INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

# Rational Trigonometry Applied to Robotics

Robot Kinematic Modeling using Rational Trigonometry

**João Pequito Almeida**

Dissertação para obtenção do grau de Mestre em

## Engenharia Electrotécnica e de Computadores

**Júri**

Presidente: Por atribuir

Orientador: Prof. Doutor João Fernando Cardoso Silva Sequeira

Vogais: Por atribuir

Por atribuir

**9/2007**

# Agradecimentos

Em primeiro lugar agredeço ao meu Orientador, o Prof. João Silva Sequeira, que desde cedo apoiou este trabalho, sempre com a máxima paciência e espírito crítico.

Em segundo lugar, a todos os meus familiares e amigos, sem excepção, que sempre toleraram os lapsos de atenção por excesso de trabalho e que souberam sempre dar-me um mundo real para onde voltar. A todos, Ana "Iana", Andrea C., André "Cupido", Bruno C., Cláudia S., Henrique L., Inês S., João R., Laura "Suka", Luís "Bak_teria", Luís C., Luís S., Lurdes, Maria F., Paula "Tisha", Pedro B., Pedro V., Quim, Raquel "<3", Rita, Sérgio G. e a todos os que não ficaram por qualquer razão, um muito obrigado.

Por último, agradeço às comunidades virtuais de notícias científicas através das quais conheci a Trigonometria Racional (Slashdot.org, Physorg.com), sem as quais o acaso nunca me levaria a enveredar por este tema, e às longas horas de espera nos transportes públicos sem as quais o tédio nunca produziria as primeiras investidas neste tema.

# Abstract

Rational Trigonometry, as defined by N. J. Wildberger, provides a description of the separation of points (*Quadrance*) and lines (*Spread*) that is independent of reference frames. This thesis explores a few possibilities of Rational Trigonometry when applied to Robotics, namely to kinematic models.

Two main studies are done in this work. The first describes a framework to apply Rational Trigonometry concepts to point description in a standard workspace reference frame. It also provides an informal coordinate system for representing points.

The second is a new model for Robot Kinematics that can be applied to any robotic structure. Several support concepts are introduced and examples are given for Forward Kinematics of some common robotic structures. Inverse Kinematics is discussed briefly and focusing on the major problems at hand.

## Keywords

Robot Kinematics, Rational Trigonometry

# Resumo

A Trigonometria Racional, tal como é definida por N. J. Wildberger, dá-nos uma descrição da separação de pontos (*Quadrância*) e rectas (*Abertura*) que é independente do referencial. Esta tese explora algumas possibilidades da Trigonometria Racional quando aplicada à Robótica, mais concretamente a modelos cinemáticos.

São feitos dois estudos neste trabalho. O primeiro descreve um conjunto de conceitos para aplicar conceitos da Trigonometria Racional à descrição de pontos em referenciais comuns de espaços de trabalho. Também é fornecido um sistema de coordenadas informal para representar pontos.

O segundo é um novo modelo para Cinemática de Robots que pode ser aplicado a qualquer estrutura robótica. São introduzidos vários conceitos de suporte e são fornecidos exemplos para a Cinemática Directa de algumas estruturas robóticas comuns. É feita uma discussão breve da Cinemática Inversa, focando os principais problemas envolvidos.

## Palavras chave

Cinemática de Robots, Trigonometria Racional

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

- **BM**, Base Matrix
- **BMs**, Base Matrices
- **FFM**, Fixed Frames Model
- **JE**, Joint Equation
- **RFM**, Reference Frame Matrix
- **RFMs**, Reference Frame Matrices
- **RK**, Rational Kinematics
- **RT**, Rational Trigonometry
- **SE**, Structure Equation
- **SAE**, Sensor-Actuator Equation

# List of Operators and Notation

## Operators

Let $A$ and $B$ be generic matrices:

- $A \times B$, ordinary matrix product between $A$ and $B$;
- $A \otimes B$, *Kronecker* product (also called *Tensor Product*) between $A$ and $B$;
- $A \bullet B$, *Hadamard* product between $A$ and $B$ (the item-wise multiplication of matrix elements);
- $A \oplus B$, direct sum of matrices $A$ and $B$;
- $\left[\begin{array}{c|c} A & B \end{array}\right]$, horizontal concatenation of matrices $A$ and $B$;
- $\left[\frac{A}{B}\right]$, vertical concatenation of matrices $A$ and $B$;
- $S(\alpha)$, conversion of an angle $\alpha$ to the spread $S_\alpha$;
- $S^{-1}(s_\alpha)$, conversion of a spread $s_\alpha$ to an angle $\alpha$;
- $diag(A_1, A_2, \ldots, A_n)$, generates a diagonal matrix with the $A_i$ elements on the main diagonal and zeros on all other positions (equivalent to $\bigoplus_{i=1}^{n} A_i$);
- $\delta_{ij}$ denotes the Kronecker delta ($\delta_{ij} = 1 \; if \; i = j, \; 0 \; otherwise$);
- $\sqrt{(A)}$, if $A$ is a matrix then $\sqrt{(A)}$ is a matrix with the element-wise square root of the elements of $A$ ($(\sqrt{(A)})_{ij} = \sqrt{A_{ij}}$).

## Notation

- $x$, a coordinate value (lowercase roman character);
- $\alpha$, an angle (lowercase greek character);
- $s$, $s_\alpha$, a spread and the spread of an angle $\alpha$ respectively;
- $c$, $c_\alpha$, a cross and the cross of an angle $\alpha$ respectively;
- $t$, $t_\alpha$, a twist and the twist of an angle $\alpha$ respectively;
- $Q$, $Q_x$, a quadrance and quadrance value for a coordinate $x$ respectively (uppercase roman character);

- $R$, the quadrance radius of polar coordinates;
- $A^T$, if $A$ is a matrix, $A^T$ is its transposed matrix;
- $I_n$, $n \times n$ identity matrix;
- $\mathbb{1}_{n,m}$, $n \times m$ matrix of ones;
- $\mathbb{O}_{n,m}$, $n \times m$ matrix of zeros;
- $\mathbb{L}_n$, $n \times n$ lower diagonal matrix of ones (e.g. $\mathbb{L}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$);

# Chapter 1

# Introduction

In this thesis a generic kinematic model is presented, based on a relatively new trigonometry [Wildberger]. This new trigonometry is called *Rational Trigonometry* (RT) and it is still wrapped in some controversy since it is a relatively new work. Refraining from any address of this matter, validity of its main concepts is clear and provable using "Classical" Trigonometry[1] and therefore will be accepted as a valid theoretical basis. This work applies those main concepts to generic kinematic modeling of robotic structures, namely, sets of interconnected joints with motors that have angular and radial motion.

## 1.1 Motivation

RT simplifies the analysis of triangles, fact that encourages its application to common geometric problems. The main motivation behind applying such trigonometry to robots is that it favors an algebraic approach that allows for a different view of common problems faced when dealing with robot kinematics, more specifically, the heavy use of circular functions. These functions are not only difficult to work with for complex robotic structures but also have some problems when their inverse functions are used, usually needing some sort of per-case adaptation, mostly to select the appropriate signs. RT avoids that as it has a measure of separation of lines that is the same anywhere you make it, and its properties, with appropriate treatment, allow for simpler possibilities for inverse calculations, polar optimization and computational implementations.

## 1.2 Rational Trigonometry

Before any theoretical development, here is a short description of the main concepts of RT.

RT is a new approach to old concepts, namely the separation of points (*quadrance*) and lines (*spread*). Though fully developed independently of "Classical" Trigonometry, it bears a lot of similarities with the latter. In fact, equations 1.1 and 1.2 can be very useful for a first approach.

$$quadrance = distance^2 \tag{1.1}$$

$$spread = sine(line\ separation\ angle)^2 \tag{1.2}$$

One would argue that even though we can use these equations for a first approach, one of the main aspects

[1] Classical in the sense of being prior to the rational formulation by N. J. Wildberger

of RT is that it was developed independently and therefore has its own concepts that should be used instead of their "Classical" counterparts. The following sections include a small description of the main equations and laws, adapted from a small article [Wildberger 2] that is a good starting point for deeper study of the subject.

### 1.2.1 Main concepts of RT

**Quadrance** A point $A$ is an ordered pair $[x, y]$ of numbers. The quadrance between two points $A_1 \equiv [x_1, y_1]$ and $A_2 \equiv [x_2, y_2]$ has the value given by the equation 1.3.

$$Q(A_1, A_2) \equiv (x_2 - x_1)^2 + (y_2 - y_1)^2 \tag{1.3}$$

**Spread** The spread between two lines $l_1$ and $l_2$ with respective equations $a_1x + b_1y + c_1 = 0$ and $a_2x + b_2y + c_2 = 0$ has the value given by the equation 1.4.

$$s(l_1, l_2) \equiv \frac{(a_1b_2 - a_2b_1)^2}{(a_1^2 + b_1^2)(a_2^2 + b_2^2)} \tag{1.4}$$

**Cross and Twist** Two useful definitions are those of the **cross** and the **twist**, the first measures perpendicularity and the second the quotient between the spread and the cross. Their formulas are those of equations 1.5 and 1.6, respectively.

$$c(l_1, l_2) \equiv 1 - s(l_1, l_2) \tag{1.5}$$

$$t(l_1, l_2) \equiv \frac{s(l_1, l_2)}{c(l_1, l_2)} \tag{1.6}$$

### 1.2.2 Main laws of RT

Given three distinct points $A_1$, $A_2$ and $A_3$, define the quadrances $Q_1 \equiv Q(A_2, A_3)$, $Q_2 \equiv Q(A_1, A_3)$ and $Q_3 \equiv Q(A_1, A_2)$ , the spreads $s_1 \equiv s(A_1A_2, A_1A_3)$, $s_2 \equiv s(A_2A_1, A_2A_3)$ and $s_3 \equiv s(A_3A_1, A_3A_2)$ and the crosses $c_1 \equiv 1 - s_1$, $c_2 \equiv 1 - s_2$ and $c_3 \equiv 1 - s_3$.

**Pythagoras' theorem** The lines $A_1A_3$ and $A_2A_3$ are perpendicular when

$$Q_1 + Q_2 = Q_3. \tag{1.7}$$

**Spread law** For any triangle $\overline{A_1A_2A_3}$

$$\frac{s_1}{Q_1} = \frac{s_2}{Q_2} = \frac{s_3}{Q_3}. \tag{1.8}$$

**Cross law** For any triangle $\overline{A_1A_2A_3}$

$$(Q_1 + Q_2 - Q_3)^2 = 4Q_1Q_2c_3. \tag{1.9}$$

**Triple quad formula** The points $Q_1$, $Q_2$ and $Q_3$ are collinear when

$$(Q_1 + Q_2 + Q_3)^2 = 2(Q_1^2 + Q_2^2 + Q_3^2). \tag{1.10}$$

**Triple spread formula** For any triangle $\overline{A_1A_2A_3}$

$$(s_1 + s_2 + s_3)^2 = 2(s_1^2 + s_2^2 + s_3^2) + 4s_1s_2s_3. \tag{1.11}$$

## 1.3 Representing Rotation

### 1.3.1 Rotation in RT

RT may seem an unnatural way to approach rotation since spread and quadrance are both non-linear. Also, it lacks the clarity of angular position and its derivatives. So why use it in the first place?

A possible argument is that spread is a unusual modulus of rotation, making it useful in cases where symmetries are frequent. Now what does this mean?

Consider a point in plain circular motion with fixed radius around the center of a reference frame. Now plot a line uniting both the point and the origin of the reference frame and plot a reference line (it can be one of the axis, for example). Traditionally, one would measure the progression of the angle between these two as the point moves. But if we use spread, we'll quickly realize that it has repeated values throughout an orbit. Immediately a question emerges: if there are repeated values of spread in a full rotation, given a spread, where are we in that trajectory? Clearly, knowing the spread isn't enough to know where a point is. But do we always need to know exactly where it is?

All other possible locations are just reflections of the current one, meaning that for the same spread, we can obtain not only one point but a *set of symmetrical points* that share spread with the first one. This means that to draw a circle we only need to render the first quarter and the remaining three can be obtained by reflection (i.e., swapping the coordinate signs). This is only possible if the trajectory is fully symmetrical around the center of the reference frame, but if so, we have an $n$-fold decrease in the number of calculations needed for full coverage of a trajectory with $n$ symmetries.

Mathematically, if we need the fully signed coordinates of a point, extra values are needed to add location information to the spread (i.e., which symmetry is it). For example, consider the following quadrance circle in a Cartesian reference frame:

$$(Q_x, Q_y) = \begin{cases} (sR, (1-s)R) \; if \; x > 0 \; and \; y > 0 \\ (-sR, (1-s)R) \; if \; x < 0 \; and \; y > 0 \\ (-sR, -(1-s)R) \; if \; x < 0 \; and \; y < 0 \\ (sR, -(1-s)R) \; if \; x > 0 \; and \; y < 0 \end{cases} \tag{1.12}$$

In this example, the location information is given by the signs of $x$ and $y$ in the coordinate system.

So one can say that using spreads a full rotation can be represented using a spread and radius value and then considering all reflections, hence calling it a strange modulus for rotation.

Another issue emerges when one attempts to rotate an existing vector. The most practical solution is to convert its coordinates to their RT polar equivalent and alter the spread. Again, if motion makes the point move to another slice of the reference frame, the appropriate reflection must be selected.

All this complexity makes a good case for angular values, that are both simple and elegant in both of the referred cases. So why use these coordinates on circular problems? Hopefully by the end of this work it will be clear, for now, let's examine other ways of portraying rotation of points.

### 1.3.2 Other representations

There are a lot of representations for rotation, each with its set of advantages and disadvantages. Some of the most common models used to to represent rotation are briefly presented below.

#### Euler Angles and Nautical Angles

A common way of representing both rotation and attitude is to use angles that independently rotate the object about each axis of a reference frame. The most common case is when a three dimensional space is used and therefore, three angles are used. Various conventions exist and the most widely used are both Euler Angles ($\alpha$, $\beta$ and $\gamma$) and Nautical Angles or Tait-Brian Angles ($roll$, $pitch$ and $yaw$). In both cases, a generic three dimensional rotation is represented, considering an object in its reference frame and the sequential rotations about different axis. Though some differences exist, they are mostly a matter of deciding about which axis does the object rotate and in what order. As for the mathematical representation of these rotations, a $3 \times 3$ rotation matrix is used, consisting mostly of circular functions of the referred angles and identity-like elements for the coordinates in the fixed axis.

An important issue with this representation of rotation is that of the *gimbal lock*, where basically when the object is rotated to face directly up or down (in its reference frame), one of the rotations is cancelled and a degree of freedom is lost. Though solutions exist to this problem, it is a singularity that must be kept in mind whenever such a representation is used.

#### Quaternions

Another common way of representing rotation is by using quaternions. Quaternions are an extension of complex numbers and consist of two distinct blocks, one that squares to a positive number (the *real* part), and one that squares to a negative number (the *imaginary* part), the first being comprised of one number and the latter being comprised of three. Therefore, it is a combination four real numbers that are its coordinates using the basis $1$, $i$, $j$ and $k$, respectively (much like $1$ and $i$ in planar complex numbers).

To operate with Quaternions the appropriate definition of products and sums is needed, and some complexity to the calculation is added. Despite that fact, their structure provides a very straightforward way of rotating a generic point about a given axis. To do that, an angle $\alpha$ and a vector $\bar{v}$ must be provided, the latter being a normalized quaternion vector with null real part that represents the said axis. The rotated point is then obtainable by pre-multiplying the original point by the quaternion $z = \cos(\frac{\alpha}{2}) + \sin(\frac{\alpha}{2})\bar{v}$ and post-multiplying by its multiplicative inverse (the provided formula is for a counterclockwise rotation), for example, to rotate the vector $\bar{u}$ about the axis $\bar{v}$ by the angle $\alpha$, the expression will be $z\bar{u}z^{-1}$. The referred calculation is perhaps the most elegant, but matrices can also be used to cause the said rotation of quaternions by matrix multiplication.

