Robot Kinematics

Robotica
for Computer Engineering students

A.A. 2007/2008
Study of Motion

- **Kinematics** studies the relation between the independent variables of the joints and the Cartesian positions reached by the robot.

- **Dynamics** studies the equations that characterize the robot motion (speed and acceleration).

- **Trajectories computation** consists of determining a way to provide a robot for the sequence of points (or joint variables) to move from one point to another (kinematics) with suitable speeds and accelerations (dynamics).

- **Control** aims at performing trajectories that are similar to those computed.
Two types of kinematics:

- **Forward kinematics (angles to position):**
  - what are you given:
    - the length of each link
    - the angle of each joint
  - what you can find:
    - the position of any point (i.e., its (x,y,z) coordinates)

- **Inverse Kinematics (position to angle):**
  - what are you given:
    - the length of each link
    - the position of some point on the robot
  - what you can find:
    - the angles of each joint needed to obtain that position
Forward Kinematics: Example with Planar RR

**Situation**
- you have a robotic arm that starts aligned with the $x_0$-axis
- you tell the first link to move by $\theta_1$ and the second link to move by $\theta_2$

**The quest:**
- what is the position of the end of the robotic arm?

**Solution:**
- Geometric approach
  - easier in simple situations
- Algebraic approach
  - involves coordinate transformations
Forward Kinematics: Example with 3 link arm

Situation
- you have a 3 link arm that starts aligned with the x-axis
- \(l_1, l_2, l_3\) are the lengths of the 3 links
- The three links are moved respectively by \(\theta_1, \theta_2, \theta_3\)

The quest
- Find the Homogeneous matrix to get the position of the yellow dot in the \(X^0Y^0\) frame

Solution
\[ H = R_z(\theta_1) \times T_{x1}(l_1) \times R_z(\theta_2) \times T_{x2}(l_2) \times R_z(\theta_3) \]
- multiplying \(H\) by the position vector of the yellow dot in the \(X^3Y^3\) frame gives its coordinates relative to the \(X^0Y^0\) frame
Forward Kinematics

With more than 3 joints and with kinematic chains that do not lay on the plane the geometric method is too difficult.

We need a systematic method:

- represent the open kinematic chain with the same formalism
- find algebraic solutions
  - using homogeneous coordinates
- build references using a quasi-algorithmic procedure
  - Denavit-Hartenberg Notation
Each joint is assigned a coordination frame.

Using the Denavit-Hartenberg notation, you need 4 parameters to describe how a frame \((i)\) relates to a previous frame \((i-1)\).

To align two axes we need 4 parameters: \(\alpha, a, d, \theta\).
1) $a_i$

- **Technical definition:** is the length of the perpendicular between the joint axes. The joint axes is the axes around which revolution takes place which are the $Z_{(i-1)}$ and $Z_i$ axes. These two axes can be viewed as lines in space. The common perpendicular is the shortest line between the two axis-lines and is perpendicular to both axis-lines.
1) \(a_i\)

**Visual approach:** “A way to visualize the link parameter \(a_i\) is to imagine an expanding cylinder whose axis is the \(Z_{(i-1)}\) axis - when the cylinder just touches the joint axis \(i\) the radius of the cylinder is equal to \(a_i\)” (Manipulator Kinematics)
1) \( a_i \)

- **It’s Usually on the Diagram Approach:** If the diagram already specifies the various coordinate frames, then the common perpendicular is usually the \( X_{(i-1)} \) axis. So \( a_i \) is just the displacement along the \( X_{(i-1)} \) to move from the \( i-1 \) frame to the \( i \) frame.

- If the link is prismatic, then \( a_i \) is a variable
The Parameters

1) $\alpha_i$

**Technical Definition:** Amount of rotation around the common perpendicular so that the joint axes are parallel.

i.e. How much you have to rotate around the $X_{(i-1)}$ axis so that the $Z_{(i-1)}$ is pointing in the same direction as the $Z_i$ axis. Positive rotation follows the right hand rule.
1) \( d_i \)

Technical Definition: The displacement along the \( Z_i \) axis needed to align the \( a_{(i-1)} \) common perpendicular to the \( a_i \) common perpendicular.

