

# Rational Trigonometry Applied to Robotics

## Robot Kinematic Modeling using Rational Trigonometry

João Pequito Almeida

6 de Novembro de 2007

# Overview

- 1 Overview
- 2 Point Description using Rational Trigonometry
- 3 The Fixed Frames Model for Robot Kinematics
- 4 Conclusions
- 4 Perspectives and Future Work
- 5 Q&A

# Definition

## Definition

Rational Trigonometry is the study of triangles using *spreads* and *quadrances*.

- It is a formal framework for geometrical analysis;
- It revisits old concepts in a new way.

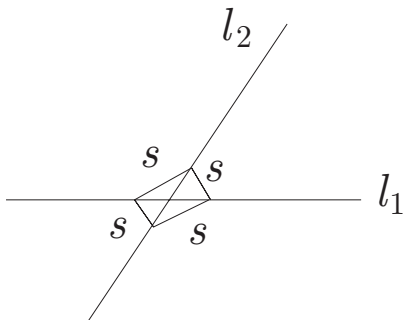
# Quadrance

## Definition

$$Q(A, B) \equiv (B_1 - A_1)^2 + (B_2 - A_2)^2 + \dots + (B_n - A_n)^2 \quad (1)$$

- A simpler concept than distance as it avoids the square root function;
- A metric for separation of points;
- Cannot be added like distances can (i.e.,  $Q(A, C) \neq Q(A, B) + Q(B, C)$ ).

# Spread



## Definition

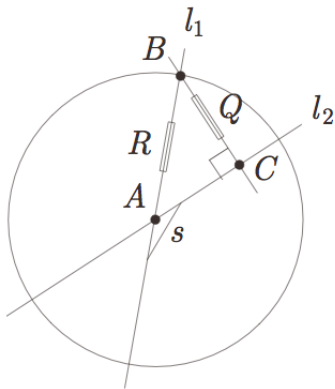
$$l_1 \equiv a_1x + b_1y + c_1 = 0$$

$$l_2 \equiv a_2x + b_2y + c_2 = 0$$

2.

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

## Spread, Cross and Twist



Definition

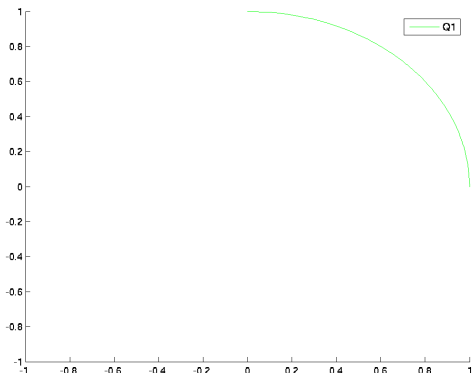
$$S(l_1, l_2) \equiv \frac{Q(B, C)}{Q(A, B)} = \frac{Q}{R} \quad (3)$$

$$C(l_1, l_2) \equiv 1 - S(l_1, l_2) \quad (4)$$

$$T(l_1, l_2) \equiv \frac{S(l_1, l_2)}{C(l_1, l_2)} \quad (5)$$

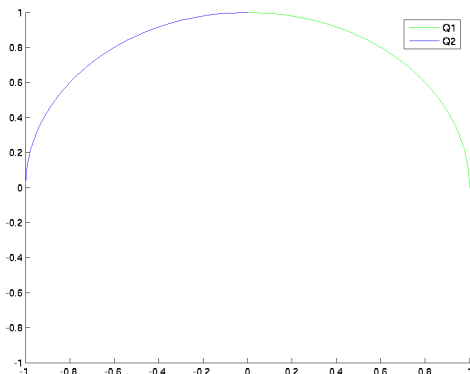
- Unambiguous measure for line separation;
- Simpler expressions than those obtained with classical trigonometry.

# Drawing a circle



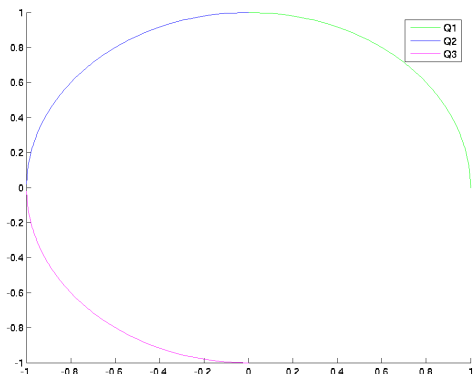
$$\begin{cases} x = \sqrt{1 - S} \\ y = \sqrt{S} \end{cases}$$

