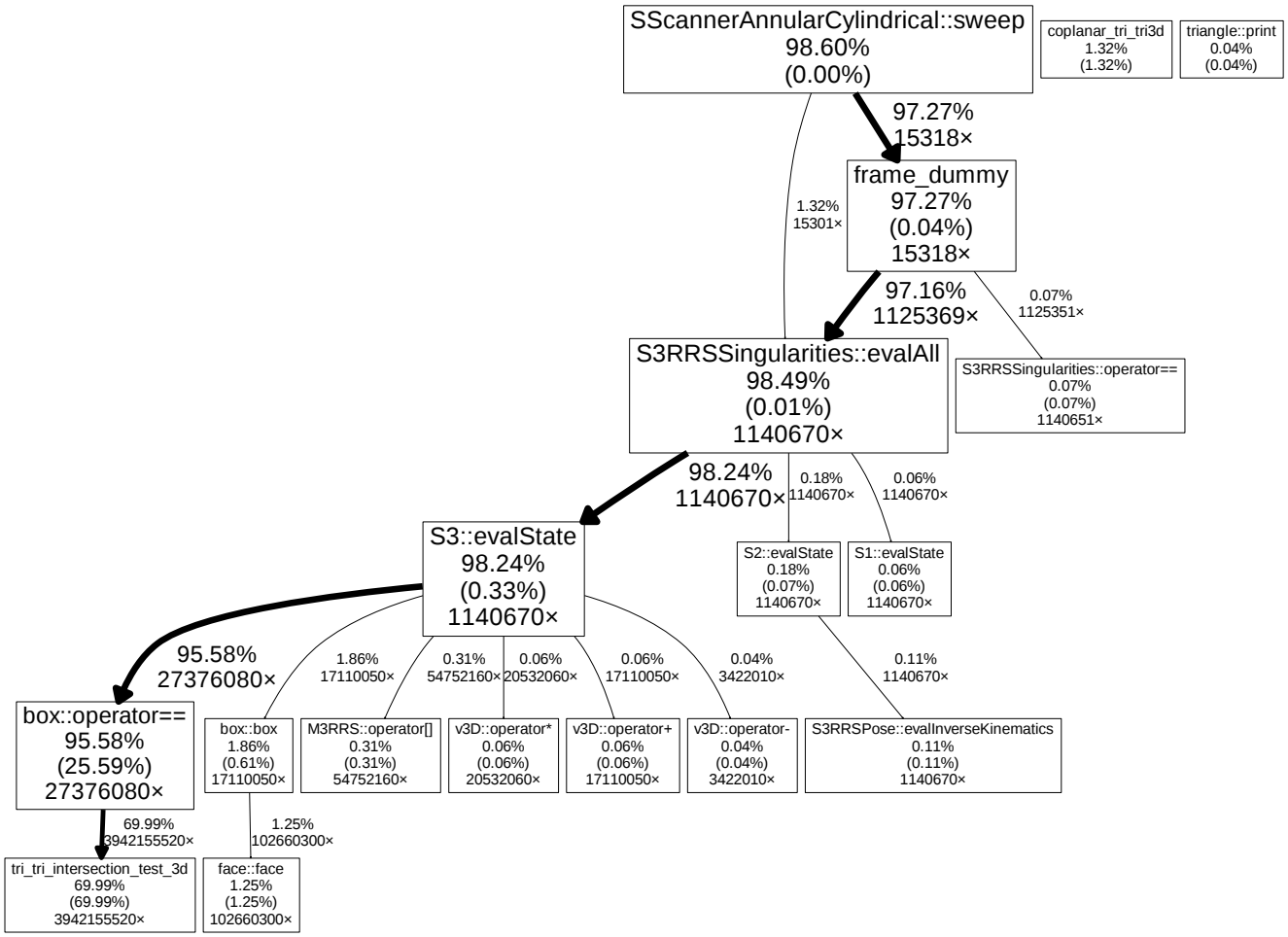


Figure 4.10: Call graph for `quick-fit` used on 3-RRS manipulator

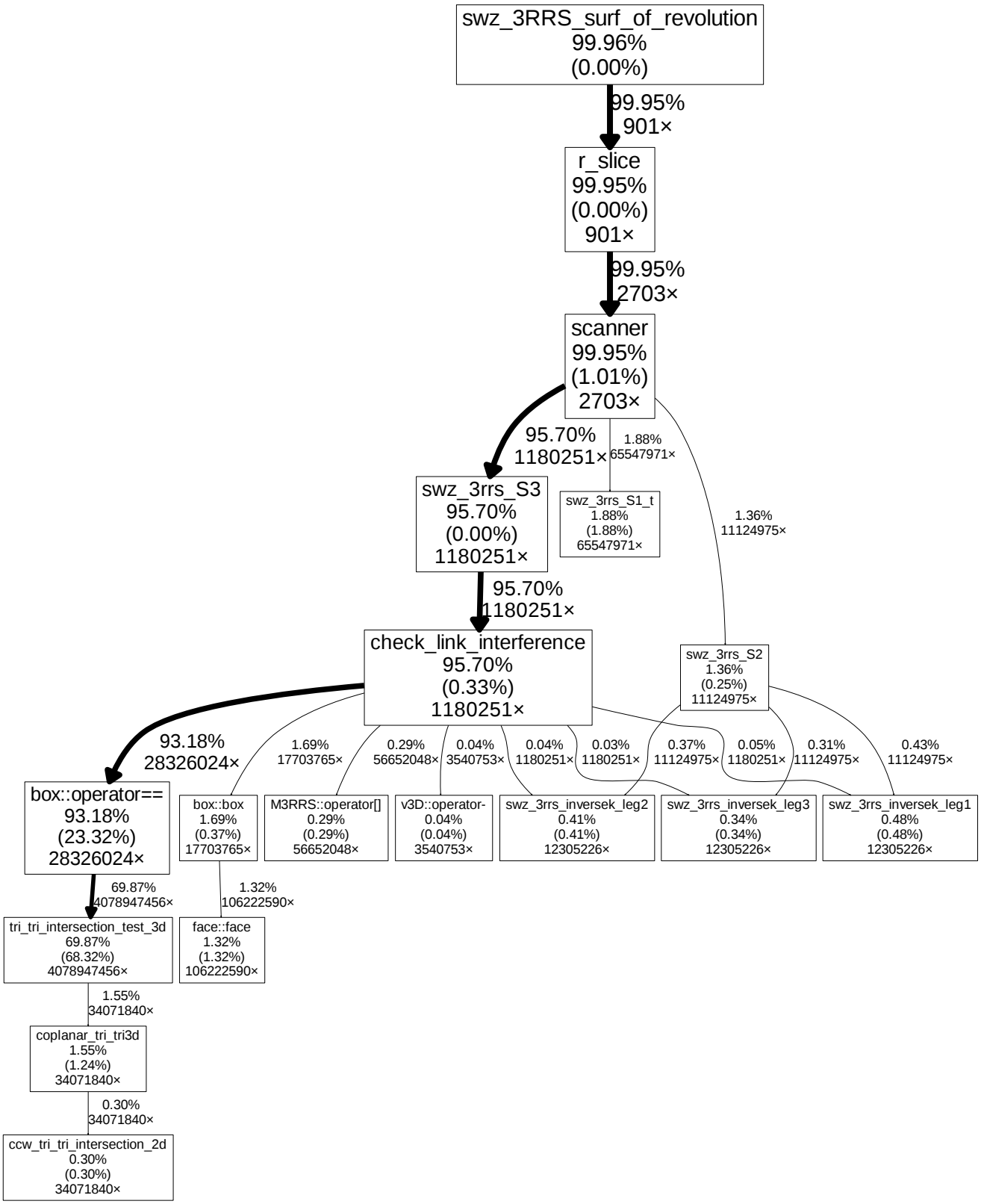


Figure 4.11: Call graph for `rectangular-grid-scan` used on 3-RRS manipulator

Analysis of maximal-fit

The output given by `maximal-fit` is MSoR as defined in section 2, which is same as the output of the `rectangular-grid-scan`, hence these two algorithms can be

compared side by side in all aspects. There are two implementations of `maximal-fit` as discussed in chapter 3 section 3.5. In this section, all the aspects of `maximal-fit -1`, `maximal-fit -2` and `rectangular-grid-scan` are compared.