In other words, displacement along the \( Z_i \) to align the \( X_{(i-1)} \) and \( X_i \) axes.
1) $\theta_i$

**Technical Definition:** Amount of rotation around the $Z_i$ axis needed to align the $X_{(i-1)}$ axis with the $X_i$ axis.
The Denavit-Hartenberg Matrix

Just like the Homogeneous Matrix, the Denavit-Hartenberg Matrix is a transformation matrix from one coordinate frame to the next. Using a series of D-H Matrix multiplications and the D-H Parameter table, the final result is a transformation matrix from some frame to your initial frame.
Algorithm to Fix the Reference Frames

1) The robot is put in resting position. $X_0Y_0Z_0$ is fixed to the robot basis
2) For each pair $i$ of consecutive joints
   1) Fix the axis of joint $i+1$ on $Z_i$
   2) Put the origin of frame $i$ at the intersection of axes $Z_i$ and $Z_{(i-1)}$.
      If the axes do not intersect, then put the origin at the intersection between $Z_i$ and the minimum distance segment between the two axes
   3) Set the $X_i$ axis on the same direction of the minimum distance segment, or on the link direction towards the hand
   4) The $Y_i$ axis is fixed following the right-hand rule
3) Fix the reference frame for the hand
Algorithm to Fix the Reference Frames

4) For each pair i of consecutive reference frames compute the four parameters
   1) Compute $d_i$: distance between axes $X_{(i-1)}$ and $X_i$ along axis $Z_{(i-1)}$ (variable for a prismatic joint)
   2) Compute $a_{(i-1)}$: distance between axes $Z_{(i-1)}$ and $Z_i$ along axis $X_i$ (link length)
   3) Compute $\theta_i$: revolution angle between $X_{(i-1)}$ and $X_i$ around axis $Z_{(i-1)}$ (variable for a revolute joint)
   4) Compute $\alpha_{(i-1)}$: revolution angle between $Z_{(i-1)}$ and $Z_i$ around axis $X_i$ (twist angle)
For what regards the reference frame of the hand:
- The origin is placed between the fingers
- The Y-axis follows the sliding direction of the fingers
- The Z-axis follows the approaching direction, pointing towards the finger opening

For what regards the reference frame of the grasp point:
- To simplify the grasping operations it may be useful to place the object reference frame on the grasping point
- Inverse kinematics is important in this context
Algorithm for Forward Kinematics

1) Put the manipulator in resting position
2) Set the reference frames to joints and links
3) Compute Denavit-Hartenberg parameters
4) Compute transformation matrix $A_i$ that allows to pass from the reference frame of the i-th joint to the one of the (i+1)-th joint
5) Multiply matrices $A_i$ to get matrix $T$ that allows to pass from the reference frame of the base $X_0Y_0Z_0$ to the one of the hand $X_nY_nZ_n$
6) From the matrix $T$ extract the coordinates of the current position
7) Look at the rotation sub-matrix and extract orientation components
The transformations required to pass from reference frame i-1 to reference frame i w.r.t. joint i-1 are:

- Rotate $X_{(i-1)}$ by $\theta_i$ around $Z_{(i-1)}$, to align it with $X_i$: $\text{Rot}(Z_{(i-1)}, \theta_i)$
- Translate the origin of reference frame i-1 by $d_i$ along $Z_{(i-1)}$, to overlap $X_{(i-1)}$ and $X_i$: $\text{Transl}(0,0,d_i)$
- Translate the origin of reference frame i-1 by $a_i$ along $X_i$, to place it at the origin of reference frame i: $\text{Transl}(a_i,0,0)$
- Rotate $Z_{(i-1)}$ around $X_i$ by $\alpha_i$ to overlap the two reference frames i-1 and i: $\text{Rot}(X_i, \alpha_i)$

$A_{i-1,i} = \text{Transl}(0,0,d_i)\text{Rot}(Z_{(i-1)}, \theta_i)\text{Transl}(a_i,0,0)\text{Rot}(X_i, \alpha_i)$
Transformation Matrix $A$