Quaternions are more numerically stable when compared to rotation matrices and can be easily converted to and from other coordinates easily. They also avoid the referred *gimbal lock* problem and have no singularities, ensuring their use in many applications in mechanics and computer graphics.

Despite being more compact than rotation matrices (four numbers instead of nine), they require extra

calculations due to the particular nature of their algebra and are non-commutative (fact that is coherent to the non-commutative nature of sequential rotations). Also, in optimization problems involving rotation parameters, the quaternions involved must remain normalized so an extra restriction must be added to the calculations and, if iterative methods are used, they have to be normalized frequently throughout iterations.

Despite the drawbacks and added complexity, Quaternions have a wide acceptance and as the computational capacities available tend to grow, so the added calculations bring little extra weight and make Quaternions more appealing.

As usual, the appropriate choice of a rotation model has the potential to greatly simplify the user's calculations.

## 1.4 Kinematics modelling

Kinematics is the study of the motion of *points* in a *coordinate system* with no regard to *masses* or *forces*. This abstraction allows a mathematical description of the motion of real world objects.

### 1.4.1 Kinematics and Robots

Robots play a increasingly active role in our lives, from manufacturing to entertainment, mindless workers are not only useful but also cheap and efficient. A deep knowledge of Robot Kinematics is therefore needed for any of these applications. But before any theoretical development, the appropriate Robot definition in a mathematical framework must be given.

### 1.4.2 Defining Robots

In this model robots are *a set of points* attached to each other according to a *structure*. This immediately implies that they are a *whole entity* comprised of smaller *parts*. Each part has, most commonly, one or more actuators[2] and a physical body described by one or more *points*. Since most of these points are part of a rigid block, describing the motion of a single point is enough to describe the motion of the remainder points. These blocks will be referred to as *joints*. It is assumed that somehow these actuators present in each part change the location in space of the points connected to it.

So, in this approach, a Robot is *a structured set of points* (a structure of joints) whose motion is related to both *motion of the actuators* and *motion of neighbouring points*.

## 1.5 Structure

This thesis is organized in two main parts. The first introduces all concepts used in a detailed manner, the second approaches forward kinematics in a global perspective. The first part (chapters 2, 3 and 4) will introduce both the coordinates used and the concepts developed to approach robot kinematics. The second part (chapters 5, 6, 7, 8) will provide a global perspective of how to approach forward kinematics of any robotic structure, including some examples.

The chapters are organized as follows:

[2] Actuators are a device for moving or controling a mechanism, in this case, a part of a Robot.

- Chapter 2 describes the initial approach and study of the algebraic structure of using RT concepts for point representation;
- Chapter 3 applies these concepts and introduces a framework that allows for the use of RT concepts in a regular workspace, providing an informal coordinate system;
- Chapter 4 explores the possibilities of this coordinate system, namely in the potential for simultaneously simulate various reflections of a point at once;
- Chapter 5 introduces the main concepts of the kinematics model developed in this work, a main difference of it from other models is that it uses fixed reference frames instead of moving reference frames;
- Chapter 6 introduces concepts that allow this model to be applied to commonly used sensors and actuators;
- Chapter 7 provides a set of equations to deal with restrictions of robotic structures;
- Chapter 8 summarizes the model used and expressions developed and applies it to several robotic structures as examples;
- Chapter 9 explores the consequences in general of the model developed, its consequences for inverse kinematics and future work.

## 1.6 Contributions

This work provides a framework that gives a global view of the problems involved in robot kinematics, mainly of complex robotic structures. By applying this framework to a generic robot, both forward and inverse kinematics have a straightforward way of being implemented, since most of the elements can be obtained by simple inspection of a robot. It also has a performance gain, mainly in aspects of computational implementation as it explores natural redundancies in robotic systems and avoids calls to circular functions. On the other hand, the formulation may lack clarity at times due to the particularities of the objects used, mainly the reflection addressing problems when dealing with motion.

# Chapter 2

# Approach

Rational Trigonometry concepts provide a trade-off between operation simplicity and expression simplicity. Quadrances and Spreads can no longer be added using standard sum, yet the expressions obtained by using them are much simpler. A natural approach to apply RT concepts to a standard workspace is to, first of all, analyze the algebraic structure that RT coordinates exhibit.

## 2.1 Algebraic structure of RT concepts

The structure that was analyzed was one that allowed for an identical reasoning for Quadrances and Spreads as the reasoning for Distances and Angles, respectively. The operators obtained reflect the objectives of this work, as more operators exist and other ways can be used to obtain these results.

While the product of Quadrances and Spreads is identical to the product of distances and angles, the sum is very different, as portrayed by the equations 2.1 and 2.2 that were obtained. These operators are very important to understand the coordinates used in this work.

$$\begin{gathered} \textit{Let } A \textit{ and } B \textit{ be generic quadrances,} \\ A \boxplus B = (\sqrt{A} + \sqrt{B})^2 \\ A \boxminus B = (\sqrt{A} - \sqrt{B})^2 \end{gathered} \tag{2.1}$$

$$\begin{gathered} \textit{Let } s_A \textit{ and } s_B \textit{ be generic spreads,} \\ s_A \oplus s_B = (\sqrt{s_A(1-s_B)} + \sqrt{s_B(1-s_A)})^2 \\ s_A \ominus s_B = (\sqrt{s_A(1-s_B)} - \sqrt{s_B(1-s_A)})^2 \end{gathered} \tag{2.2}$$

Note that these operators have a mirroring behavior when a point approaches the reference frame planes. They also have been simplified as they lack the appropriate signedness. This fact discussed in appendix A and is key to the approach followed in this work; if signedness must be addressed, a framework for it must be conceived so that these operators are valid in every situation.

But why are reflections so important? The main argument is that each reflection is a small slice of space in which an arc and an angle have a unique relation, allowing for a direct conversion between the two. But RT concepts alone leave a problem to be solved: how to uniquely select a reflection. A solution to this problem is proposed with the informal coordinate system presented in the following chapters. By using the proposed coordinates it is possible to convert rectangular to polar coordinates and vice versa with no ambiguity.

# Chapter 3

# Point description using Rational Trigonometry

A point in RT is quite similar to a point in $\mathbb{R}^n$, the main difference being that its coordinates are measured as quadrance. An intuitive way of portraying a point is by defining a reference frame and then obtaining its coordinates on that reference frame. When referring to robots, it is also useful to portray not only the location of a point but also the attitude (what spatial direction is the robot facing). This is also very useful for manipulators, so that they approach a work target the right way.

Throughout this work points will be used instead of vectors, so their equivalent moving vector is the one referred to the origin of the reference frame.

## 3.1 Base matrices for trajectory representation

The algebraic structure obtained demonstrates that there is a common unsigned part that signed reflections share. A way to represent this common part is by setting up a Base Matrix (BM) that holds the variable parameters for the unsigned part of the coordinates. By varying this Base Matrix, all reflections are altered and therefore the choice of the appropriate BM is key to an accurate description of the trajectory. As the most commonly used actuator block is a radial one, usually with fixed radius, the polar RT conversion is ideal for representing it. The conversion between rectangular and polar coordinates is given by the equations 3.1 and 3.2, exemplifying respectively the two dimensional and three dimensional case. The equations are generalized to further dimensions in [Wildberger] but use for further dimensions of position isn't required for this work.

$$\begin{cases} Q_x = s_\theta R \\ Q_y = (1 - s_\theta) R \end{cases} \tag{3.1}$$

$$\begin{cases} Q_x = s_\theta s_\varphi R \\ Q_y = (1 - s_\theta) s_\varphi R \\ Q_z = (1 - s_\varphi) R \end{cases} \tag{3.2}$$

Note that a slightly different notation is used than that of [Wildberger]. This choice was made so that the standard polar angles $\theta$ and $\varphi$ are explicitly depicted.

The actual BMs are given by the equations 3.3 and 3.4. Note that BMs relate *actuators* and *trajectories*, meaning that for different robots BMs may vary substantially.

$$B = \begin{bmatrix} s_\theta R & 0 \\ 0 & (1 - s_\theta)R \end{bmatrix}$$
$$\begin{bmatrix} Q_x & Q_y \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix} \times B \tag{3.3}$$

$$B = \begin{bmatrix} s_\theta s_\varphi R & 0 & 0 \\ 0 & (1 - s_\theta)s_\varphi R & 0 \\ 0 & 0 & (1 - s_\varphi)R \end{bmatrix}$$
$$\begin{bmatrix} Q_x & Q_y & Q_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \times B \tag{3.4}$$

Identical BMs can be obtained for attitude. Consider a common Euler Angles representation and its spread equivalents, the conversion between joint spreads ($s_\theta$ and $s_\varphi$) and Euler spreads ($s_\alpha$, $s_\beta$ and $s_\gamma$) can be obtained by the expressions of equation 3.5. Note that for two dimensions the conversion is direct from the angle ($s_\theta = S(\theta)$).

$$\begin{cases} s_\alpha = s_\theta \\ s_\beta = \frac{1}{1+s_\theta \frac{s_\varphi}{1-s_\varphi}} \\ s_\gamma = \frac{1}{1+(1-s_\theta)\frac{s_\varphi}{1-s_\varphi}} \end{cases} \tag{3.5}$$

Again, the equation 3.6 depicts this as a regular BM. Note that the values used are angles as spreads need no expansion. The $R$ matrix may vary depending on the angle chosen for a spread.

$$\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} = \mathcal{R} \times \begin{bmatrix} S^{-1}(s_\theta) & 0 & 0 \\ 1 & 0 & 0 \\ 0 & S^{-1}(\frac{1}{1+s_\theta \frac{s_\varphi}{1-s_\varphi}}) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & S^{-1}(\frac{1}{1+(1-s_\theta)\frac{s_\varphi}{1-s_\varphi}}) \\ 0 & 0 & 1 \end{bmatrix}$$
$$\mathcal{R} = \begin{bmatrix} 1 & \frac{\pi}{2} & 1 & \frac{\pi}{2} & 1 & \frac{\pi}{2} \end{bmatrix} \tag{3.6}$$

The coordinate base matrix relates *robot actuator data* to *user defined coordinates*. It is possible to combine multiple base matrices with each other if more user defined coordinates are needed. How to relate these with a real world reference frame?

## 3.2 Reference Frame Matrices

Commonly used reference frames relate an unsigned number to its sign by the means of an axis system composed of number lines that have each one or more signs (e.g., the three-dimensional Cartesian reference frame has 3 number lines, each with the signs $+$ and $-$).

All we need to do in order to add signedness to the BMs described above is set up a matrix that has the right signs in the right elements. For example, a standard three-dimensional Cartesian Reference Frame yields the following Reference Frame Matrix (RFM) depicted on equation 3.7. Its elements are either $1$ or $-1$ depending on which octant we are and what signs does a point in that octant have. Note that signed quadrances are used but if we apply the square root function to all elements of the base matrix the results will be the signed distances for each reflection.

$$\underset{\textit{Base Matrix}}{B} = \begin{bmatrix} s_\theta s_\varphi R & 0 & 0 \\ 0 & (1-s_\theta)s_\varphi R & 0 \\ 0 & 0 & (1-s_\varphi)R \end{bmatrix}, \quad \underset{\textit{Reference Frame Matrix}}{\mathcal{R}} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} Q_{x1} & Q_{y1} & Q_{z1} \\ Q_{x2} & Q_{y2} & Q_{z2} \\ Q_{x3} & Q_{y3} & Q_{z3} \\ Q_{x4} & Q_{y4} & Q_{z4} \\ Q_{x5} & Q_{y5} & Q_{z5} \\ Q_{x6} & Q_{y6} & Q_{z6} \\ Q_{x7} & Q_{y7} & Q_{z7} \\ Q_{x8} & Q_{y8} & Q_{z8} \end{bmatrix} = \mathcal{R} \times B \tag{3.7}$$

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \\ x_6 & y_6 & z_6 \\ x_7 & y_7 & z_7 \\ x_8 & y_8 & z_8 \end{bmatrix} = \mathcal{R} \times \sqrt{(B)}$$

Identical reasoning can be done for Euler Angles. It is up to the user to decide what kind of convention should be used to portray all angles equivalent to a spread. That convention will be expressed in his choice for a RFM. The equation 3.8 depicts just that, an expansion of attitude spreads to their angular counterparts considering that a spread $s$ has as its four *reflections*, the angles $\{S^{-1}(s), \pi - S^{-1}(s), \pi + S^{-1}(s), -S^{-1}(s)\}$ (in radians).

$$B = \begin{bmatrix} S^{-1}(s_\theta) & 0 & 0 \\ 1 & 0 & 0 \\ 0 & S^{-1}(\frac{1}{1+s_\theta \frac{s_\varphi}{1-s_\varphi}}) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & S^{-1}(\frac{1}{1+(1-s_\theta)\frac{s_\varphi}{1-s_\varphi}}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathcal{R} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ -1 & \pi & -1 & 0 & 1 & 0 \\ 1 & -\pi & -1 & 0 & -1 & \pi \\ -1 & 0 & 1 & 0 & -1 & \pi \\ 1 & 0 & -1 & \pi & -1 & 0 \\ -1 & \pi & 1 & -\pi & -1 & 0 \\ 1 & -\pi & 1 & -\pi & 1 & -\pi \\ -1 & 0 & -1 & \pi & 1 & -\pi \end{bmatrix} \tag{3.8}$$

$$\begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \\ \alpha_4 & \beta_4 & \gamma_4 \\ \alpha_5 & \beta_5 & \gamma_5 \\ \alpha_6 & \beta_6 & \gamma_6 \\ \alpha_7 & \beta_7 & \gamma_7 \\ \alpha_8 & \beta_8 & \gamma_8 \end{bmatrix} = \mathcal{R} \times B$$

These matrices exhibit a very intuitive structure that depicts the reference frame used for an application. It also allows for conversion between different work conventions used by simply applying a line-swapping matrix or, if the conversion is a complex one, a generic conversion matrix. The key function of these RFMs is to relate the *trajectory* portrayed by the BMs to the *workspace*, where signedness is present. Setting up RFMs is very intuitive and has a direct link to everyday workspaces, keeping a mathematical depiction of the directions used for positive and negative motion.

## 3.3 Matrices to address reflections

In order to address a reflection in a unique way, we simply need to select the right line from the matrices described in the previous sections. For that we can simply use a line selecting matrix that will pick the desired reflection, as described by the 3D example of equation 3.9.

*Let $P$ be the desired point in standard rectangular coordinates,*

$$B = \begin{bmatrix} s_\theta s_\varphi R & 0 & 0 \\ 0 & (1-s_\theta)s_\varphi R & 0 \\ 0 & 0 & (1-s_\varphi)R \end{bmatrix}, \; \mathcal{R} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix} \tag{3.9}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P = L \times \mathcal{R} \times \sqrt{(}B)$$

Note that in this case a single line is selected but we can select multiple reflections at once. In fact, if the $L$ matrix is, for instance, an $I_8$ matrix, for each value of the base matrix the $8$ reflections are generated. This is very useful in rendering of a full sphere for instance, since first we can render $1/8$th of the sphere with the base matrix and then, for each point, expand it to its $8$ reflections, with no extra calculations (only the sign changes).

Changing the $L$ matrix allows for greater flexibility when calculating one or more reflections. It can be seen as an extra discrete dimension, key to the bijective mapping between polar and rectangular coordinates. Note that the $L$ matrix elements aren't limited to unary values and can weight, for instance, the proximity of one point to a slice of the reference frame, though this possibility isn't the one explored in this work.

So putting these three objects together allows for a unique point description in a generic reference frame and this will be referred to as the Joint Equation (JE). A slightly more detailed explanation of this can be found in appendix A.2.

## 3.4 The Joint Equation

A robot joint can be generated entirely from a single point, since it is assumed to have a rigid body. So the equation described above can be used to describe the points of a robot joint, hence calling it the Joint Equation. In fact, only one point is needed since all others move with it. But what point should be considered the key point of a joint? The choice made in this work is to select the point where the actuator of the joint attached to it is connected. This means that, for example, on a series manipulator, if we consider the first joint to be the one attached to the base, the joints will be attached in sequence from the origin to the terminal point and the points considered for simulation will be the extremities of each joint.