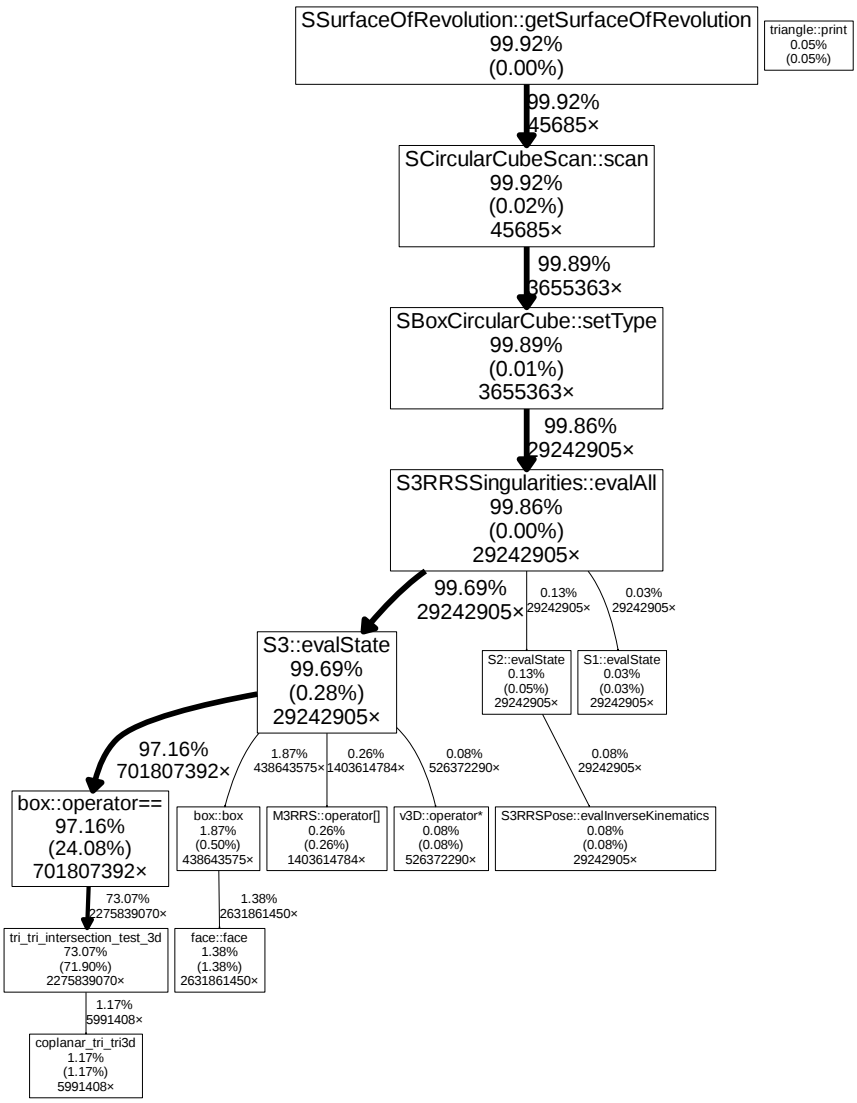


Figure 4.12: Call graph for `maximal-fit -1` used on 3-RRS manipulator

Table 4.5 shows the running times[4] of `maximal-fit -1`, `maximal-fit -2` and `rectangular-grid-scan` for finding the MSoR for the 3-RRS manipulator. The drastic decrease in the running time (approximately $1/8^{th}$) of `maximal-fit -2` when compared to `maximal-fit -1` can be explained by comparing their call graphs shown in Fig. 4.13 and Fig. 4.12 respectively. The ratio of number of times the singularity functions are evaluated in `maximal-fit -1` to that in `maximal-fit -2` is approximately $8 : 1$.

[4]The code was run on a single core of `Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz`

	`maximal-fit -1`	`maximal-fit -2`	`rectangular-grid-scan`
Average running times (s)	2139.21	275.18	340.11

Table 4.5: Average running times for finding the maximal surface of revolution for 3-RRS manipulator