- **Revolute joint**

\[
\begin{bmatrix}
\cos \theta_i & -\cos \alpha_i \sin \theta_i & \sin \alpha_i \sin \theta_i & a_i \cos \theta_i \\
\sin \theta_i & \cos \theta_i \cos \alpha_i & -\sin \alpha_i \cos \theta_i & a_i \sin \theta_i \\
0 & \sin \alpha_i & \cos \alpha_i & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

- **Prismatic joint**

\[
\begin{bmatrix}
\cos \theta_i & -\cos \alpha_i \sin \theta_i & \sin \alpha_i \sin \theta_i & 0 \\
\sin \theta_i & \cos \theta_i \cos \alpha_i & -\sin \alpha_i \cos \theta_i & 0 \\
0 & \sin \alpha_i & \cos \alpha_i & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

- **Transformation Matrix A**
Aim: given a point in the mobile system, we want its coordinates in the fixed system

\[ T = A_{0,6}^0 \cdot A_{1,2}^1 \cdot A_{2,3}^2 \cdot A_{3,4}^3 \cdot A_{4,5}^4 \cdot A_{5,6}^5 \]

\[ P_{i-1} = A_{i-1,i}^{i-1} \cdot P_i \]

\[
T = \begin{bmatrix} x_i & y_i & z_i & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_i & p_i \\ 0 & 1 \end{bmatrix}
\]
**RR Manipulator**

- Workspace: internal area of a sphere
- Initial part of a spherical or a hinged robot

\[
T = A_{0,1}^0 \cdot A_{1,2}^1 = \begin{bmatrix}
\cos \theta_1 & 0 & \sin \theta_1 & 0 \\
\sin \theta_1 & 0 & -\cos \theta_1 & 0 \\
0 & 1 & 0 & d_1 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
\cos \theta_2 & -\sin \theta_2 & 0 & l_2 \cdot \cos \theta_2 \\
\sin \theta_2 & \cos \theta_2 & 0 & l_2 \cdot \sin \theta_2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>Link</th>
<th>$\theta$</th>
<th>$\alpha$</th>
<th>$a$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\theta_1$</td>
<td>$90^\circ$</td>
<td>0</td>
<td>$d_1$</td>
</tr>
<tr>
<td>2</td>
<td>$\theta_2$</td>
<td>$0^\circ$</td>
<td>$l_2$</td>
<td>0</td>
</tr>
</tbody>
</table>
Planar RR Manipulator

- Workspace: circle area
- Possible part of a SCARA robot

$$T = A_{0,1}^0 \cdot A_{1,2}^1 = \begin{bmatrix}
\cos\theta_1 & -\sin\theta_1 & 0 & l_1 \cdot \cos\theta_1 \\
\sin\theta_1 & \cos\theta_1 & 0 & l_1 \cdot \sin\theta_1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} \cdot \begin{bmatrix}
\cos\theta_2 & -\sin\theta_2 & 0 & l_2 \cdot \cos\theta_2 \\
\sin\theta_2 & \cos\theta_2 & 0 & l_2 \cdot \sin\theta_2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}$$

<table>
<thead>
<tr>
<th>Link</th>
<th>$\theta$</th>
<th>$\alpha$</th>
<th>$a$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\theta_1$</td>
<td>0°</td>
<td>$l_1$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$\theta_2$</td>
<td>0°</td>
<td>$l_2$</td>
<td>0</td>
</tr>
</tbody>
</table>
Workspace: circle area
Possible part of a cylindric robot

\[
T = A_{0,1}^0 \cdot A_{1,2}^1 = 
\begin{bmatrix}
\cos \theta_1 & 0 & \sin \theta_1 & 0 \\
n\sin \theta_1 & 0 & -\cos \theta_1 & 0 \\
0 & 1 & 0 & d_1 \\
0 & 0 & 0 & 1
\end{bmatrix} \cdot 
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d_2 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Workspace: round cornered rectangle

\[
T = A_{0,1}^0 \cdot A_{1,2}^1 = \\
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 1 & 0 & d_1 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} \cdot \\
\begin{bmatrix}
\cos\theta_2 & -\sin\theta_2 & 0 & l_2 \cdot \cos\theta_2 \\
\sin\theta_2 & \cos\theta_2 & 0 & l_2 \cdot \sin\theta_2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
TT Manipulator