The equation obtained for the description of points related to robot joints, referred to hereon as the Joint Equation, has the expression provided in equation 3.10.

*Let P be the desired point, L the Line Selecting Matrix, R the Reference Frame Matrix and B the Base Matrix,*

$$P = L \times \mathcal{R} \times B \tag{3.10}$$

This concludes the main concepts for point description using Rational Trigonometry concepts and in the next chapter the possibilities for analyzing multiple points at once is discussed.

# Chapter 4

# Analyzing sets of points

The point representation described, composed of the three elements $L$, $\mathcal{R}$ and $B$ can provide not only one point but a set of points. But whenever we use sets instead of a single point, an issue emerges: how to combine each point of that set with every other point of every other set. In order to obtain all combinations, an expression that uses a few algebra tricks was devised to obtain fully unique combinations of these reflections that can posteriorly be combined in unique pairs using standard matrix algebra.

Understanding this is essential for the proof that this point representation can cover a full workspace, as will be discussed. Even though they can be called coordinates informally and for simplicity, there is no attempt in this work to prove that they are a fully valid coordinate system.

## 4.1 Combining multiple reflections

There are several ways of representing sets of points. The chosen manner was by using matrices that have in each column the coordinates used. If we are considering a set instead of a single point, the set will be a matrix with the same number of columns than a point but with each point in a row of that matrix. This means that when we use $c$ coordinates and have $n$ points in one set the matrix will be $n \times c$.

Consider now that we have $m$ sets of points, each being $n_i \times c$. In order to obtain all combinations possible between all points one would have to isolate each row of each set and combine it with every row of every other set. To do this, an algebra trick was used. Each of these sets will have its elements repeated by rows $I$ times and then all its elements repeated by rows $O$ times. By varying the value of $I$ and $O$ in each set different matrices are obtained. These matrices were designed so that if all $m$ matrices are concatenated horizontally with each other all lines are unique. In a way, it is a process similar to the cartesian product of sets that yields a set of matrices that can be uniquely combined by rows. These matrices will have a number of rows that depends on how many points each set has. If each set $i$ has $n_i$ points, the final matrices will be $(\prod_{1}^{m} n_i) \times c$.

The expressions devised are presented in equations 4.1, 4.2, 4.3, and 4.4.

To better understand the $I$ and $O$ values and the $i$ indexing, the figure 4.1 depicts the convention used, the $I$ standing for *inner* and $O$ standing for *outer* joints. Also, the function $CC$ stands for *Combination Count*, as the function calculates the total number of combinations possible in a given range of joints. Note that if the joints are coherently numbered, they do not need to be physically sequential, the expression will always provide coherent results. Also note that the value $rank(L_i)$ is the total number of tuples (reflections) to be combined.

*For an ordered joint $i$ in a robot with $n$ joints, let $P_i$ be the actuator point obtained using the JE,*

$$M_j = [\delta_{1j}\delta_{2j}\dots\delta_{rank(L_i),j}] \tag{4.1}$$

$$CC(a,b) = \prod_{i=a}^{b} rank(L_i) \tag{4.2}$$

$$I = CC(1, i-1);\ O = CC(i+1, n) \tag{4.3}$$

$$P_i^* = \left( \sum_{j=1}^{rank(L_i)} \left( M_j \times P_i \otimes \mathbb{1}_{I,1} \otimes M_j^T \right) \right) \otimes \mathbb{1}_{O,1}. \tag{4.4}$$

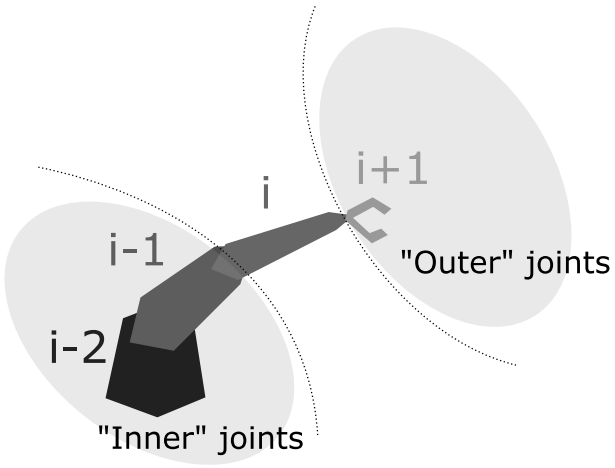


Figure 4.1: Depiction of the notation used for joint indexing

Hereon, these expressions will be referred to as $perm(P_i)$, as it does permutations of tuples to obtain a per-row unique set of tuples. One important feature of this expression is that each $P^*$ obtained can be combined with every other in a standard algebraic manner and each row will represent a unique combination. Also note that it is valid for any set of matrices. Its final formulation to be used hereon is portrayed in equation 4.5.

$$P_i^* = perm(P_i) \tag{4.5}$$

Though this may seem as a complex operation, it is rarely needed. In fact, most uses will convert one angle to one spread and disregard reflections therefore the $P_i$ matrices obtained all have $rank = 1$, so $P_i = perm(P_i) = P_i^*$.

This equation is very useful for a full rendering of multiple reflections combined with each other in a workspace and for a good understanding of the inverse problem, but should be used with notice to the fact that it grows exponentially with $n$ (for 3D for instance, if all reflections are used, the number of rows of the matrices is $8^n$).

So, using the $perm$ formula, a $P_i^*$ matrix is obtained that contains the tuples organized so that regular matrix algebra can be used to combine them.

But if the points have RT coordinates, their special algebra must be used. Since by using the JE all signs are separated from the values, the algebraic combination of RT coordinates, can be done by applying the square root function to all quadrances in the BMs. As for the angular data, since the operator is more complex, the simplest approach is to use the inverse mapping of spreads to linear angular coordinates so that they can also be added using regular addition.

This fact allows for a formulation that can use both RT coordinates and other coordinates in the same matrices, as long as the appropriate algebraic rules are respected. That provides great flexibility for adding extra data to a robot analysis and paves way for the final formulations obtained.

## 4.2 Covering a full workspace

Understanding the combination of reflections is key to understanding the coverage of a workspace. Each reflection, or tuple, is a slice of a reference frame. Each of these slices coincides with the standard notion of quadrants, octants and other ways of referring to sections of reference frames. So if each reflection covers a full slice of a reference frame, in order to cover a full workspace, all slices must be covered in all workspaces.

The proof is provided in appendix A for an example BM and RFM, but a proof must be done for each BM and RFM combination used, as there may be matrices that do not allow a full coverage of a workspace.

# Chapter 5

# The Fixed Frames Model for Robot Kinematics

By using the coordinates described in the previous chapter a difficulty is encountered when approaching Robot Kinematics (RK) in a standard manner. There are multiple transformation matrices for these coordinates, each with limited validity depending on context, so using the standard approach that involves transforming and rotating each reference frame of a robot joint proved to be much more difficult than usual. In order to address this difficulty, a very different approach was used.

Instead of transforming each reference frame, all reference frames are kept fixed and with the same attitude as the reference frame defined as the root. Also, all motion is referred to these fixed reference frames as though each joint was floating independently in space.

The following chapters explore the consequences of this approach. All examples will be given using $L\times\mathcal{R}\times B$ coordinates but all formulation developed can be applied using other coordinates.

This model will hereon be referred to as the Fixed Frames Model (FFM), and is perhaps the biggest consequence of this study.

## 5.1 Representing and analysing robotic structures

A key aspect of RK is how to represent a given robotic structure. Usually, a per-case technique is used where the robotic structure has an influence on how the calculations are done and how they turn out.

By keeping the reference frames fixed and parallel axis-wise, each value for each axis can be added to each other accordingly, allowing for a simple, sum-like approach, to the matter. To exemplify, consider a series manipulator with $n$ joints, starting at a fixed point and with the following joints all attached to the previous. In order to know the coordinates of the actuator point (the last point of the last joint in the workspace reference frame), all we have to do is sum each of the contributions of the joints to each coordinate, as shown in equation 5.1 (the $f$ is for final and the $P_i$ means the point coordinates of joint $i$).

*Let $P_f$ be the actuator coordinates of the manipulator,*

$$P_f = \sum_{i=1}^{n} P_i^* \tag{5.1}$$

Note that this is only possible because reference frames do not move. A similar reasoning can be done for intermediate points, as they are also a sum of the per-dimension contributions of the previous joints. So, for a joint with index $j$, the coordinates of its actuator point are given by the equation 5.2.

$$P_{f_j} = \sum_{i=1}^{j} P_i^* \tag{5.2}$$

By using matrix sum instead of products, there is a slight computational performance gain as no multiplications are needed. It also turns a product of matrices into a sum of matrices that allows a linear system formulation.

If we consider every point of every joint and apply the shown equations to each, a nice formula emerges, best exemplified by equation 5.3.

$$\begin{cases} P_{f_1} = P_1^* \\ P_{f_2} = P_1^* + P_2^* \\ \vdots \\ P_{f_n} = P_1^* + P_2^* + \ldots + P_n^* \end{cases} \tag{5.3}$$

Which says that each of the final points considered is given by a linear combination of the points of each joint. This means that using this approach there is a linear mapping between point coordinates of each joint and points on the workspace reference frame.

## 5.2 Structure Matrices

If we apply matrix notation to the previous problem, we obtain a type of matrices that are intrinsically connected to the structure of the robot. This is most explicitly clear when matrix notation is used for the previous problem, exemplified in equation 5.4. If more than one reflection per joint is used, the matrix must be cloned for each combination of points used. This can be done using the $CC$ function and the Kronecker product.

$$\begin{bmatrix} P_{f_1} \\ P_{f_2} \\ \vdots \\ P_{f_n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 1 \end{bmatrix} \otimes I_{CC(1,n)} \times \begin{bmatrix} P_1^* \\ P_2^* \\ \vdots \\ P_n^* \end{bmatrix} \tag{5.4}$$

Hereon, $\begin{bmatrix} P_1^* \\ P_2^* \\ \vdots \\ P_n^* \end{bmatrix}$ will be referred to as $P^*$.

Now this works for any series manipulator and is especially useful for robots with many sequential joints like snakes or tentacles.

But another common structure is that of a parallel manipulator, where multiple series manipulators are connected to each other, usually by the actuator. So, as a first example, consider a parallel robot that has two series blocks united by a single block. For simplification, it is assumed that the final block has free-moving

attachments to the actuators. As an example, consider the middle point of that final block, noted as $P_f$. Its coordinates are given by the equation 5.5, where $P_a$ and $P_b$ are coordinates of each series block.

$$\begin{cases} P_{a_f} = \sum_{i_a=1}^{n_a} P^*_{a_i} \\ P_{b_f} = \sum_{i_b=1}^{n_b} P^*_{b_i} \\ P_f = \frac{P_{a_f} + P_{b_f}}{2} \end{cases} \tag{5.5}$$

Again, the final point is a linear combination of the joint points, so a matrix can also be obtained, provided in equation 5.6. Note that, since the parallel block is linearly dependant of the other blocks, the matrix obtained isn't square. This is coherent with the fact that only the joints of the series blocks are available for motion.

$$\begin{bmatrix} P_{a1_f} \\ P_{a2_f} \\ \vdots \\ P_{an_f} \\ P_{b1_f} \\ P_{b2_f} \\ \vdots \\ P_{bn_f} \\ P_f \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \otimes I_{CC(1,n)} \times \begin{bmatrix} P^*_{a_1} \\ P^*_{a_2} \\ \vdots \\ P^*_{a_n} \\ P^*_{b_1} \\ P^*_{b_2} \\ \vdots \\ P^*_{b_n} \end{bmatrix} \tag{5.6}$$

Also note that the last row is a simple average of of both of the actuator points. For example, if we were to consider a parallel structure with 3 series blocks instead of two, the last row would have $\frac{1}{3}$ instead of $\frac{1}{2}$ as its value. If we extend this reasoning, for a parallel block with $n$ joints attached to it, the last row would have $\frac{1}{n}$ as its value.

### 5.2.1 Applying constraints

An issue with parallel blocks that does not emerge in series blocks is that there are constraints that limit its possible positions.

The linear formulation obtained also simplifies this analysis and a simple set of equations can be used.

These are discussed in chapter 7.

## 5.3 The Structure Equation

Using this approach we have obtained an equation that relates per-joint points with their coordinates in a structure. This equation will be referred to as the Structure Equation (SE) and is given by the equation 5.7, where $P_f$ has all the per-joint terminal points in the workspace reference frame and $P^*$ has all the joint coordinates in their own reference frames after the $perm$ operation.

$$P_f = \mathcal{S} \times P^* \tag{5.7}$$

An interesting fact about the SE is that it can be obtained by simple inspection of a robot's structure, as series blocks are always represented by a lower diagonal matrix of ones, and parallel blocks are always represented as rows that calculate averages of actuator points. Also, a complex structure of $n$ series manipulators will always have an $\mathcal{S}$ matrix that is the direct sum of the $n$ smaller $\mathcal{S}$ matrices. If the last block of such a structure is a parallel block, the generic equation 5.8 is obtained, where $\mathcal{S}_i$ is the structure matrix of a series block $i$ and $\mathcal{S}_f$ is the final structure matrix.

$$\mathcal{S}_f = \begin{bmatrix} \bigoplus_{i=1}^{n} S_i \\ \frac{1}{n} \quad \cdots \quad \frac{1}{n} \end{bmatrix} \tag{5.8}$$

The SE provides a very straightforward way to approach any robotic structure, even a complex one. The expense is, obviously, the size of the structure matrix, that not only grows with the number of elements of a robot structure but also the number of reflections considered. In fact, if we have a total of $CC(1,n)$ reflections, the final structure matrix will be at least $nCC(1,n) \times nCC(1,n)$. But this is a very rare case and most uses will have an $\mathcal{S}$ matrix that is $n \times n$, since multiple reflections are rarely simulated. This means that instead of $n$ products of matrices as it is common with other methods (these matrices being transformation matrices), we have a single product of an $n \times n$ matrix with an $n \times d$ matrix, where $d$ is the total number of dimensions of the coordinates used. Therefore, the SE is both easy to obtain and can provide noticeable performance gains.

We can say that the SE is the relation between each joint's data and the whole robot structure, hence the name, and its properties are key for both forward and inverse kinematics.

# Chapter 6

# Representation of angular and sensor data

In this approach, dealing with sensor data requires some adaptation as this method uses coordinates that are slightly different from those obtained from raw sensor data.

A key concept is also introduced that addresses the issue of moving sensors, as data that comes from sensors is usually referred to a moving reference frame and not a fixed one as proposed in this model.

## 6.1 Sensor data conversion matrices

One issue that emerges when fixed reference frames are used is that most angles provided by sensors and actuators are referred to a reference frame attached to the joint. When the joint moves, the sensor does not adapt its reading so its position in the fixed reference frame may have changed but its reading may not. In order to obtain the proper angles in the fixed reference frames to feed this model a conversion of the sensor data is needed. Usually, each angle of a rotation sensor will have an offset and will be affected by the angles of the previous joints. To solve this, all we have to do is remove this offset and the influence of the angles of previous joints, with an equation such as 6.1.

*Let $\Theta_c$ be the converted angles, $\Theta_s$ the sensor angles and $\Theta_{offset}$ the offset angles of the same sensors,*

$$\Theta_c = \mathcal{C} \times \Theta_s + \Theta_{offset} \tag{6.1}$$

The $\mathcal{C}$ matrix is always square and has a very particular structure, as it portrays a mapping between raw angles from sensors and the angles referred to the fixed RFs.

A common case for a series block is that of equation 6.2. Note that the conversion matrix $\mathcal{C}$ is similar to the structure matrix $\mathcal{S}$, both having block-wise lower diagonal matrices that appear when multiple joints are connected in series.