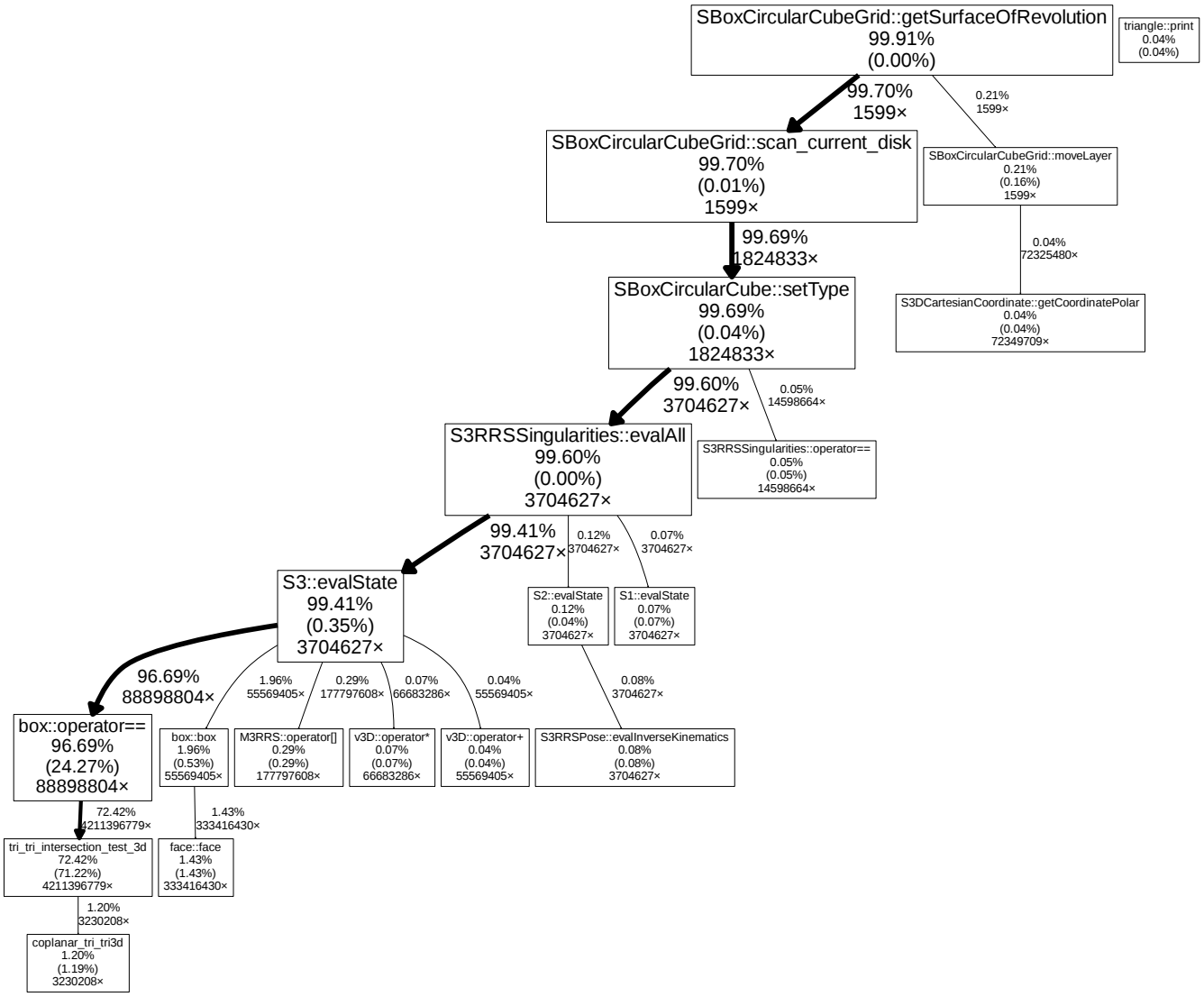


Figure 4.13: Call graph for `maximal-fit -2` used on 3-RRS manipulator

Hence by utilising the concept of *kinematic node* completely, the running time for finding the MSoR is reduced by seven-eighth in the case of the 3-RRS manipulator.

Also, as seen from the call graph for `rectangular-grid-scan` in Fig. 4.14, the `maximal-fit -2` eliminates the redundant calls to inverse kinematic functions,