- Workspace: rectangle
- Possible part of a Cartesian robot

\[
T = A_{0,1}^0 \cdot A_{1,2}^1 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 1 & 0 & d_1 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d_2 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>Link</th>
<th>θ</th>
<th>α</th>
<th>a</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0°</td>
<td>90°</td>
<td>0</td>
<td>d_1</td>
</tr>
<tr>
<td>2</td>
<td>0°</td>
<td>0°</td>
<td>0</td>
<td>d_2</td>
</tr>
</tbody>
</table>
We use the Euler's angles...

\[
R = \text{Rot}(z, \phi) \cdot \text{Rot}(u, \theta) \cdot \text{Rot}(w, \psi) =
\begin{bmatrix}
\cos \phi \cos \psi - \sin \phi \cos \theta \sin \psi & -\cos \phi \sin \psi - \sin \phi \cos \theta \cos \psi & \sin \phi \sin \theta \\
\sin \phi \cos \psi + \cos \phi \cos \theta \sin \psi & -\sin \phi \sin \psi + \cos \phi \cos \theta \cos \psi & -\cos \phi \sin \theta \\
\sin \theta \sin \psi & \sin \theta \cos \psi & \cos \theta
\end{bmatrix}
\]

... and equal them to the orientation sub-matrix:

\[
T = \begin{bmatrix}
x_i & y_i & z_i & p_i \\
0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
R_i & p_i \\
0 & 1
\end{bmatrix}
\]

\[
R_i = \begin{bmatrix}
n_x & o_x & a_x \\
n_y & o_y & a_y \\
n_z & o_z & a_z
\end{bmatrix}
\]
A simple solution can be obtained by solving the set of 9 equations and 3 unknown quantities:

\[
\theta = \cos^{-1}[a_z]
\]

\[
\psi = \cos^{-1}\left[ \frac{o_z}{\sin \theta} \right]
\]

\[
\phi = \cos^{-1}\left[ \frac{-a_y}{\sin \theta} \right]
\]

This solution is useless:

- the accuracy of arccos function depends by the angle since \( \cos(\theta) = \cos(-\theta) \)
- when \( \theta \) approaches \( 0^\circ \) or \( 180^\circ \) the last two equations give inaccurate or indefinite solutions
Hand Orientation: using atan2

A trigonometric function that does not have these problems is the atan2(x,y) function, that computes $\tan^{-1}(y/x)$ placed in the right sub-frame

$$\theta = \text{atan2}(y, x) = \begin{cases} 
0^\circ \leq \theta \leq 90^\circ, & \text{for } +X \wedge +Y \\
90^\circ \leq \theta \leq 180^\circ, & \text{for } -X \wedge +Y \\
-180^\circ \leq \theta \leq -90^\circ, & \text{for } -X \wedge -Y \\
-90^\circ \leq \theta \leq 0^\circ, & \text{for } +X \wedge -Y 
\end{cases}$$
The Paul method computes the hand orientation using the atan2 function.

The method consists of **pre-multiplying** both ends of equation by the inverse of one of the rotation matrices.

\[ R_z^{-1} \cdot \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = R_u,\theta \cdot R_w,\psi \]

Now \( \Phi \) is on the left side:

\[ \phi = \tan^{-1} \left( \frac{a_x}{-a_y} \right) = \text{atan2}(a_x, -a_y) \]

\[ \psi = \tan^{-1} \left( \frac{\sin \psi}{\cos \psi} \right) = \tan^{-1} \left( \frac{-o_x \cos \phi - o_y \sin \phi}{n_x \cos \phi + n_y \sin \phi} \right) = \text{atan2}(-o_x \cos \phi - o_y \sin \phi, n_x \cos \phi + n_y \sin \phi) \]

\[ \theta = \tan^{-1} \left( \frac{\sin \theta}{\cos \theta} \right) = \tan^{-1} \left( \frac{a_x \sin \phi - a_y \cos \phi}{a_z} \right) = \text{atan2}(a_x \sin \phi - a_y \cos \phi, a_z) \]
The same method can be applied through post-multiplying both ends of equation by the inverse of one of the rotation matrices.