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix} \tag{6.2}$$

The $\mathcal{C}$ matrix always has an inverse as there is a bijective relation between angles referred to fixed RFs and angles referred to the moving RFs. If the $\mathcal{C}$ matrix has a structure like the one portrayed in equation 6.2, it inverse is also determined and it is given by the equation 6.3 for an $n \times n$ $\mathcal{C}$ matrix.

$$\mathcal{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix} \tag{6.3}$$

These expressions should be used when the robots used require them, and again, linear operations are assumed, so if the sensor is providing data in spread, it should be converted to a linearly-addable angular measure or added using appropriate operators.

After obtaining an angular measure referred to the fixed reference frames, the data is ready to be used in the equations described in the previous chapters. Converting to and from these two angular representations is easy and non-lossy and therefore appropriate for experimental implementations.

This equation will hereon be referred to as the Sensor-Actuator Equation (SAE) and its expression is given by equation 6.1. It is very useful for the practical implementations of this method.

## 6.2 Angular data from sensors and spreads

Another issue when dealing with sensor data is the conversion between the measure the sensors provide and spreads. If the angular sensor provides angles, the conversion can be done using the squared sine and a per-convention function that maps these angles to $L$ matrices. An example function is provided in equation 6.4, considering the $L \times \mathcal{R} \times B$ notation and consequent conventions.

*Given an angle $\theta$ in radians,*

$$B = \begin{bmatrix} \theta \\ \pi \end{bmatrix}, \mathcal{R} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 1 & 1 \\ -1 & 0 \end{bmatrix}$$
$$L = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} & if\ 0 \leq \theta < \frac{\pi}{2} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} & if\ \frac{\pi}{2} \leq \theta \leq \pi \\ \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} & if\ -\frac{\pi}{2} \leq \theta < 0 \\ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} & if\ -\pi \leq \theta < -\frac{\pi}{2} \end{cases} \tag{6.4}$$

Functions like this one can be obtained for any convention used for rotation and provide a bijective mapping

to convert angular data from and to $L \times \mathcal{R} \times B$ notation. Implicit in this is the fact that for each angle there is only one spread and vice versa.

A different case appears if the angular sensor has, for instance, a limited number of steps per lap. Instead of converting steps to angles and operating on them as was previously described, we can obtain an exact spread for each step, some without using any circular functions. This is especially useful for implementing this method on chips where function calls are an expensive resource. For example, consider a $16$ step motor. Each quarter lap has $4$ steps, so this gives the fixed mapping in equation 6.5, considering that steps progress in a counterclockwise manner. Note that only one value uses a circular function and it can be hardcoded to avoid function calls.

$$s = \begin{cases} 0 \; if \; steps = 0 \; or \; 8 \\ sin(\frac{\pi}{8})^2 \; if \; steps = 1, 7, 9 \; or \; 15 \\ 0.5 \; if \; steps = 2, 6, 10 \; or \; 14 \\ 1 - sin(\frac{\pi}{8})^2 \; if \; steps = 3, 5, 11 \; or \; 13 \\ 1 \; if \; steps = 4 \; or \; 12 \end{cases} \tag{6.5}$$

A simple mapping of $L$ matrices for $L \times \mathcal{R} \times B$ notation is also possible, using an identical procedure as with angles. An example is described in equation 6.6, assuming identical conventions to those of equation 6.4.

$$L = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} & if \; steps = 1, 2, 3, 4 \\ \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} & if \; steps = 5, 6, 7, 8 \\ \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} & if \; steps = 13, 14, 15, 16 \\ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} & if \; steps = 9, 10, 11, 12 \end{cases} \tag{6.6}$$

These mappings can be seen as adding an auxiliary dimension to the angle-to-spread conversion allowing an easier conversion between the two.

Using these methods, sensor data from a robot can be approached in a generic manner, supporting the use of this method in practice.

# Chapter 7

# Analysis of constraints

A robot structure may have three kinds of constraints:

- **Workspace constraints** that limit the possible poses of the robot in a given workspace;
- **Actuator constraints** that limit the command and sensor values to a limited range;
- **Structural constraints** limit the poses that a robot can assume due to connections between parts of its structure, limiting its motion.

The first two are the simplest and involve a simple logic test of the values being simulated.

Workspace constraints can be tested by setting up a set of valid points and then verifying that all points of the robot belong to that set. More formally, let $W$ be the set of valid workspace points and $P_f$ the set of all robot points in the workspace reference frame, $P$ is valid if and only if $\forall_{P \in P_f} P \in W$.

Identical reasoning can be done for actuator constraints. Instead of a set of valid workspace points, a set of valid per-joint angles is needed. In a robot with $n$ joints, let $A_i$ be the set of valid angles of joint $i$ and $\theta_i$ the angle for joint $i$. $\theta_i$ is valid if and only if $\forall_{i \in [1,n]} \theta_i \in A_i$.

The last case is slightly more complex as it involves distances between points. If a point is connected to another point by a rigid block then that distance (and quadrance) is fixed. Since the interest are real world objects, the distance concept used is the square root of the per-coordinate difference squared (Euclidean distance). For calculation simplicity quadrances will be used for constraints as this will avoid the square root function without altering the results.

The points to be tested are, again, the final robot points $P_f$ in the workspace reference frame.

Consider a point $P_{f_1}$ with $c$ coordinates connected to point $P_{f_2}$ by a rigid block with length $K_1$. This means that the quadrance between $P_{f_1}$ and $P_{f_2}$ must be $K_1^2$. Let $D$ be the distance between $P_{f_1}$ and $P_{f_2}$ ($D = P_{f_1} - P_{f_2}$). The final points are valid with respect to the structural constraints if and only if $D \bullet D \times \mathbb{1}_{c,1} = K_1 \bullet K_1$.

If more than one constraint exists, all can be placed in a $D$ column vector that has all distances between constrained points. A coherent $K$ vector must be set up that has all valid distance values for the robot structure in question. The points will only be valid if and only if $D \bullet D \times \sigma = K \bullet K$.

## 7.1 The Constraint Equations

For a robot with $n$ joints and $c$ coordinates, let $P_f$ be set of the final points of a robot in the workspace reference frame, $\theta_i$ the angle being tested for joint $i$, $W$ the set of valid workspace points, $A_i$ the set of valid angles for actuator $i$, $D$ the distance vector with the distances between constrained points and $K$ the corresponding fixed distance values for those points. $\sigma$ is an auxiliary matrix for the calculation of distances and allows for restrictions with more than one value (for instance, with position and attitude).

Data provided for a robotic structure simulation will be valid if and only if the set of rules of equation 7.1 is respected.

$$\begin{cases} \forall_{P \in P_f} P \in W \\ \\ \forall_{i \in [1,n]} \theta_i \in A_i \\ \\ D \bullet D \times \sigma = K \bullet K \end{cases} \tag{7.1}$$

The distance vector $D$ can be obtained from the final point vector $P_f$ by matrix multiplication. For the case described above, the matrix formulation is provided in equation 7.2.

$$D = \begin{bmatrix} 1 & -1 \end{bmatrix} \times \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \tag{7.2}$$

This matrix is not unique since distance can be measured from $P_{f_1}$ to $P_{f_2}$ or vice-versa. The results however are the same as the $D$ vector is posteriorly squared. For more complex structures the $D$ vector can still be obtained from a matrix multiplication, so, in a generic form the $D$ vector can be calculated using the equation 7.3 where $\Delta$ is a matrix that calculates distances between the points in the $P_f$ vector.

$$D = \Delta \times P_f \tag{7.3}$$

Examples are provided in section 8.1 for better understanding of this process.

This formulation of restrictions allows a very straightforward way to test the validity of a set of actuator data provided for a robot with little computational expense.

But if the only controllable variable of a block is its radius, a problem emerges when we need to know if there are angles that allow for a given radius to be fed to that block as a command.

To solve this, section 8.1.3 has a small example describes how to test if there are possible angles and if so, what are they. The solution involves solving multi-variable quadratic equations and the results depend on the structure provided.

Regardless of this fact, a closed form solution is possible in some cases, meaning that a parallel block with restrictions can be solved without using numeric approximations and optimizations using equations 7.1 and 7.3.

# Chapter 8

# Forward Kinematics

Perhaps the best way to summarize this model and its consequent implementation is to provide a step-by-step algorithm from the sensor data obtained to the final simulation points.

The key expressions are given by equations 8.1, 8.2 and 8.3 and encompass all of the model developed throughout this work.

- The Joint Equation and $perm$ function:

$$P_i^* = perm(L_i \times \mathcal{R}_i \times B_i) \tag{8.1}$$

- The Structure Equation:

$$P_f = \mathcal{S} \times P^* \tag{8.2}$$

- The Sensor-Actuator Equation:

$$\Theta_c = \mathcal{C} \times \Theta_s + \Theta_{offset} \tag{8.3}$$

- The Constraint Equations:

$$\begin{cases} \forall_{P \in P_f} P \in W \\ \forall_{i \in [1,n]} \theta_i \in A_i \\ D \bullet D \times \sigma = K \bullet K \end{cases} \tag{8.4}$$

Finally, the following steps summarize a typical usage of the model using the coordinates described in this work:

1. **Set up the system used**:
   (a) Define the Base Matrices for each joint by inspecting the actuators of the robot;
   (b) Define the Reference Frame Matrices for each joint by inspecting the workspace;
   (c) Define all constraints for the Constraint Equations by inspecting the Robot and the workspace;

(d) Define the Structure Matrix and the Conversion Matrix of the Structure Equation and the Sensor-Actuator Equation, respectively;

2. **Simulate**:

   (a) Process sensor data with the Sensor Equation obtaining corrected angles;

   (b) Obtain the simulation of each joint by applying the Joint Equation to each joint and, if desired, combine the reflections with the $perm$ operation;

   (c) Combine all joint simulations by applying the Structure Equation;

   (d) Use the data as desired and repeat from step 2a if needed.

## 8.1 Examples

For all examples, block 2 of the steps described previously is always the same as the simulation algorithm does not change. Also, only one reflection will be considered each time so $P^* = perm(P)?P$.

### 8.1.1 A snake

Consider a snake with $n$ joints each with free $\theta_i$ and $\varphi_i$ and fixed radius $r_i$.

1. (a) For each joint $i$, its Base Matrix will be $B_i = B_{P_i} \bigoplus B_{E_i}$ where

$$B_{P_i} = \begin{bmatrix} s_{\theta_i} s_{\varphi_i} R_i & 0 & 0 \\ 0 & (1 - s_{\theta_i}) s_{\varphi_i} R_i & 0 \\ 0 & 0 & (1 - s_{\varphi_i}) R_i \end{bmatrix}$$

,

$$B_{E_i} = \begin{bmatrix} S^{-1}(s_{\theta_i}) & 0 & 0 \\ 1 & 0 & 0 \\ 0 & S^{-1}(\frac{1}{1+s_{\theta_i}\frac{s_{\varphi_i}}{1-s_{\varphi_i}}}) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & S^{-1}(\frac{1}{1+(1-s_{\theta_i})\frac{s_{\varphi_i}}{1-s_{\varphi_i}}}) \\ 0 & 0 & 1 \end{bmatrix}$$

   and $R_i = r_i^2$;

   (b) For each joint $i$, its Reference Frame Matrix will be $R_i = \left[ \begin{array}{c|c} R_{C_i} & R_{E_i} \end{array} \right]$ were

$$R_{C_i} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

and

$$R_{E_i} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ -1 & \pi & -1 & 0 & 1 & 0 \\ 1 & -\pi & -1 & 0 & -1 & \pi \\ -1 & 0 & 1 & 0 & -1 & \pi \\ 1 & 0 & -1 & \pi & -1 & 0 \\ -1 & \pi & 1 & -\pi & -1 & 0 \\ 1 & -\pi & 1 & -\pi & 1 & -\pi \\ -1 & 0 & -1 & \pi & 1 & -\pi \end{bmatrix}$$

. Note that the same convention for rotation was used for all joints but that needs not be;

(c) The Structure Matrix and Conversion Matrix are the same as there are no linearly dependant blocks or parallel blocks and its structure has been discussed previously, it is a simple lower diagonal matrix, in this case, $n \times n$, so $\mathcal{S} = \mathcal{C} = \mathbb{L}_n$.

### 8.1.2 A standard ROB3 model and method comparison.

Consider a standard ROB3 robot. In this example, the key aspect is to clarify how to portray motion in a single angular dimension in this model. So, first, the joints considered are depicted on figure 8.1. Both steps 1a and 1b are the same, as all joints also have radial movement and standard reference frames are used. A different attitude base matrix is used to obtain results that are comparable to those of the standard model, provided in equation 8.5.

$$B_{euler} = \begin{bmatrix} S^{-1}(s_\theta) & 0 & 0 \\ 1 & 0 & 0 \\ 0 & S^{-1}(s_\varphi) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & S^{-1}(s_\varphi) \\ 0 & 0 & 1 \end{bmatrix} \tag{8.5}$$

So the big difference is when angular data is fed to the model: instead of feeding all angles, some of them will be fixed when the joints do not move in that angle. Table 8.1 has the data for each joint and the fixed angle to be fed to the Sensor-Actuator Equation. In order to represent blocks that vary only in attitude a zero radius block is added.

Table 8.1: ROB3 joint data

| Joint | Radius (mm$^2$) | $\theta$ (rad) | $\phi$ (rad) |
|---|---|---|---|
| 1 | $275^2$ | free | 0 |
| 2 | $200^2$ | 0 | free |
| 3 | $130^2$ | 0 | free |
| 4 | $130^2$ | 0 | free |
| 5 | 0 | $\frac{\pi}{2}$ | free |

Consider that the sensors read the angular data of equation 8.6. Fixed values are between parenthesis.

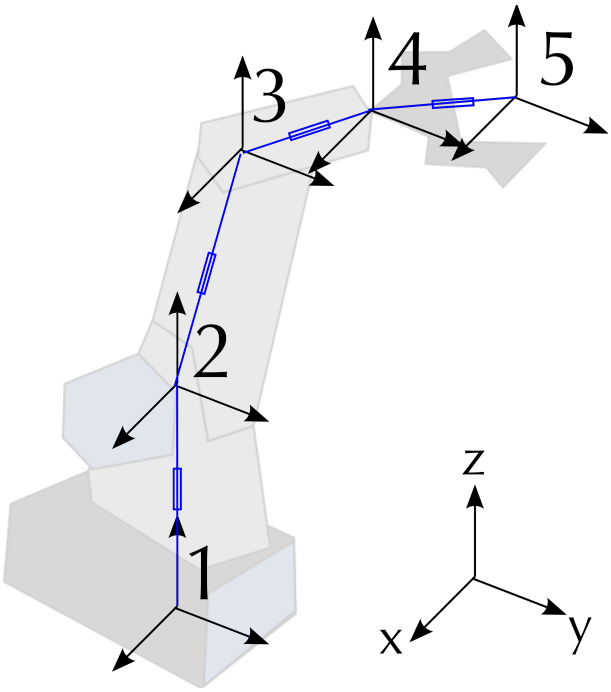


Figure 8.1: Joints considered for the ROB3 robot

$$\begin{bmatrix} \theta_1 & \varphi_1 \\ \theta_2 & \varphi_2 \\ \theta_3 & \varphi_3 \\ \theta_4 & \varphi_4 \\ \theta_5 & \varphi_5 \end{bmatrix} = \begin{bmatrix} \frac{\pi}{4} & (0) \\ (0) & \frac{\pi}{8} \\ (0) & -\frac{\pi}{4} \\ (0) & \frac{\pi}{3} \\ (\frac{\pi}{2}) & \pi \end{bmatrix} \tag{8.6}$$

Next, we apply the Sensor-Actuator Equation. First, for $\theta$ angles, only joint 5 is unaffected, so the correction needed is provided in equation 8.7.

$$\begin{bmatrix} \theta_{c_1} \\ \theta_{c_2} \\ \theta_{c_3} \\ \theta_{c_4} \end{bmatrix} = \mathbb{L}_4 \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} = \begin{bmatrix} \frac{\pi}{4} \\ \frac{\pi}{4} \\ \frac{\pi}{4} \\ \frac{\pi}{4} \end{bmatrix} \tag{8.7}$$