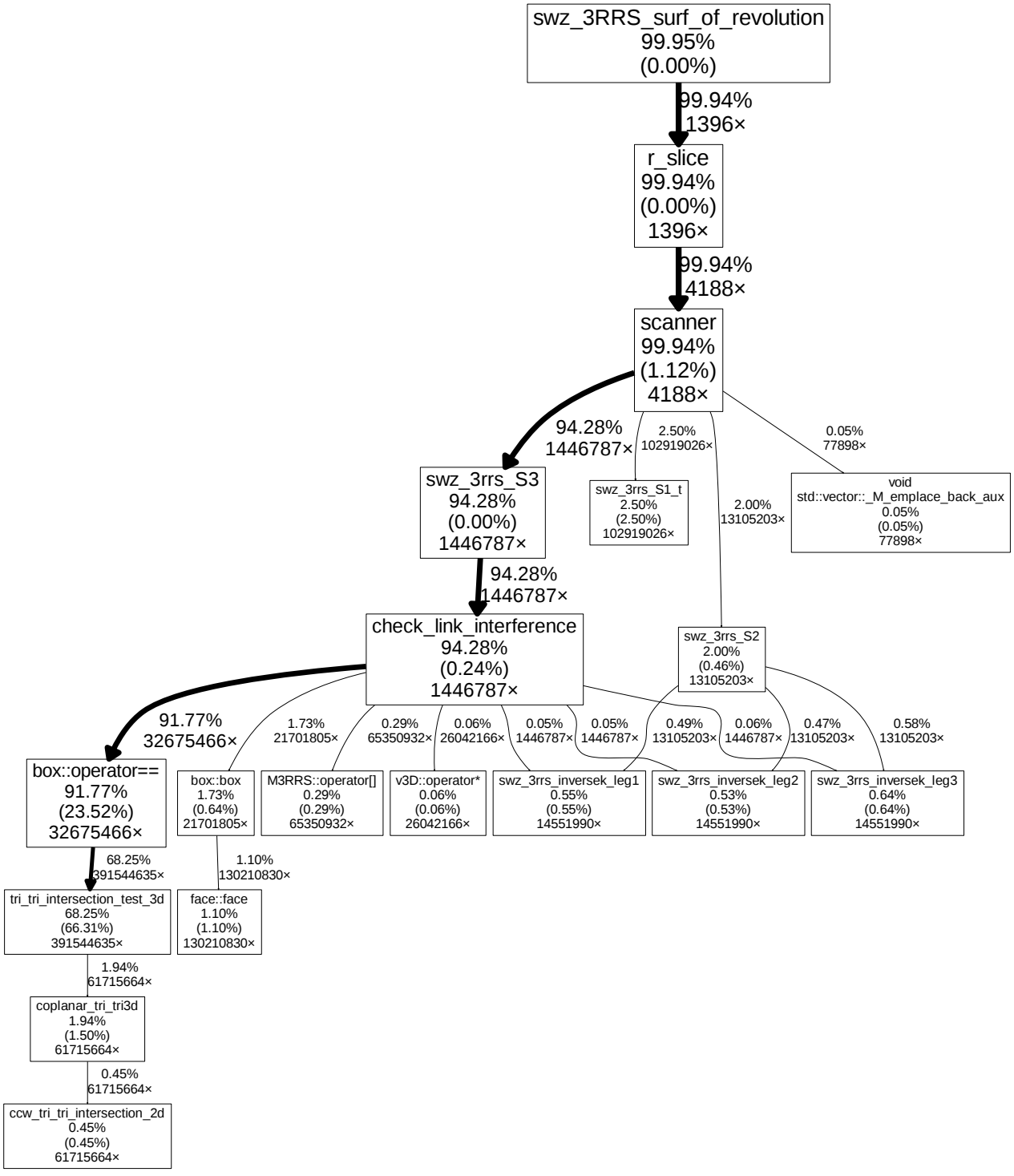


Figure 4.14: Call graph for `rectangular-grid-scan` used on 3-RRS manipulator

which reduces the total time spent in the inverse kinematic function in `maximal-fit-2` to one-tenth of that in `rectangular-grid-scan`.

4.4 Conclusion

This chapter substantiated empirically the claims made in Chapter 3 as features of the `quick-fit` and `maximal-fit`, by applying these algorithms to a five-bar and a 3-RRS manipulator. The `quick-fit` algorithm performs 10 times faster than `rectangular-grid-scan` in the case of a five-bar. It was also observed that the running time of `quick-fit` is sensitive to both the input, i.e., the Z-range, as well as the output, i.e., the SWZ radius.

The `maximal-fit` algorithm takes 20% lesser running time than `rectangular-grid-scan`, to compute the MSoR of a 3-RRS manipulator. It also overcomes the limitations of continuity, mentioned in Section 3.2.1.

CHAPTER 5

Summary

5.1 Conclusion

The main contribution of this work is the conception, development, and implementation of new and adaptable algorithms for finding the SWZ of parallel manipulators. The limitations of the existing algorithms were identified clearly and an attempt was made to overcome these in the new algorithms. The major limitation of the existing algorithm is its discontinuous approach of scanning in one of the three directions, which can lead to erroneous results. Also, the scope for improving the running time was identified.