\[
\begin{bmatrix}
  n_x & o_x & a_x \\
  n_y & o_y & a_y \\
  n_z & o_z & a_z
\end{bmatrix} \cdot R_{w,\psi}^{-1} = R_{z,\phi} \cdot R_{u,\theta}
\]

\[
\psi = \tan^{-1} \left( \frac{n_z}{o_z} \right) = \text{atan2} \left( n_z, o_z \right)
\]

\[
\phi = \tan^{-1} \left( \frac{\sin \phi}{\cos \phi} \right) = \tan^{-1} \left( \frac{n_y \cos \psi + o_y \sin \psi}{n_x \cos \psi + o_x \sin \psi} \right) = \text{atan2} \left( n_y \cos \psi + o_y \sin \psi, n_x \cos \psi + o_x \sin \psi \right)
\]

\[
\theta = \tan^{-1} \left( \frac{\sin \theta}{\cos \theta} \right) = \tan^{-1} \left( \frac{n_z \sin \psi + o_z \cos \psi}{a_z} \right) = \text{atan2} \left( n_z \sin \psi + o_z \cos \psi, a_z \right)
\]

We do not know when it is better to use the pre or the post-method.
The forward kinematic problem has always one and only one solution, that can be obtained through the computation of transformation matrix $T$.

Since $T$ is obtained through the product of 6 matrices, for computational reasons use only two matrices:
- one for the 3 DOF of the arm
- one for the 3 DOF of the hand

Nevertheless there are still some real-time problems to compute the Cartesian coordinates:
- Often joint variables are used
Inverse Kinematics: Problem Formulation

Given a position and an orientation in the Cartesian space, inverse kinematics aims at finding a configuration of joints that allows to reach them

- Solution existence
- Solution uniqueness
- Solution methods
- Real-time solutions

\[
\theta = \arctan \left( \frac{y}{x} \right)
\]

\[
S = \sqrt{x^2 + y^2}
\]
Inverse Kinematics: the Solution

Unfortunately, the transformation of position from Cartesian to joint coordinates generally does not have a closed-form solution.

The solution can be obtained by solving the system obtained by setting the T matrix of the hand (that contains the Cartesian coordinates) equal to its symbolic expression (that contains the joint coordinates)

- 12 equations and 6 unknowns
- only 3 of the 9 rotation terms are independent
- so we have 6 non-linear, transcendental equations with 6 unknowns

\[
\begin{bmatrix}
    x_y & y_y & z_y & p_x \\
    x_z & y_z & z_z & p_x \\
    0 & 0 & 0 & 1
\end{bmatrix}
= \begin{bmatrix}
    n_x & o_x & a_x & d_x \\
    n_y & o_y & a_y & d_x \\
    n_z & o_z & a_z & d_x \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]
Inverse Kinematics: Solution Existence

- If the point to be reached falls into the workspace and the robot has 6 DOF then a solution exists.
- If the robot has less than 6 DOF it has to be verified that the point can be reached.
- It is hard to determine whether a point falls or not inside the workspace.
- Things are even harder when considering the dexterous space.
- The solution does not exist when:
  - the goal point is outside the workspace.
  - the goal point is inside the workspace but there are physical constraints.
  - the goal point must be reached following a trajectory that requires infinite acceleration.
What makes Inverse Kinematics a hard problem?

- Redundancy: a unique solution to this problem does not exist.
- The number of possible solutions increases with the number of DOF.
- The number of solutions depends by the number of Denavit-Hartenberg non-null parameters.
  - For a 6R manipulator we may have at most 16 solutions.