Next, since the $\varphi$ angles of joints $2$, $3$ and $4$ are referred to the horizontal plane, so joint $1$ has no effect on them. Therefore, the Conversion Matrix for $\varphi$ affects only joints $2$, $3$ and $4$. Therefore, only $\varphi$ values need conversion, so the Sensor-Actuator Equation obtained is presented in equation 8.12.

$$\begin{bmatrix} \varphi_{c_2} \\ \varphi_{c_3} \\ \varphi_{c_4} \end{bmatrix} = \mathbb{L}_3 \times \begin{bmatrix} \varphi_2 \\ \varphi_3 \\ \varphi_4 \end{bmatrix} \tag{8.8}$$

Equation 8.12 yields the final corrected angular values of equation 8.9. These are the values to be converted to the new coordinate system. All values omitted remain identical ($\varphi_c = \varphi$).

$$\begin{bmatrix} \varphi_{c_2} \\ \varphi_{c_3} \\ \varphi_{c_4} \end{bmatrix} = \begin{bmatrix} \frac{\pi}{8} \\ -\frac{\pi}{8} \\ \frac{5\pi}{24} \end{bmatrix} \tag{8.9}$$

After the angular correction, the final angles are provided in equation 8.10.

$$\begin{bmatrix} \theta_1 & \varphi_1 \\ \theta_2 & \varphi_2 \\ \theta_3 & \varphi_3 \\ \theta_4 & \varphi_4 \\ \theta_5 & \varphi_5 \end{bmatrix} = \begin{bmatrix} \frac{\pi}{4} & 0 \\ \frac{\pi}{4} & \frac{\pi}{8} \\ \frac{\pi}{4} & -\frac{\pi}{8} \\ \frac{\pi}{4} & \frac{5\pi}{24} \\ (\frac{\pi}{2}) & \pi \end{bmatrix} \tag{8.10}$$

Now, applying the appropriate lookup function the $L$ matrix and spreads are obtained. Usually the lookup function will be like the one provided in equation 8.11 and it combines the reasoning of equations 6.4 and 6.5.

$$L_i = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & if\ 0 \leq \theta_i < \frac{\pi}{2}, -\frac{\pi}{2} \leq \varphi_i < \frac{\pi}{2} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & if\ \frac{\pi}{2} \leq \theta_i < \pi, -\frac{\pi}{2} \leq \varphi_i < \frac{\pi}{2} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & if\ -\pi \leq \theta_i < -\frac{\pi}{2}, -\frac{\pi}{2} \leq \varphi_i < \frac{\pi}{2} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} & if\ -\frac{\pi}{2} \leq \theta_i < 0, -\frac{\pi}{2} \leq \varphi_i < \frac{\pi}{2} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} & if\ 0 \leq \theta_i < \frac{\pi}{2}, \frac{\pi}{2} \leq \varphi_i < \frac{3\pi}{2} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} & if\ \frac{\pi}{2} \leq \theta_i < \pi, \frac{\pi}{2} \leq \varphi_i < \frac{3\pi}{2} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} & if\ -\pi \leq \theta_i < -\frac{\pi}{2}, \frac{\pi}{2} \leq \varphi_i < \frac{3\pi}{2} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & if\ -\frac{\pi}{2} \leq \theta_i < 0, \frac{\pi}{2} \leq \varphi_i < \frac{3\pi}{2} \end{cases} \tag{8.11}$$

$$s_{\theta_i} = \sin^2 \theta_i$$

$$s_{\varphi_i} = \sin^2 \varphi_i$$

So, using the referred lookup function, the table 8.2 has the values obtained.

Table 8.2: $L$ matrices and $s$ values obtained from the lookup function.

| Joint $i$ | $L_i$ | $s_{\theta_i}$ | $s_{\varphi_i}$ |
|---|---|---|---|
| 1 | $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ | .5 | 0 |
| 2 | $\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ | .5 | 0.14645 |
| 3 | $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$ | .5 | 0.14645 |
| 4 | $\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ | .5 | 0.37059 |
| 5 | $\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ | 1 | 0 |

Now, we can apply the Joint Equation to obtain the values of table 8.3 (transposed for better page fit).

Finally, apply the Structure Equation to the values obtained and the final actuator points are calculated, displayed in table 8.4. Note that in this case the Structure Matrices are the ones provided in equation 8.12, and the attitude matrices are slightly different to provide identical results to those of ROB3. This is due to the fact that attitude is represented differently with these matrices than with regular transformation matrices. However, results are compatible by adapting the Structure Equation.

Table 8.3: Joint Equation values for the ROB3 structure.

| Joint $i$ | $P^T = (L_i \times R_i \times B_i)^T$ |
|---|---|
| 1 | $\begin{bmatrix} \sqrt{.5 * 0} * 275 = 0\ (mm) \\ \sqrt{.5 * 0} * 275 = 0\ (mm) \\ \sqrt{1} * 275 = 275\ (mm) \\ \frac{\pi}{4}\ (rad) \\ 0\ (rad) \\ 0\ (rad) \end{bmatrix}$ |
| 2 | $\begin{bmatrix} \sqrt{.5 * 0.14645} * 200 = 54.120\ (mm) \\ \sqrt{.5 * (0.14645)} * 200 = 54.120\ (mm) \\ -\sqrt{(1 - 0.14645)} * 200 = -184.78\ (mm) \\ \frac{\pi}{4}\ (rad) \\ \frac{7\pi}{8}\ (rad) \\ 0\ (rad) \end{bmatrix}$ |
| 3 | $\begin{bmatrix} -\sqrt{.5 * 0.14645} * 130 = -35.178\ (mm) \\ -\sqrt{.5 * (0.14645)} * 130 = -35.178\ (mm) \\ -\sqrt{(1 - 0.14645)} * 130 = -120.10\ (mm) \\ \frac{\pi}{4}\ (rad) \\ -\frac{7\pi}{8}\ (rad) \\ 0\ (rad) \end{bmatrix}$ |
| 4 | $\begin{bmatrix} \sqrt{.5 * 0.37059} * 130 = 55.96\ (mm) \\ \sqrt{.5 * 0.37059} * 130 = 55.96\ (mm) \\ -\sqrt{(1 - 0.37059)} * 130 = -103.14\ (mm) \\ \frac{\pi}{4}\ (rad) \\ \frac{19\pi}{24}\ (rad) \\ 0\ (rad) \end{bmatrix}$ |
| 5 | $\begin{bmatrix} 0\ (mm) \\ 0\ (mm) \\ 0\ (mm) \\ 0\ (rad) \\ 0\ (rad) \\ \pi\ (rad) \end{bmatrix}$ |

$$\begin{bmatrix} P_{f_x} & P_{f_y} & P_{f_z} \end{bmatrix} = \mathbb{L}_5 \times \begin{bmatrix} P_x & P_y & P_z \end{bmatrix}$$

$$\begin{bmatrix} P_{f_\alpha} \end{bmatrix} = \begin{bmatrix} \mathbb{1}_{5,1} \mid \mathbb{O}_{5,4} \end{bmatrix} \times \begin{bmatrix} P_\alpha \end{bmatrix}$$

$$P_{f_\beta} = \left[ \frac{\mathbb{O}_{1,1} \oplus \mathbb{L}_3}{\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \end{bmatrix}} \right] \times \begin{bmatrix} P_\beta \end{bmatrix} \tag{8.12}$$

$$P_{f_\gamma} = \mathbb{O}_{4,4} \oplus \mathbb{1}_{1,1} \times \begin{bmatrix} P_\gamma \end{bmatrix}$$

$$P_f = \begin{bmatrix} P_{f_x} & P_{f_y} & P_{f_z} & P_{f_\alpha} & P_{f_\beta} & P_{f_\gamma} \end{bmatrix}$$

Table 8.4: Final actuator points of the ROB3 structure.

| Joint $i$ | $P^T = (L_i \times R_i \times B_i)^T$ |
|---|---|
| 1 | $\begin{bmatrix} 0\ (mm) \\ 0\ (mm) \\ 275\ (mm) \\ \frac{\pi}{4}\ (rad) \\ 0\ (rad) \\ 0\ (rad) \end{bmatrix}$ |
| 2 | $\begin{bmatrix} 54.120\ (mm) \\ 54.120\ (mm) \\ 90.224\ (mm) \\ \frac{\pi}{4}\ (rad) \\ \frac{7\pi}{8}\ (rad) \\ 0\ (rad) \end{bmatrix}$ |
| 3 | $\begin{bmatrix} 18.942\ (mm) \\ 18.942\ (mm) \\ -29.882\ (mm) \\ \frac{\pi}{4}\ (rad) \\ 0\ (rad) \\ 0\ (rad) \end{bmatrix}$ |
| 4 | $\begin{bmatrix} 74.902\ (mm) \\ 74.902\ (mm) \\ -133.0162\ (mm) \\ \frac{\pi}{4}\ (rad) \\ \frac{19\pi}{24}\ (rad) \\ 0\ (rad) \end{bmatrix}$ |
| 5 | $\begin{bmatrix} 74.902\ (mm) \\ 74.902\ (mm) \\ -133.0162\ (mm) \\ \frac{\pi}{4}\ (rad) \\ \frac{19\pi}{24}\ (rad) \\ \pi\ (rad) \end{bmatrix}$ |

As a comparison, here is the same data fed to a standard model using the Denavit-Hartenberg (D-H) method. Figure 8.2 has the reference frame definition and tables 8.5 and 8.5 have the D-H parameters and transformation matrices respectively. Angles $\theta_1$ to $\theta_5$ and distances $L_1$ to $L_4$ are each of the structures rigid radiuses ($L_1 = 275, L_2 = 200, L_3 = 130, L_4 = 130$).

Each point can now be obtained, the results are exactly the same as the ones using the FFM and are

Table 8.5: D-H parameters for the ROB3 structure.

| i | $\alpha_{i-1}$ | $a_{i-1}$ | $d_i$ | $\Theta_i$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | $\Theta_1$ |
| 2 | $-90$ | 0 | 0 | $\Theta_2$ |
| 3 | 0 | $L_2$ | 0 | $\Theta_3$ |
| 4 | 0 | $L_3$ | 0 | $\Theta_4$ |
| 5 | 90 | 0 | 0 | $\Theta_5$ |
| 6 | 0 | 0 | $L_4$ | 0 |

Table 8.6: D-H intermediate transformation matrices.

| Matrix | Value |
|---|---|
| ${}^1_0T$ | $\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ |
| ${}^1_0T$ | $\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ |
| ${}^2_1T$ | $\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_2 & -\cos\theta_2 & L_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ |
| ${}^3_2T$ | $\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ |
| ${}^4_3T$ | $\begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & L_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ |
| ${}^5_4T$ | $\begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & L_3 \\ 0 & 0 & -1 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ |
| ${}^6_5T$ | $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ |

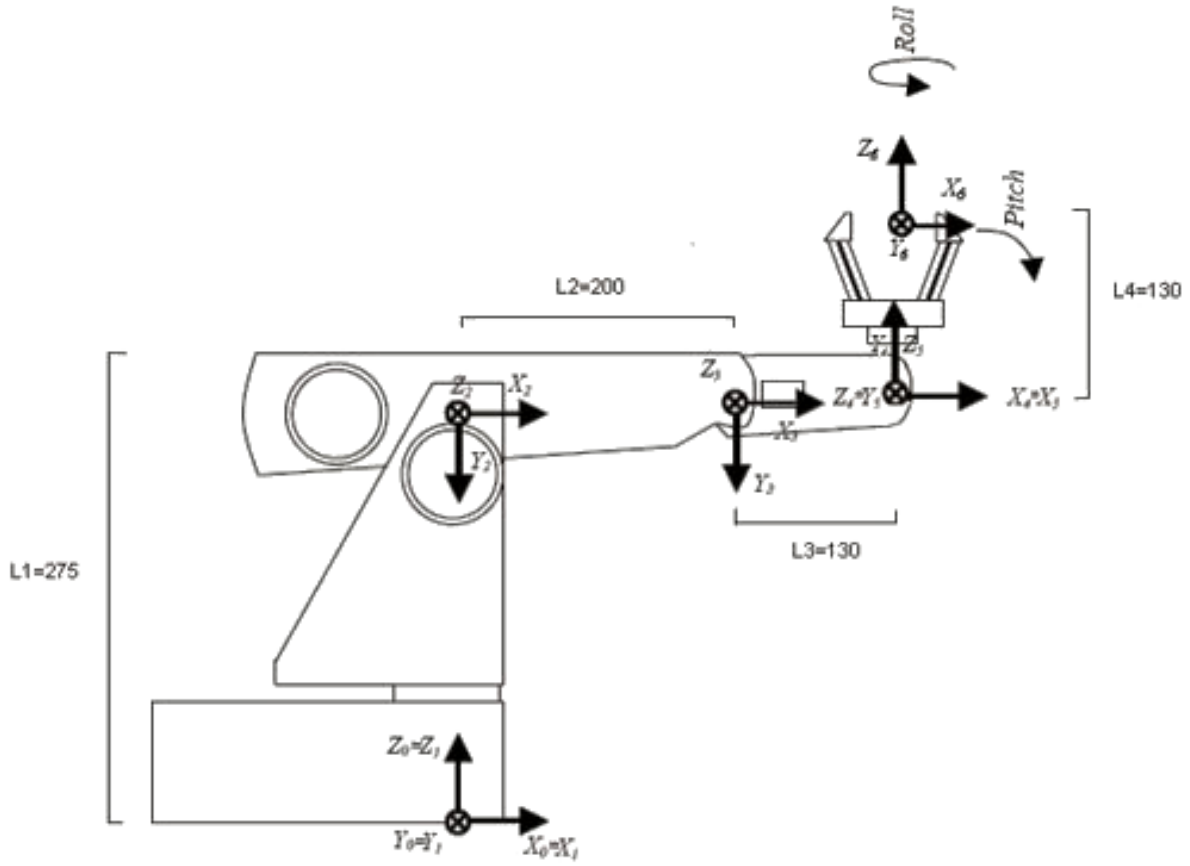


Figure 8.2: Reference Frame definition for ROB3 using D-H

provided in table 8.7. The only difference is the indexing used for both. Attitude isn't included as only the final actuator was analyzed using this method.

Both of the methods were simulated and the performance of the D-H method was significantly better (average 150 times faster). This is due to the fact that the FFM has too much overhead when compared to a small set of transformation matrices. This tends to disappear as the robots grow larger and more complex and benchmarks done for snakes with over many joints demonstrated that the FFM took about 50% the time to simulate than the standard counterpart as the number of joints increased.

### 8.1.3 A simple two-dimensional platform on 3 stands

In order to exemplify the common parallel simulation, consider the structure of figure 8.3.

Joints $2$, $3$ and $5$ have free radial motion and no angular constraints on the stands. Joints $1$ and $6$ are "dummy" joints that represent the different locations of the stands and joint $4$ is a physical connection between joints $2$ and $3$ on one side and joint $5$ on the other. The Reference Frame considered for the workspace will be $R_2$ and the actuator will be the center of joint $4$ ($P_{f_7}$).

First obtain the Base Matrices and Reference Frame Matrices. The ones used are standard, as depicted in equation 8.13.