# Drawing a circle



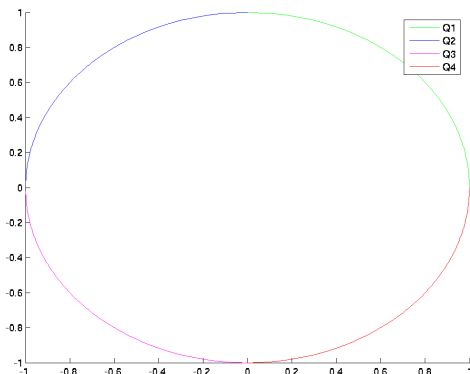
$$\begin{cases} x = -\sqrt{1-S} \\ y = \sqrt{S} \end{cases}$$

# Drawing a circle



$$\begin{cases} x = -\sqrt{1-S} \\ y = -\sqrt{S} \end{cases}$$

# Drawing a circle



$$\begin{cases} x = \sqrt{1-S} \\ y = -\sqrt{S} \end{cases}$$

## Drawing a circle

### Problem

- *In order to draw a circle, we needed 4 reflections of the same values;*
- *These reflections depend on the reference frames used.*

### Solution

Use an expansion matrix to represent the reference frame and the consequent reflections: The Reference Frame Matrix.

# Reference Frame Matrices

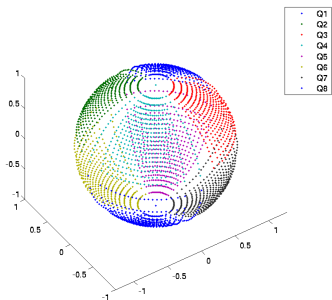
Reference Frame Matrices allow the expansion of values common in all sections of space. They also provide a representation of the reference frame used in a problem.

Example

$$\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} \sqrt{1-S} & 0 \\ 0 & \sqrt{S} \end{bmatrix}$$

The Cartesian Reference Frame

## 3D Example: A Sphere



$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

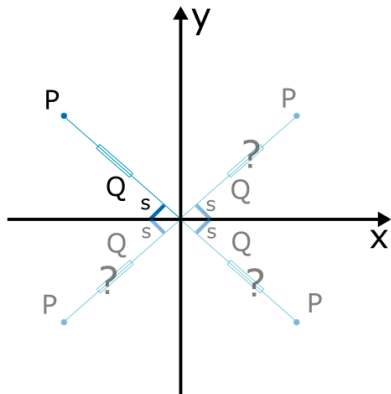
$$\begin{bmatrix} \sqrt{(1-S_\theta)S_\varphi} & 0 & 0 \\ 0 & \sqrt{S_\theta S_\varphi} & 0 \\ 0 & 0 & \sqrt{(1-S_\varphi)} \end{bmatrix}$$

## Representing Attitude

Using conventional representations for attitude (e.g. Euler angles) there are infinite angles for a given attitude whereas using spreads they are finite. Again, since spread only covers part of the sphere, one can use the same reasoning and expand angles like position:

$$\begin{aligned}
 & [\alpha \quad \beta \quad \gamma] = \\
 & = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ -1 & \frac{1}{2} & -1 & 0 & 1 & 0 \\ 1 & -\frac{1}{2} & -1 & 0 & -1 & \frac{1}{2} \\ -1 & 0 & 1 & 0 & -1 & \frac{1}{2} \\ 1 & 0 & -1 & \frac{1}{2} & -1 & 0 \\ -1 & \frac{1}{2} & 1 & -\frac{1}{2} & -1 & 0 \\ 1 & -\frac{1}{2} & 1 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -1 & 0 & -1 & \frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix} \times \begin{bmatrix} S^{-1}(s_\alpha) & 0 & 0 \\ 1 & 0 & 0 \\ 0 & S^{-1}(s_\beta) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & S^{-1}(s_\gamma) \\ 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

# Point Addressing



As single values of *spread* expand to multiple values of position, addressing is required for unique values. Since these expanded values are obtained using Reference Frame Matrices, one can simply use a line selecting matrix:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \times R$$

## The Joint Equation

If we use a Base Matrix with the values to be expanded, the following definition emerges:

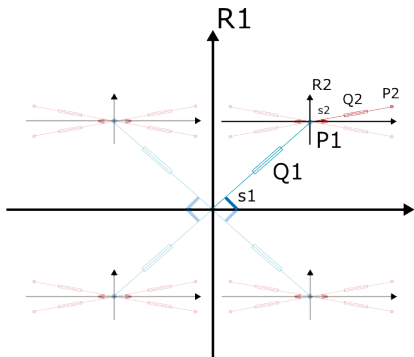
### Definition

An actuator point  $P$  in a generic Reference Frame  $R$  can be calculated by its polar components in a base matrix  $B$  using:

$$P = L \times R \times B$$

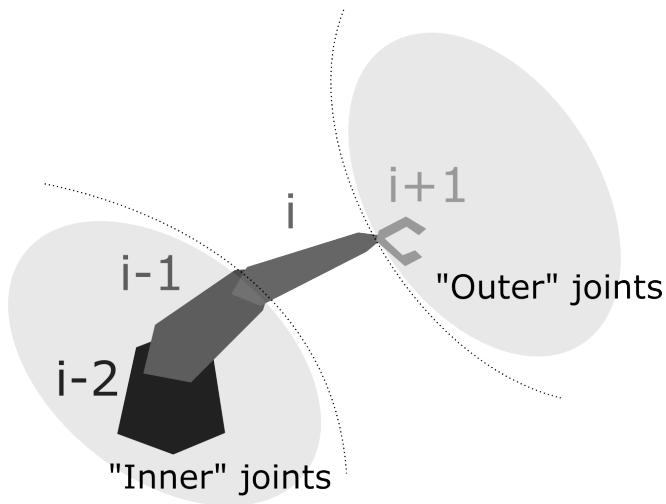
Where  $L$  is a line selecting matrix.

# Combining sets of points



Using the  $L \times R \times B$  notation, one can access any reflection of any reference frame. By combining each reflection with every other, all workspace is covered. How to obtain these combinations?

# Notation for robotic structures



The operation is similar to the Cartesian Product of Sets, but since each set is a matrix and each element is a line of that matrix, using some algebra tricks one can use the following formula on the points  $P_i = L \times R \times B$ :

### Definition

$$M_j = [\delta_{1j} \delta_{2j} \dots \delta_{\text{rank}(L_i), j}], \quad \delta_{ij} = 1 \text{ if } i = j, \quad 0 \text{ otherwise}$$

$$\text{Sol}(a, b) = \prod_{i=a}^b \text{rank}(L_i);$$

$$I = \text{Sol}(1, i - 1); \quad O = \text{Sol}(i + 1, n);$$

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

# $M_j$ Explained

## Definition

$$M_j = [\delta_{1j} \delta_{2j} \dots \delta_{rank(L_i)j}], \quad \delta_{ij} = 1 \text{ if } i = j, \quad 0 \text{ otherwise}$$

$M_j$  is a line matrix with a one (1) on the  $j$ th entry. This matrix is used to access each line of an  $L \times R \times B$  point.

$\delta_{ij}$  denotes the Kronecker Delta.

# $Sol(a, b)$ Explained

## Definition

$$Sol(a, b) = \prod_{i=a}^b rank(L_i);$$

The  $Sol$  function simply counts the number of possible combinations in a set from the element indexed by  $a$  to the element indexed by  $b$ . Since the set is a matrix, it's rank provides the number of tuples present (note that  $rank(L_i) = rank(P_i)$ ).

# $I$ and $O$ Explained

## Definition

$$I = \text{Sol}(1, i - 1); \quad O = \text{Sol}(i + 1, n);$$

Given a joint at index  $i$ ,  $I$  has the inner possible solutions (solutions toward index  $i = 0$ ) and  $O$  has the outer possible solutions.

## $P_i^*$ Explained

### Definition

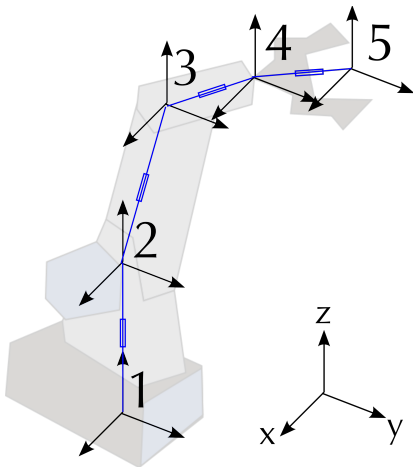
$$P_i^* = \left( \sum_{j=1}^{\text{rank}(L_i)} (M_j \times P_i \otimes \text{ones}(I, 1) \otimes M_j^T) \right) \otimes \text{ones}(O, 1)$$

This is a cooked up expression divided into two main blocks: the first, involving the inner combinations ( $I$ ), repeats each tuple in consecutive lines  $I$  times.

The second, involving the outer combinations ( $O$ ), repeats each block of tuples  $O$  times.