The concept of *kinematic node* was formulated to cut down on redundant computations to improve the running time. The major challenge was to implement this concept such that it can be used for any manipulator without the need of reimplementing it. This was achieved by extensive use of *class templates* and *class composition* in `C++`. To further minimise the computation, the approach of growing from inside with a predefined convex shape was adopted instead of scanning the entire space. This formulation also addressed the issue of continuity present present in `rectangular-grid-scan`.

The algorithms presented here are implemented in `C++` and can be used for any two or three degree-of-freedom manipulator by simply adding a header file with the singularity functions.

5.2 Scope for extension

As noted in Chapter 4, Section 4.3.3, more than 90% of the computation time spent on the singularity functions goes into computing the S_3 function. This can be reduced by using efficient collision detection libraries, like BULLET PHYSICS (Coumans *et al.* (2013)).

Owing to the geometric nature of the algorithms, the implementation can be adopted to make use of GPUs using `CUDA/C++`, to achieve massive improvements in the running time.

REFERENCES

1. **Bohigas, O.**, **M. E. Henderson**, **L. Ros**, **M. Manubens**, and **J. M. Porta** (2013). Planning singularity-free paths on closed-chain manipulators. *IEEE Transactions on Robotics*, **29**(4), 888–898. ISSN 1552-3098.

2. **Coumans, E.** *et al.* (2013). Bullet physics library. *Open source: bulletphysics. org*.

3. **Dash, A. K.**, **I.-M. Chen**, **S. H. Yeo**, and **G. Yang** (2005). Workspace generation and planning singularity-free path for parallel manipulators. *Mechanism and Machine Theory*, **40**(7), 776 – 805. ISSN 0094-114X.

4. **Ghosal, A.**, *Robotics: Fundamental Concepts and Analysis*. Oxford University Press, New Delhi, 2006.

5. **Gosselin, C.** and **J. Angeles** (1990). Singularity analysis of closed-loop kinematic chains. *IEEE Transactions on Robotics and Automation*, **6**, 281–290.

6. **Jiang, Q.** and **C. M. Gosselin** (2009). Determination of the maximal singularity-free orientation workspace for the Gough-Stewart platform. *Mechanism and Machine Theory*, **44**(6), 1281–1293. ISSN 0094-114X.

7. **Karnam, M.**, **R. A. Srivatsan**, and **S. Bandyopadhyay** (2016). Computation of the safe working zone of a parallel manipulator. Under preparation.

8. **Kilaru, J.**, **M. K. Karnam**, **S. Agarwal**, and **S. Bandyopadhyay**, Optimal design of parallel manipulators based on their dynamic performance. *In The 14th IFToMM World Congress*. 2015.

9. **Leguay-Durand, S.** and **C. Reboulet** (1997). Optimal design of a redundant spherical parallel manipulator. *Robotica*, **15**, 399–405. ISSN 1469-8668.

10. **Li, H.**, **C. M. Gosselin**, and **M. J. Richard** (2006). Determination of maximal singularity-free zones in the workspace of planar three-degree-of-freedom parallel mechanisms. *Mechanism and Machine Theory*, **41**(10), 1157 – 1167. ISSN 0094-114X.

11. **Macho, E.**, **O. Altuzarra**, **C. Pinto**, and **A. Hernandez** (2008). Workspaces associated to assembly modes of the 5R planar parallel manipulator. *Robotica*, **26**, 395–403. ISSN 1469-8668.

12. **Nasa, C.** and **S. Bandyopadhyay**, Trajectory-tracking control of a planar 3-RRR parallel manipulator with singularity avoidance. *In 13th World Congress in Mechanism and Machine Science*. 2011.

13. **Patel, D.**, **R. Kalla**, **H. Tetik**, and **S. Bandyopadhyay** (2016). Computing the safe working zone of a 3-RRS parallel manipulator. Accepted for 6 th European Conference on Mechanism Science, 2016 Nantes, France.

14. **Srivatsan, R. A.** and **S. Bandyopadhyay**, *Computational Kinematics: Proceedings of the 6th International Workshop on Computational Kinematics (CK2013)*, chapter Determination of the Safe Working Zone of a Parallel Manipulator. Springer Netherlands, Dordrecht, 2014. ISBN 978-94-007-7214-4, 201–208.