Table 8.7: Final actuator points of the ROB3 structure using the D-H method.

| Joint $i$ | $P^T = (L_i \times R_i \times B_i)^T$ |
|---|---|
| 1 | $\begin{bmatrix} 0\ (mm) \\ 0\ (mm) \\ 0\ (mm) \end{bmatrix}$ |
| 2 | $\begin{bmatrix} 0\ (mm) \\ 0\ (mm) \\ 275\ (mm) \end{bmatrix}$ |
| 3 | $\begin{bmatrix} 54.120\ (mm) \\ 54.120\ (mm) \\ 90.224\ (mm) \end{bmatrix}$ |
| 4 | $\begin{bmatrix} 18.942\ (mm) \\ 18.942\ (mm) \\ -29.882\ (mm) \end{bmatrix}$ |
| 5 | $\begin{bmatrix} 18.942\ (mm) \\ 18.942\ (mm) \\ -29.882\ (mm) \end{bmatrix}$ |
| 6 | $\begin{bmatrix} 74.902\ (mm) \\ 74.902\ (mm) \\ -133.0162\ (mm) \\ \frac{\pi}{4}\ (rad) \\ \frac{19\pi}{24}\ (rad) \\ \pi\ (rad) \end{bmatrix}$ |

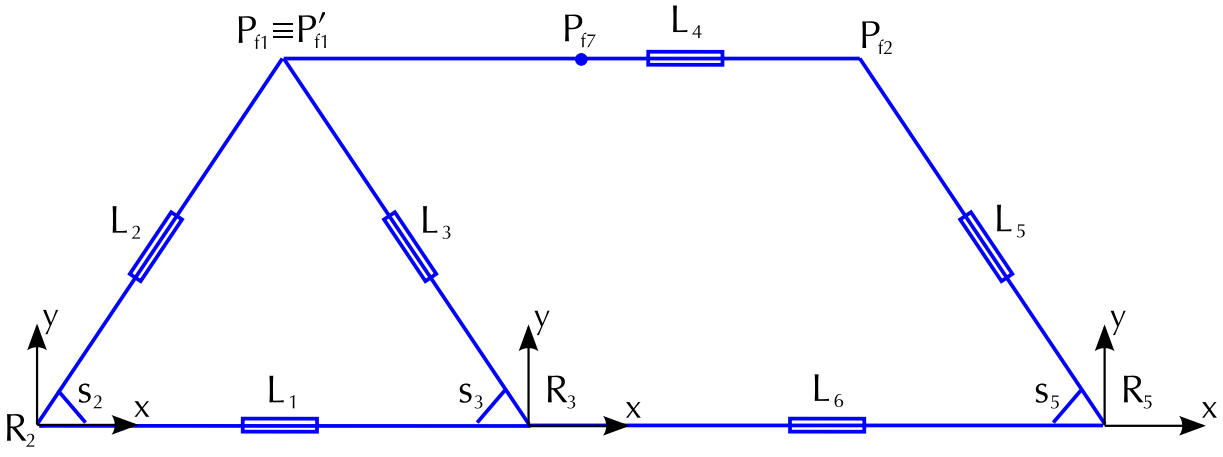


Figure 8.3: A 2D platform on 3 stands

$$B_2 = B_3 = B_5 = \begin{bmatrix} \sqrt{Q_x} & 0 \\ 0 & \sqrt{Q_y} \end{bmatrix} \oplus \begin{bmatrix} S^{-1}(s_\theta) \\ \pi \end{bmatrix}$$

$$R_2 = R_3 = R_5 = \left[\begin{array}{c|c} \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \\ 1 & -1 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 1 & 1 \\ -1 & 0 \end{bmatrix} \end{array}\right] \tag{8.13}$$

Second, set up the Structure Matrix. The matrix obtained may vary depending on the user's convention for the workspace Reference Frame. By using the referred choice the matrix obtained is presented in equation 8.14.

$$P_f = \mathcal{S} \times P$$

$$\begin{bmatrix} P_{f_1} \\ P_{f_1}^{,} \\ P_{f_2} \\ P_{f_7} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ .5 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{bmatrix} \tag{8.14}$$

Third, since this robot has structural constraints, we will need to set up the appropriate constraint equations, provided in equation 8.15.

$$D = \Delta \times P_f$$

$$\Delta = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \end{bmatrix} \tag{8.15}$$

$$\rightsquigarrow D = \begin{bmatrix} P_{f_1} - P_{f_1}^{,} \\ P_{f_1} - P_{f_2} \end{bmatrix}$$

Next, apply the actual restrictions by defining a multi-dimensional equation in terms of the $D$ vector, as presented in equation 8.16. In this case each $P_f$ vector has $3$ coordinates but constraints are only applied to position so $\sigma^T = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$. As for the $K$ vector, it forces that the quadrance between point $P_{f_1}$ and $P_{f_1}^{,}$ to $0$ and quadrance between points $P_{f_1}$ and $P_{f_2}$ to $R^2$.

$$D \bullet D \times \sigma = K \bullet K$$

$$\rightsquigarrow \begin{bmatrix} P_{f_1} - P_{f_2} \\ P_{f_2} - P_{f_3} \end{bmatrix} \bullet \begin{bmatrix} P_{f_1} - P_{f_2} \\ P_{f_2} - P_{f_3} \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ R^2 \end{bmatrix}$$

$$\rightsquigarrow \begin{cases} (x_1 - x_2)^2 + (y_1 - y_2)^2 = 0 \\ \\ (x_2 - x_3)^2 + (y_2 - y_3)^2 = R^2 \end{cases}$$

$$\rightsquigarrow \begin{cases} x_1 = x_2 \\ \\ y_1 = y_2 \\ \\ (x_2 - x_3)^2 + (y_2 - y_3)^2 = R^2 \end{cases} \tag{8.16}$$

$$\rightsquigarrow \begin{cases} \sqrt{s_1 R_1} = \sqrt{s_2 R_2} \\ \\ \sqrt{(1 - s_1) R_1} = \sqrt{(1 - s_2) R_2} \\ \\ \sqrt{s_2 s_3} + \sqrt{(1 - s_2)(1 - s_3)} = \frac{-R^2 + R_2 + R_3}{2\sqrt{R_2 R_3}} \end{cases}$$

These results imply that the angles and radiuses possible for this structure are part of a very small set of numbers described by the final equations of 8.16. Though not possible in this study, it is a key aspect of parallel analysis to further develop these results. It proves however that formal solutions are possible and easily obtainable.

### 8.1.4 A platform on 5 stands

Consider a platform composed of 5 stands, each a series block with 3 joints with free angular motion and one that portrays the different locations of all joints (a *dummy* joint). The top platform is connected to the series blocks with a freely moving connector with no actuators as portrayed in figure 8.4.

As previously discussed, both the Structure Matrix and the Conversion Matrix are the same as all joints have free angular motion. This structure is very similar to the one presented in equation 5.8. Again, Base Matrices and Reference Frame Matrices are the same, and the lookup function used is the same as in section 8.1.2. So, for each series block we have the matrices of equation 8.17.

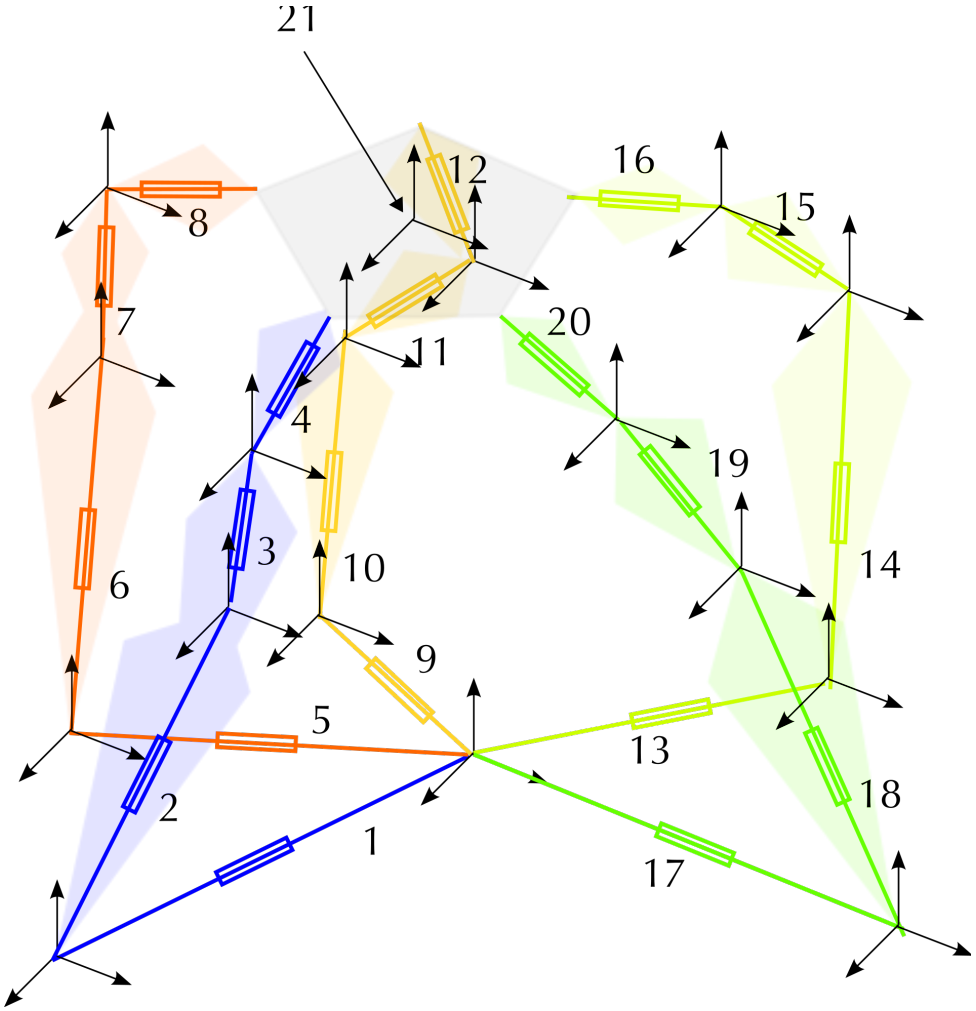


Figure 8.4: A platform on 5 stands

$$
\begin{bmatrix} P_{f_1} \\ P_{f_2} \\ P_{f_3} \\ P_{f_4} \end{bmatrix} = \mathbb{L}_4 \times \begin{bmatrix} P_1^* \\ P_2^* \\ P_3^* \\ P_4^* \end{bmatrix} \quad \begin{bmatrix} P_{f_5} \\ P_{f_6} \\ P_{f_7} \\ P_{f_8} \end{bmatrix} = \mathbb{L}_4 \times \begin{bmatrix} P_5^* \\ P_6^* \\ P_7^* \\ P_8^* \end{bmatrix}
$$

$$
\begin{bmatrix} P_{f_9} \\ P_{f_{10}} \\ P_{f_{11}} \\ P_{f_{12}} \end{bmatrix} = \mathbb{L}_4 \times \begin{bmatrix} P_9^* \\ P_{10}^* \\ P_{11}^* \\ P_{12}^* \end{bmatrix} \quad \begin{bmatrix} P_{f_{13}} \\ P_{f_{14}} \\ P_{f_{15}} \\ P_{f_{16}} \end{bmatrix} = \mathbb{L}_4 \times \begin{bmatrix} P_{13}^* \\ P_{14}^* \\ P_{15}^* \\ P_{16}^* \end{bmatrix} \tag{8.17}
$$

$$
\begin{bmatrix} P_{f_{17}} \\ P_{f_{18}} \\ P_{f_{19}} \\ P_{f_{20}} \end{bmatrix} = \mathbb{L}_4 \times \begin{bmatrix} P_{17}^* \\ P_{18}^* \\ P_{19}^* \\ P_{20}^* \end{bmatrix}
$$

As for the final block, it is the mean of the final actuator points as portrayed in equation 8.18.

$$
P_{f_{21}} = \mathbb{1}_{1,5} \times 0.2 \times \begin{bmatrix} P_4^* \\ P_8^* \\ P_{12}^* \\ P_{16}^* \\ P_{20}^* \end{bmatrix} \tag{8.18}
$$

By combining all these smaller structure matrices into the global one we obtain the matrix of equation 8.19.

$$
\begin{bmatrix} P_{f_1} \\ \vdots \\ P_{f_{20}} \\ P_{f_{21}} \end{bmatrix} = \begin{bmatrix} \mathbb{L}_4 \oplus \mathbb{L}_4 \oplus \mathbb{L}_4 \oplus \mathbb{L}_4 \oplus \mathbb{L}_4 \\ \hline \mathbb{1}_{1,16} \times 0.2 \end{bmatrix} \times \begin{bmatrix} P_1^* \\ \vdots \\ P_{20}^* \end{bmatrix} \tag{8.19}
$$

As seen on equation 8.19, a Structure Matrix always has a very particular structure and can be broken into smaller blocks, each portraying a different block of the robot.

In order to test the constraints, again we apply the Constraint Equations. In this case, only the formulation is provided in equation 8.20 as the full constraint equation system is, like the previous parallel example, not solved. Consider that the pentagonal platform is regular and each side has a length of $L$ and that $\phi$ denotes the golden ratio ($\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$).

$$K = \begin{bmatrix} L \\ \phi L \\ \phi L \\ L \\ L \\ \phi L \\ \phi L \\ L \\ \phi L \\ L \end{bmatrix}, D = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} P_{f_4} \\ P_{f_8} \\ P_{f_{12}} \\ P_{f_{16}} \\ P_{f_{20}} \end{bmatrix}, \sigma = \begin{bmatrix} \mathbb{1}_{3,1} \\ \hline \mathbb{O}_{3,1} \end{bmatrix}$$

$$\rightsquigarrow \begin{cases} (x_4 - x_8)^2 + (y_4 - y_8)^2 + (z_4 - z_8)^2 = L^2 \\ (x_4 - x_{12})^2 + (y_4 - y_{12})^2 + (z_4 - z_{12})^2 = (\phi L)^2 \\ (x_4 - x_{16})^2 + (y_4 - y_{16})^2 + (z_4 - z_{16})^2 = (\phi L)^2 \\ (x_4 - x_{20})^2 + (y_4 - y_{20})^2 + (z_4 - z_{20})^2 = L^2 \\ (x_8 - x_{12})^2 + (y_8 - y_{12})^2 + (z_8 - z_{12})^2 = L^2 \\ (x_8 - x_{16})^2 + (y_8 - y_{16})^2 + (z_8 - z_{16})^2 = (\phi L)^2 \\ (x_8 - x_{20})^2 + (y_8 - y_{20})^2 + (z_8 - z_{20})^2 = (\phi L)^2 \\ (x_{12} - x_{16})^2 + (y_{12} - y_{16})^2 + (z_{12} - z_{16})^2 = L^2 \\ (x_{12} - x_{20})^2 + (y_{12} - y_{20})^2 + (z_{12} - z_{20})^2 = (\phi L)^2 \\ (x_{16} - x_{20})^2 + (y_{16} - y_{20})^2 + (z_{16} - z_{20})^2 = L^2 \end{cases} \tag{8.20}$$

This system contains all valid points for this structure. By using a polar coordinate substitution, it is possible to obtain all radiuses and angles that this structure allows but this problem remains unsolved.

### 8.1.5 An arm and hand

This example provides a simple solution to a very complex structure and hopefully exemplifies the power of this model. Consider the arm and hand-like structure of figure 8.5. Again we will consider the standard coordinates so the Base Matrices, Reference Frame Matrices and lookup functions are the same. In order to approach this structure, the best is to traverse the whole structure in an ordered manner, following the indexes chosen in figure 8.5.