Note that if  $O = I = 1$ ,  $P_i^* = P_i$ , this being the most common situation.

# The Fixed Frames Model for Robot Kinematics



Using the proposed framework transformation matrices are very complex so a new approach was tested:

Reference frames are **kept fixed** with equal attitude referring to the workspace reference frame.

# Representing Robotic Structures using FFM

So far, all expressions were valid for both Quadrances and Distances. But as we apply this model to the real world, using Distance is again more useful.

By keeping Reference Frames with equal attitude, the effect of their motion can be calculated using simple addition since all axis are parallel.

# Processing sensor data

If reference frames do not rotate and sensors do, a correction must be applied to the data from and to the actual robot:

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

Typically, a  $\mathcal{C}$  matrix will have a very specific structure, e.g.,

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

# Series Manipulators

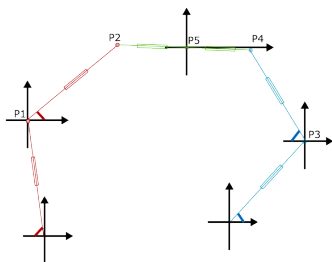
A Series manipulator

$$\begin{aligned}
 P_{f_1} &= P_1^* \\
 P_{f_2} &= P_1^* + P_2^* \\
 &\dots \\
 P_{f_n} &= P_1^* + P_2^* + \dots + P_J^*
 \end{aligned}
 \Rightarrow$$

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



# Complex Examples



$$\begin{bmatrix} P_{f1} \\ P_{f2} \\ P_{f3} \\ P_{f4} \\ P_{f5} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \otimes I_{Sol(1,n)} \times \begin{bmatrix} P1 \\ P2 \\ P3 \\ P4 \end{bmatrix}$$

# The Structure Equation

As shown in the previous examples, a pattern emerges whenever one tries to solve a particular structure, always involving a matrix with a very similar structure.

## Definition

A Structure Matrix  $\mathcal{S}$  relates single joint points  $P_i$  to each other according to a Robot's Structure to obtain workspace points  $P_{f_i}$ .

$$P_{f_i} = \mathcal{S} \times P_i$$

# 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.

# Constraint analysis equations

The following equations were obtained to describe each of the constraints:

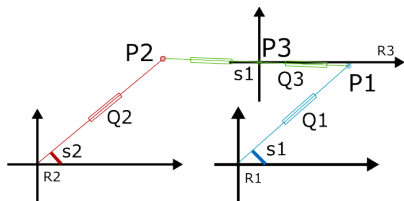
$$\left\{ \begin{array}{l} \forall P \in P_f, P \in W \\ \forall i \in [1, n] \theta_i \in A_i \\ D \bullet D \times \sigma = K \bullet K \end{array} \right. \quad (6)$$

$$D = \Delta \times P_f \quad (7)$$

## Application example

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

$$D = \Delta \times \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$$



$$\sigma = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \Delta = [ 1 \quad -1 ], K = [ R_3 ]$$

$$\rightsquigarrow (P_1 - P_2) \bullet (P_1 - P_2) \times \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = [ R_3 ] \bullet [ R_3 ]$$

# Compact Form

This method can be summed up in 4 main equations:

- The Joint Equation and *perm* function

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

- The Structure Equation  $P_f = \mathcal{S} \times P^*$

- The Sensor-Actuator Equation  $\Theta_c = \mathcal{C} \times \Theta_s + \Theta_{\text{offset}}$

- The Constraint Equations  $\left\{ \begin{array}{l} \forall P \in P_f P \in W \\ \forall_{i \in [1, n]} \theta_i \in A_i \\ D \bullet D \times \sigma = K \bullet K \end{array} \right.$

# Conclusions

This framework provides:

- A straightforward way to approach any robotic structures in a generic way;
- An algebraic representation that allows a global perspective of the problems at hand (e.g. the existence of redundant solutions).

# Perspectives and Future Work

## Perspectives:

- It's clear that using this framework the inversion of cinematic models might be simplified for many cases and current work involves optimizing this inversion in a global way;
- Computational versions are easy to obtain and implement.

## Future Work

- Current plans include solving the inverse model in an optimal way (hopefully globally optimal)
- Implement a working toolbox for MatLab/Octave and/or a library for C/C++/Java
- Developments available online at <http://web.ist.utl.pt/ist152027/content/tfc/>

## Q&amp;A

?

<http://web.ist.utl.pt/ist152027/content/tfc/>

[jpal@mega.ist.utl.pt](mailto:jpal@mega.ist.utl.pt)

<3