So, for each colored block we have the following Structure and Sensor-Actuator Matrices:

- Indexes $1$ and $2$, a regular series block so $\mathcal{S}_1 = \mathcal{C}_1 = \mathbb{L}_2$;
- Indexes $3$, $4$ and $5$, a series block connected to the first one, so $\mathcal{S}_2 = \mathcal{C}_2 = \left[\begin{array}{c|c} \mathbb{1}_{3,2} & \mathbb{L}_3 \end{array}\right]$;
- Index $6$ is a series block that is connected to blocks $1$ and $2$, so $\mathcal{S}_3 = \mathcal{C}_3 = \left[\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$;
- Indexes $7$, $8$ and $9$ are a series block connected to joint $6$ so $\mathcal{S}_4 = \mathcal{C}_4 = \left[\begin{array}{c|c} \mathcal{S}_3 \otimes \mathbb{1}_{3,1} & \mathbb{L}_3 \end{array}\right]$;
- Indexes $10$, $11$ and $12$ are again a series block connected to joint $6$ so $\mathcal{S}_5 = \mathcal{C}_5 = \left[\begin{array}{c|c} \mathcal{S}_3 \otimes \mathbb{1}_{3,1} & \left[\begin{array}{c|c} \mathbb{O}_{3,3} & \mathbb{L}_3 \end{array}\right] \end{array}\right]$;
- Indexes $13$, $14$ and $15$ are a series block like the previous so $\mathcal{S}_6 = \mathcal{C}_6 = \left[\begin{array}{c|c} \mathcal{S}_3 \otimes \mathbb{1}_{3,1} & \left[\begin{array}{c|c} \mathbb{O}_{3,6} & \mathbb{L}_3 \end{array}\right] \end{array}\right]$;
- Finally, indexes $16$, $17$ and $18$ are identical to the previous so $\mathcal{S}_7 = \mathcal{C}_7 = \left[\begin{array}{c|c} \mathcal{S}_3 \otimes \mathbb{1}_{3,1} & \left[\begin{array}{c|c} \mathbb{O}_{3,9} & \mathbb{L}_3 \end{array}\right] \end{array}\right]$;

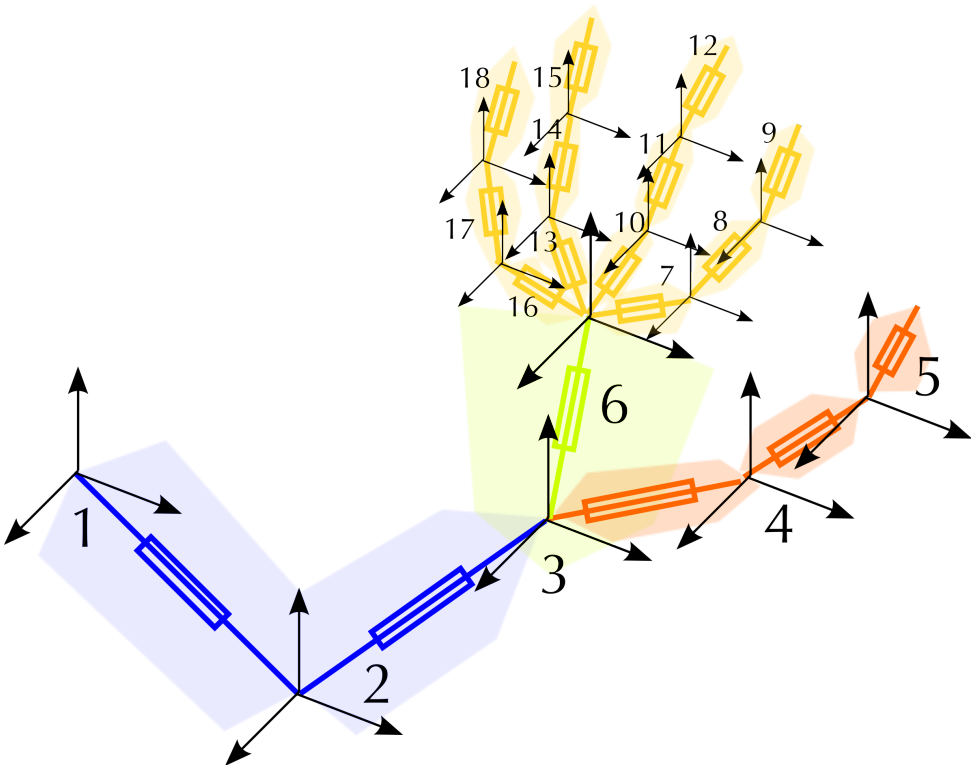


Figure 8.5: An arm and hand

As you can see, the structure of the whole robot was broken down into smaller parts. In order to obtain the final matrices all we have to do is vertically concatenate all the previous matrices and add zeros to the empty space, as portrayed in equation 8.21. A less abstract depiction of those matrices is provided in equation 8.22. Despite being relatively large ($18 \times 18$), their structures are simple and intuitive.

$$\mathcal{S} = \mathcal{C} = \begin{bmatrix} \left[\begin{array}{c|c} \mathcal{S}_1 & \mathbb{O}_{2,16} \end{array}\right] \\ \hline \left[\begin{array}{c|c} \mathcal{S}_2 & \mathbb{O}_{3,13} \end{array}\right] \\ \hline \left[\begin{array}{c|c} \mathcal{S}_3 & \mathbb{O}_{1,12} \end{array}\right] \\ \hline \left[\begin{array}{c|c} \mathcal{S}_4 & \mathbb{O}_{3,9} \end{array}\right] \\ \hline \left[\begin{array}{c|c} \mathcal{S}_5 & \mathbb{O}_{3,6} \end{array}\right] \\ \hline \left[\begin{array}{c|c} \mathcal{S}_6 & \mathbb{O}_{3,3} \end{array}\right] \\ \hline \mathcal{S}_7 \end{bmatrix} \tag{8.21}$$

$$\mathcal{S} = \mathcal{C} = \left[\begin{array}{cc|cccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \tag{8.22}$$

# Chapter 9

# Conclusions

The algorithm and model presented differs from common models as it attempts to provide a set of equations and algorithms that do not change from robot to robot. Whenever a new robot needs to be analyzed, all changes happen in the matrices ($\mathcal{S}$, $\mathcal{C}$, $\sigma$ and $\Delta$). This favors the analysis and development of new robotic structures, and due to the nature of the FFM, the complexity of a structure adds little to the complexity of the referred matrices. In fact, this model not only makes it easier to simulate complex structures but also favors them, both for Forward and Inverse Kinematics.

## 9.1 Possibilities for Inverse Kinematics

Due to the algebraic formulation obtained, Inverse Kinematics can be obtained by explicitly inverting the $\mathcal{S}$ and $\mathcal{C}$ matrices. If less than all the extremity points of the robot are desired, the inversion of the $\mathcal{S}$ matrix is an optimization problem by definition, and has optimal solutions described by its pseudo-inverse. In fact, if a joint was entirely free (free angles and radiuses), the solution to its optimal inverse, whatever the structure of the robot, would be the said inversion of the Structure Matrix. Obviously, reality is far from that simplicity and the inversion of the $\mathcal{S}$ matrix provides the whole optimization problem to be solved. By including all constraints, these matrices are enough to set up a numeric or formal solution, depending on the case. The fact that the $\mathcal{C}$ is used does not change this as it always has full rank.

This means that Inverse Kinematics using the FFM pave way for a globally optimal, generic algorithm for Inverse Kinematics of any robot. This work, however, does not approach that topic and it is a key point for further study.

## 9.2 Applying the model developed using other coordinates

The FFM developed is generic in a way that allows for it to be used with any type of coordinates, be it Cartesian coordinates or Quaternions. All that needs to be done is replace the $P^*$ matrices with the appropriate replacement, with each coordinate in a column and each tuple in a row. Most coordinate systems have only one tuple, so $P^*$ matrices would end up being a simple row vector. The only restriction is that whatever the coordinates used, both the appropriate sum operators are used and the appropriate corrective equations are possible (i.e., it must be possible to convert sensor data referred to the moving reference frames to data referred to the fixed reference frames).

For example, for 2D Cartesian Coordinates, one would obtain $P^* = \begin{bmatrix} x & y \end{bmatrix}$. There are no restrictions to the data used to feed the FFM beyond the ones previously discussed, and this is a major flexibility point that favors the use and implementation of this model in various systems.

## 9.3 Analysis of gains obtained by using this model

The main paradigm shift, that of keeping the reference frames fixed during motion, turned out to be a major difference that yielded very exciting results. A complex product of matrices was turned into a linear system of sums of matrices. Even though some adaptations are needed, this major difference in representation yields major differences in the results, mainly when structures are very complex. Constraints are easier to analyze and test, less operations are needed as matrix sums are used instead of matrix products and explicit inversion of the kinematics model is also possible.

But to gain in these aspects, the major loss is the size of the matrices used, that are much bigger than the standard transformation matrices and grow with the complexity of the robot, adding overhead to simple simulations. Despite this fact, memory in modern systems is rarely an issue and all matrices have a very simple numeric structure, usually only having the values $1$, $0$ and $-1$, that leave room for good optimization for the storage of these matrices.

No optimization analysis was done, so the performance gains are only due to the different representation obtained. There are, with no doubt, several possibilites for performance improvement of the algorithms and expressions, as the very particular structure of the matrices leaves room for optimization.

All of these aspects are major gains in terms of both representation and implementation, the major loss being in terms of the size of the matrices and initial overhead.

A last, and also major gain, is the fact that this new representation gives a new and very different point of view of problems that have been studied for a long time, so hopefully this model can prove its value by the simple fact that it looks at old problems in a new way.

## 9.4 The importance and role of RT in this work

The title of this work might induce one into thinking that the main aspect of this work is RT. This was the fact when this work started, but as the study evolved and results began to be obtained, the role of RT became auxiliary. In fact, it remains highly useful for solving the expressions obtained, but they could as easily be obtained using other coordinates. But no part of this study would have been possible if the initial study using RT had not been done. That alone proves that RT is key to this work, and even though it has evolved in different unexpected directions it remais a very important part of this study.

The main role of RT in this work was that it forced a different approach than the standard, due to difficulties that had to be resolved, that ended up forcing the paving of a new path of study.

Also, the path chosen to apply RT to this problem is one of many possible. Perhaps one of the most interesting that was not followed was to consider a robot with $J$ joints a polygon with $J+1$ edges where the last one is the edge corresponding to the actuator point. Despite other options, it is my belief that the choice of the initial path for establishing the coordinate system used had a tremendous impact on establishing the FFM.

## 9.5 Future work and perspectives

A major work to be developed is the analysis of Inverse Kinematics. The problems that Inverse Kinematics presents are very interesting when put in terms of FFM and $L \times \mathcal{R} \times B$ coordinates and, as previously discussed, allow for globally optimal solutions for any structure.

Another work at hand is the development of the simulation toolkit that was used to make some of the calculations of this work. Its license and online dimension are open to collaboration, as the biggest interest is that it can become useful as soon as possible, providing both direct and inverse simulation of robots in a generic manner.

A final, and perhaps also key problem that isn't yet solved is to obtain the generic constraint solutions for parallel structures, if they exist.

As a final thought, many doors have been opened in this work, most of which remain unexplored, and though the initial objectives have been accomplished the desire and sense of incompleteness remains, a desire to traverse all those doors that were opened. This is, perhaps, the most interesting aspect of this work, the doors it opens and the landscapes it explores.

# Bibliography

[Wildberger] Wildberger, N. J., "Divine Proportions: Rational Trigonometry to Universal Geometry", Wild Egg, 2005

[Wildberger 2] Wildberger, N. J., "A Rational Approach to Trigonometry", http://web.maths.unsw.edu.au/ norman/papers/RationalTrig.pdf, 2005

# Appendix A

# Proofs

## A.1 Proofs and discussions of equations provided

Here are included proofs of some of the expressions deduced in this work. The expressions included here are the ones that may be less clear, as most are self-explanatory.

**Proof of equation 2.1** Let $A$ and $B$ be two quadrances obtained from distances $a$ and $b$, respectively.

The quadrance $C$ given by $c^2$ with $c = a + b$ is $C = (a + b)^2$. Since $a = \pm\sqrt{A}$ and $b = \pm\sqrt{B}$, then $C = (\pm\sqrt{A} \pm \sqrt{B})^2$. This means that for each two original quadrances there are four choices for their sum. Even though two of these are the same ($(-a - b)^2$ and $(a + b)^2$), the differing signs imply that choices must be made depending on the original $a$ and $b$ signs. But in the framework used, the signs are separated from the quadrances and distances and stored in the RFM, so we just need one of the cases. The remaining cases can be obtained using RFMs. So, for simplicity, choose the case when both $a$ and $b$ are positive (to keep the referred mirroring behaviour). The solution obtained is given by the following equations, as the exact same proof can be done for the subtraction.

$$C = A \boxplus B = (\sqrt{A} + \sqrt{B})^2$$
$$C = A \boxminus B = (\sqrt{A} - \sqrt{B})^2.$$

**Proof of equation 2.2** In order to obtain a way to sum spreads with identical reasoning as angles, the simplest way is to use an angular reasoning and then convert it to spreads.

Let $\alpha$ and $\beta$ be two generic angles whose spreads are $s_\alpha$ and $s_\beta$ respectively.

The spread $s_\gamma$ of an angle $\gamma = \alpha + \beta$ is given by $(\sin(\alpha \pm \beta))^2 = (\sin(\alpha)\cos(\beta) \pm \sin(\beta)cos(\alpha))^2 \rightsquigarrow (\sqrt{\sin^2(\alpha)}\sqrt{\cos^2(\beta)} \pm \sqrt{\sin^2(\beta)}\sqrt{\sin^2(\alpha)})^2 = (\sqrt{s_\alpha(1 - s_\beta)} \pm \sqrt{s_\beta(1 - s_\alpha)})^2$, yielding

$$s_A \oplus s_B = (\sqrt{s_A(1 - s_B)} + \sqrt{s_B(1 - s_A)})^2$$
$$s_A \ominus s_B = (\sqrt{s_A(1 - s_B)} - \sqrt{s_B(1 - s_A)})^2.$$

Again, note that the conversion $\sin(\theta) = \sqrt{\sin^2(\theta)}$ is only possible because RFMs are used and signs are again separated from the value. This conversion can be seen as the referred modulus for rotation that though

less intuitive also exhibits a mirroring behaviour similar to that of the quadrance sum.

**Proof of equation 3.5** There are three Euler angles, each corresponding to a rotation about an axis. For each there is an equivalent spread: $\alpha$ is the rotation of the $y$-axis in the direction of the $x$-axis; $\beta$ the rotation of the $z$-axis toward the $x$-axis and $\gamma$ the rotation of the $z$-axis toward the $y$-axis.

The polar definition of [Wildberger] is used.

So, in terms of spreads and quadrances, the equivalent spreads are:

$$s_\alpha = \frac{q_y}{q_x + q_y} = s_\theta$$

$$s_\beta = \frac{q_z}{q_x + q_z} = \frac{1}{1 + \frac{q_x}{q_z}} = \frac{1}{1 + \frac{s_\theta s_\varphi R}{(1-s_\varphi)R}} = \frac{1}{1 + s_\theta \frac{s_\varphi}{1-s_\varphi}}$$

$$s_\gamma = \frac{q_z}{q_y + q_z} = \frac{1}{1 + \frac{q_y}{q_z}} = \frac{1}{1 + \frac{(1-s_\theta)s_\varphi R}{(1-s_\varphi)R}} = \frac{1}{1 + (1-s_\theta)\frac{s_\varphi}{1-s_\varphi}}$$

$$\rightsquigarrow \begin{cases} s_\alpha = s_\theta \\ s_\beta = \frac{1}{1+s_\theta \frac{s_\varphi}{1-s_\varphi}} \\ s_\gamma = \frac{1}{1+(1-s_\theta)\frac{s_\varphi}{1-s_\varphi}} \end{cases}$$

This equation gives us all euler angles in terms of the joint angles also used for position and whose data comes directly from motors.

**Proof of equation 3.7** Consider a point whose 3D quadrance coordinates are $\begin{bmatrix} Q_x & Q_y & Q_z \end{bmatrix}$ and whose 3D distance coordinates are $\begin{bmatrix} x & y & z \end{bmatrix}$. These coordinates are obtained from a number and its sign. The number is obtained from an always positive expression, in this case, the numbers in the diagonal of the base matrix. Now, consider a cartesian reference frame. Each octant has a set of signs that all points in it share, for instance, in the seventh (all negative) octant, the signs are $(---)$, so, to obtain a coordinate with signs and values, we need to multiply each of these signs with each of the numbers in the BM. So, in this case, we would have the following for quadrance and position:

$$\begin{bmatrix} Q_x & Q_y & Q_z \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix} \times \begin{bmatrix} s_\theta s_\varphi R & 0 & 0 \\ 0 & (1-s_\theta)s_\varphi R & 0 \\ 0 & 0 & (1-s_\varphi)R \end{bmatrix}$$

$$\begin{bmatrix} x & y & z \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix} \times \sqrt{(} \begin{bmatrix} s_\theta s_\varphi R & 0 & 0 \\ 0 & (1-s_\theta)s_\varphi R & 0 \\ 0 & 0 & (1-s_\varphi)R \end{bmatrix}).$$

If we repeat the process for each octant of the said reference frame, we will obtain a set of eight $1 \times 3$ vectors of $1$ and $-1$. By ordering them into a matrix, we have a representation of all octants of the reference frame used. The order used plays little role in the rest of the model, but if they are ordered, for instance, according to the octant number, the final matrix will have a very intuitive form and will represent the convention used for the signs of points in the octants. So, since the cartesian frame is ordered, from the first to the last, the set of signs is $(+++)$, $(-++)$, $(--+)$, $(+-+)$, $(++-)$, $(-+-)$, $(---)$ and $(+--)$ and the set of

vectors is $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$, $\begin{bmatrix} -1 & 1 & 1 \end{bmatrix}$, $\begin{bmatrix} -1 & -1 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & -1 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 1 & -1 \end{bmatrix}$, $\begin{bmatrix} -1 & 1 & -1 \end{bmatrix}$, $\begin{bmatrix} -1 & -1 & -1 \end{bmatrix}$ and $\begin{bmatrix} 1 & -1 & -1 \end{bmatrix}$.

Arranging them into the R matrix, we then obtain all eight points corresponding to the signed correspondents of the numbers in the base matrix using the following formula:

$$B = \begin{bmatrix} s_\theta s_\varphi R & 0 & 0 \\ 0 & (1-s_\theta)s_\varphi R & 0 \\ 0 & 0 & (1-s_\varphi)R \end{bmatrix},\ \mathcal{R} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} Q_{x1} & Q_{y1} & Q_{z1} \\ Q_{x2} & Q_{y2} & Q_{z2} \\ Q_{x3} & Q_{y3} & Q_{z3} \\ Q_{x4} & Q_{y4} & Q_{z4} \\ Q_{x5} & Q_{y5} & Q_{z5} \\ Q_{x6} & Q_{y6} & Q_{z6} \\ Q_{x7} & Q_{y7} & Q_{z7} \\ Q_{x8} & Q_{y8} & Q_{z8} \end{bmatrix} = \mathcal{R} \times B$$

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \\ x_6 & y_6 & z_6 \\ x_7 & y_7 & z_7 \\ x_8 & y_8 & z_8 \end{bmatrix} = \mathcal{R} \times \sqrt{(}B).$$

RFMs are a mathematical description of the conventions used for what motion is considered positive or negative and can be obtained by simple inspection of the workspace.

**Proof of equation 3.8** As previously discussed, the spread remains the same anywhere you measure it, so it does not need a reference frame. But since for a spread there are infinite angles, considering that the spread is a modulus for rotation, we can take a similar approach as we did for position. So, consider that for each spread for an Euler angle there are four Euler angles in radians that are $\{S^{-1}(s), \pi - S^{-1}(s), \pi + S^{-1}(s), -S^{-1}(s)\}$.

If we consider the $(+++)$ octant, the angles will be:

$$B = \begin{bmatrix} S^{-1}(s_\theta) & 0 & 0 \\ 1 & 0 & 0 \\ 0 & S^{-1}(\frac{1}{1+s_\theta \frac{s_\varphi}{1-s_\varphi}}) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & S^{-1}(\frac{1}{1+(1-s_\theta)\frac{s_\varphi}{1-s_\varphi}}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \times B.$$

This can be again repeated for each octant of a reference frame, and for each one we will obtain a different row vector that multiplies the base matrix. If we use the same reasoning as before to put them all together, we will obtain the following $R$ matrix:

$$R = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ -1 & \pi & -1 & 0 & 1 & 0 \\ 1 & -\pi & -1 & 0 & -1 & \pi \\ -1 & 0 & 1 & 0 & -1 & \pi \\ 1 & 0 & -1 & \pi & -1 & 0 \\ -1 & \pi & 1 & -\pi & -1 & 0 \\ 1 & -\pi & 1 & -\pi & 1 & -\pi \\ -1 & 0 & -1 & \pi & 1 & -\pi \end{bmatrix}$$

$$\begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \\ \alpha_4 & \beta_4 & \gamma_4 \\ \alpha_5 & \beta_5 & \gamma_5 \\ \alpha_6 & \beta_6 & \gamma_6 \\ \alpha_7 & \beta_7 & \gamma_7 \\ \alpha_8 & \beta_8 & \gamma_8 \end{bmatrix} = R \times B.$$

If a different convention for the four angles that correspond to each spread is used, a different RFM will emerge, but the BM remains the same. This allows for flexibility when choosing conventions for rotations. Again, the order in which the lines are placed in the RFM is a choice of the user, allowing for more intuitive choices.

**Explanation of equations 4.2, 4.3 and 4.4** These equations are a cooked up way of creating line by line uniquely combinable sets of matrices. So, first of all, to count all combinations the following function was set up:

$$CC(a,b) = \prod_{i=a}^{b} rank(L_i).$$

The $rank(L_i)$ is used to count the total of lines a given point matrix has, then the product is done over

the points to mix line-by-line with each other. This equation is obvious as if we have $n_1$, $n_2$, ..., $n_J$ total points, the total number of combinations is $n_1 \times n_2 \times \ldots \times n_J$. The $CC$ function is just an auxiliary function to count the total number of unique combinations possible for a given range $a$ to $b$.

If we want to count the total combinations of points before and after a current point, we can use the referred function to calculate the *inner* and *outer* combinations as previously discussed.

$$I = CC(1, i-1);\ O = CC(i+1, n)$$

With these auxiliary numbers, the following formula creates matrices with repeated elements that can be combined with each other:

$$P_i^* = \left( \sum_{j=1}^{rank(L_i)} \left( M_j \times P_i \otimes \mathbb{1}_{I,1} \otimes M_j^T \right) \right) \otimes \mathbb{1}_{O,1}.$$

What this does is repeat each line of the points in the $P_i$ matrix $I$ times and repeat the whole block $O$ times. As we progress along the index, the varying $I$ and $O$ totals assure us that each matrix will have distinct patterns of copies than the others.

First, a single line is selected from a $P_i$ set of positions with the product with the $M_j$ matrix. Then, this line is repeated $I$ times using the Kronecker product with a column vector of ones. By now, we have $I$ repetitions of the $j$ line of point set $P_i$.

Then, to obtain a new matrix with all repeated points of the $P_i$ matrix, again we use the kronecker product, only this time a block matrix is generated that has zeros in all positions except the one corresponding to the $i = j$ product block. This yields $rank(L_i)$ matrices that have no overlapping values. So the matrix obtained has $I \times rank(L_i)$ rows grouped in $I$ smaller blocks, each with a line repeated $I$ times.

Then, this matrix is repeated $O$ times by rows, yielding a final matrix that has a total of $I \times O \times rank(L_i)$ rows, the total of possible combinations that exist for a set of points.

For each $P_i^*$ matrix there is a unique combination of line-wise repetition and block-wise repetition, so each is therefore unique. If we choose the same line in all $P_i^*$ matrices in a set we will obtain a unique set of points, as it was desired. This formula is therefore good for the calculation of combinations of reflections.

As this is a rather dense set of expressions, this is perhaps best understood in an example. Consider two matrices $P_1$ and $P_2$ each with two rows and three coordinates.

$$P_1 = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{bmatrix}$$

If $J = 2$, the $perm$ function yields the following matrices:

$$i = 1, I = 1, O = 2$$

$$P_1^* = perm(P_1) = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_2 & y_2 & z_2 \end{bmatrix}$$

$$i = 2, I = 2, O = 1$$

$$P_2^* = perm(P_2) = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{bmatrix}$$

As you can see, the same line of each matrix is a unique ordered combination of coordinates and all lines have all possible combinations of coordinates, similar to a Cartesian Product of sets.

**Proof of equations 6.1, 6.2 and 6.3** When a sensor is attached to a joint, if that joint has a given angular motion, that motion will alter its angular relation to the horizontal plane. As this is the reference used for all the reference frames in this model, the sensor reading must have this change corrected, so that it is referred to the original horizontal plane. This is very straightforward, as the final angular position of a joint is given by the sum of the sensor reading and all previous increments of the angle in that dimension. Therefore, for example, in a two joint system, if the first joint moves according to the angle $\theta_1$ and the next joint attached to it moves according to the angle $\theta_2$, the readings on the sensor for the second angle will read the value referring to the first. Since what we want is the angle referred to the horizontal, first we need to remove the influence of the first angle. Thanks to the linearity of angles, we get $\theta_f = \theta_2 + \theta_1$, assuming that each sensor already outputs angles referred to the fixed reference frame. If it doesn't, offsets should be removed, yielding $\theta_f = (\theta_2 + \theta_{2_{offset}}) + (\theta_1 + \theta_{1_{offset}})$.

This can be induced as we did before for kinematics, and, if we consider that $\Theta$ is a column vector with all angles and $\Theta_{offset}$ is a vector with the corresponding offset angles, the following equation is obtained, proving equation 6.1.

$$\Theta_c = \mathcal{C} \times \Theta_s + \Theta_{offset}$$

The $\mathcal{C}$ matrix is very similar to the structure matrix, as best exemplified by the case when a single series block is present. This proves equation 6.2 and its inverse equation 6.3.

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

$$\mathcal{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix}$$

## A.2 Workspace coverage and uniqueness of $L \times \mathcal{R} \times B$coordinates example proof

The proof given is one of many that should be done whenever BMs and RFMs are set up together. The case presented is a simple two-dimensional point including attitude.

Consider the following BM and RFM:

$$B = \begin{bmatrix} \sqrt{s_\theta R} & 0 \\ 0 & \sqrt{(1-s_\theta)R} \end{bmatrix} \oplus \begin{bmatrix} S^{-1}(s_\theta) \\ \pi \end{bmatrix}$$

$$\mathcal{R} = \left[ \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \\ 1 & -1 \end{bmatrix} \middle| \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 1 & 1 \\ -1 & 0 \end{bmatrix} \right]$$

.

A point $P = L \times \mathcal{R} \times B$ covers a full workspace if and only if its coordinates can have all values of domain $\begin{bmatrix} \mathbb{R} & \mathbb{R} & 2\pi \end{bmatrix}$.

Let $s_\theta \in [0, 1]$ and $R \in \mathbb{R}_0^+$. All possible points covered can be obtained by setting $L = I_4$ yielding:

$$P = I_8 \times \mathcal{R} \times B$$

$$P = \begin{bmatrix} \sqrt{s_\theta R} & \sqrt{(1-s_\theta)R} & S^{-1}(s_\theta) \\ -\sqrt{s_\theta R} & \sqrt{(1-s_\theta)R} & -S^{-1}(s_\theta) + \pi \\ -\sqrt{s_\theta R} & -\sqrt{(1-s_\theta)R} & -S^{-1}(s_\theta) \\ \sqrt{s_\theta R} & -\sqrt{(1-s_\theta)R} & S^{-1}(s_\theta) + \pi \end{bmatrix}$$

$$\in \begin{bmatrix} \mathbb{R}_0^+ & \mathbb{R}_0^+ & \left[0, \frac{\pi}{2}\right] \\ \mathbb{R}_0^- & \mathbb{R}_0^+ & \left[\frac{\pi}{2}, \pi\right] \\ \mathbb{R}_0^- & \mathbb{R}_0^- & \left[\pi, \frac{3\pi}{2}\right] \\ \mathbb{R}_0^+ & \mathbb{R}_0^- & \left[\frac{3\pi}{2}, 0\right] \end{bmatrix}.$$

By calculating the union of each of the domains the global domain covered can be obtained:

$$\left[\begin{array}{ccc} \mathbb{R}_0^+ & \mathbb{R}_0^+ & \left[0, \frac{\pi}{2}\right] \end{array}\right]$$

$$\cup \left[\begin{array}{ccc} \mathbb{R}_0^- & \mathbb{R}_0^+ & \left[\frac{\pi}{2}, \pi\right] \end{array}\right]$$

$$\cup \left[\begin{array}{ccc} \mathbb{R}_0^- & \mathbb{R}_0^- & \left[\pi, \frac{3\pi}{2}\right] \end{array}\right]$$

$$\cup \left[\begin{array}{ccc} \mathbb{R}_0^+ & \mathbb{R}_0^- & \left[\frac{3\pi}{2}, 2\pi\right] \end{array}\right]$$

$$\rightsquigarrow \left[\begin{array}{ccc} \mathbb{R} & \mathbb{R} & [0, 2\pi] \end{array}\right].$$

Therefore proving that the whole workspace of a reference frame is covered. By applying the $perm$ function to the results above, the workspace of multiple joints is also fully covered.

Proofs for further dimensions can be obtained exactly the same way.

**Explanation of uniqueness of $L \times \mathcal{R} \times B$ coordinates** The previous proof also demonstrates that each line of the $L \times \mathcal{R} \times B$ block covers a different section of the workspace therefore all points obtained using $L \times \mathcal{R} \times B$ are unique except for the null lines. In that case, the user should establish a criteria for uniqueness in null lines. When the null lines overlapping issue is resolved, the coordinates become unique in all workspace.

# Appendix B

# The RTRTk toolkit


Even though it was not required by the work plan, a simple toolbox was developed to demonstrate this model, the RTRTk (Rational Trigonometry Robotics Toolkit). The UML model of the application is provided in figure B.1. The current toolkit is just a set of MatLab™ functions but, since the model itself is very simple, it should be easy to obtain other implementations for other platforms and programming languages. All benchmarks were done with this toolkit.

All development is kept online and the development model is Open Source, so it is open to contributions.

The URL is `http://web.ist.utl.pt/ist152027/content/RTRTk/`.

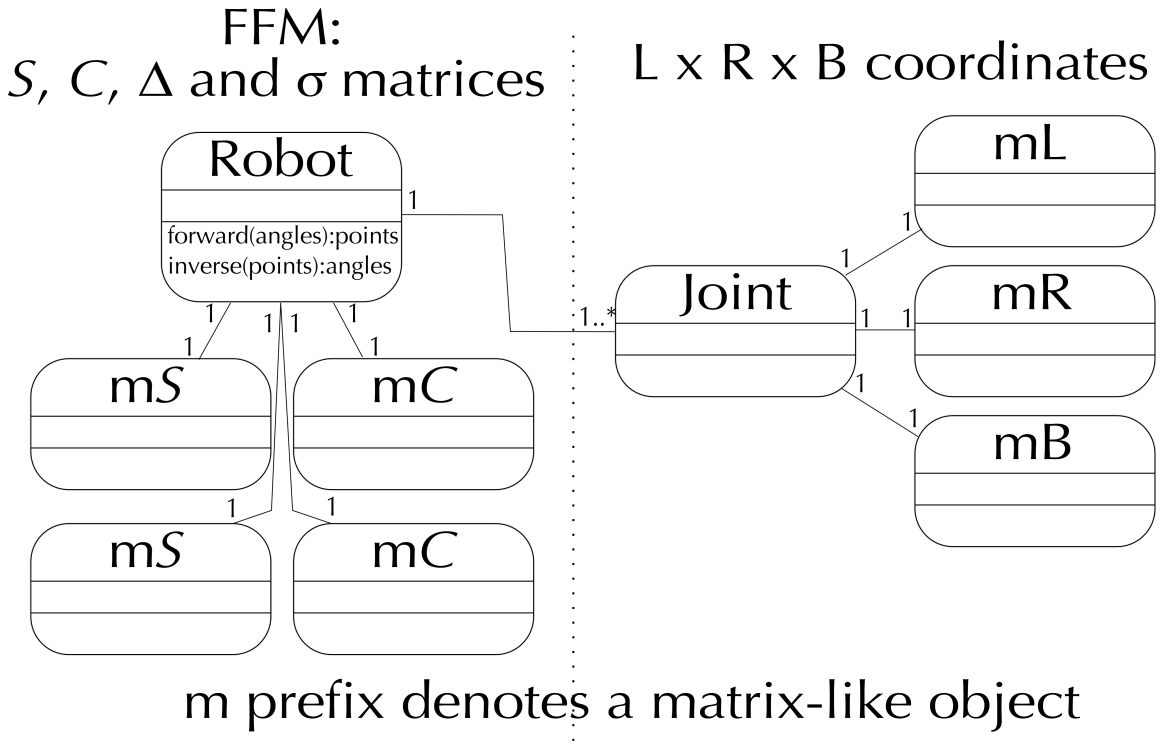


Figure B.1: UML model of the RTRTk toolkit