We are interested in all the possible solutions, so we need a criterion to select the best solution:

- The “closest” to the current configuration.
- Move outermost links the most.
- Energy minimization.
- Minimum time.
Several solution methods have been proposed:

- **Closed form solutions**
  - Geometrical methods
    - reduce the larger problem to a series of plane geometry problems
  - Algebraic methods
    - trigonometric equations

- **Iterative (numerical) solutions**
Geometrical Method: An Example

Using the law of cosines:

\[(x^2 + y^2) = l_1^2 + l_2^2 - 2l_1l_2 \cos(\pi - \theta_2)\]

\[\theta_2 = \arccos\left(\frac{(x^2 + y^2) - l_1^2 - l_2^2}{2l_1l_2}\right): \text{REDUNDANT}\]

Using the law of sines:

\[\frac{\sin(\theta_1 - \alpha)}{l_2} = \frac{\sin(\pi - \theta_2)}{\sqrt{x^2 + y^2}} = \frac{\sin(\theta_2)}{\sqrt{x^2 + y^2}}\]

\[\theta_1 = \arcsin\left(\frac{l_2 \sin(\theta_2)}{\sqrt{x^2 + y^2}}\right) + \text{atan2}(y, x): \text{REDUNDANCY caused by the two values of } \theta_2\]
We already saw the kinematics equations:

\[
\begin{align*}
x &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\
y &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \\
x^2 + y^2 &= l_1^2 + l_2^2 + 2l_1l_2(\cos(\theta_1 + \theta_2)\cos(\theta_1) + \sin(\theta_1 + \theta_2)\sin(\theta_1)) \\
x^2 + y^2 &= l_1^2 + l_2^2 + 2l_1l_2\cos(\theta_2)
\end{align*}
\]

\[
\begin{align*}
\theta_2 &= \arccos\left(\frac{(x^2 + y^2) - l_1^2 - l_2^2}{2l_1l_2}\right): \text{REDUNDANT}
\end{align*}
\]

\[
\begin{align*}
x &= l_1 \cos(\theta_1) + l_2(\cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2)) \\
y &= l_1 \sin(\theta_1) + l_2(\sin(\theta_1)\cos(\theta_2) + \cos(\theta_1)\sin(\theta_2)) \\
\theta_1 &= \arcsin\left(\frac{y(l_2\cos(\theta_2) + l_1) - x l_2 \sin(\theta_2)}{(l_1 + l_2\cos(\theta_2))^2 + (l_2\sin(\theta_2))^2}\right)
\end{align*}
\]
Inverse Kinematics: Closed Form Solutions

- An arbitrary 6 DOF manipulator cannot be solved using a closed form solution
- Pieper's method: solution of a 4\textsuperscript{th} order polynomial
- Paul's method: pre- and post-multiplies
- Other methods
  - helicoidal algebra
  - double matrices
  - quaternions
  - algebraic approaches based on substitutions
    - \( u = \tan(\theta/2) \)
    - \( \cos(\theta) = (1-u^2)/(1+u^2) \)
    - \( \sin(\theta) = 2u/(1+u^2) \)
Inverse Kinematics: Pieper's Solution

Suitable for manipulators with 6 DOF in which 3 consecutive axes intersect at a point (including robots with 3 consecutive parallel axes, since they met at a point at infinity)

Pieper's method applies to the majority of commercially available industrial robots (Puma 560, SCARA)

The Pieper's solution computes separately the first 3 and the last 3 joints:

- Locate the intersection of the last 3 joint axes
- Solve IK for first 3 joints
- Compute $T_{0,3}$ and determine $T_{3,6}$ as $T_{0,3}T_{0,6}$
- Solve IK for last three joints
Set the known Cartesian matrix $T$ equal to the manipulator matrix (containing the joint unknowns)

Search the second matrix for:
- Elements with only one joint unknown
- Pair of elements that lead to an expression with only one unknown once divided by each other
- Elements that can be simplified

Solve the equations involving the selected elements, thus finding joint unknowns as function of known elements belonging to the $T$ matrix

If no element has been identified at step 2, pre-multiply both terms for the inverse of the transformation matrix of the first link (and iteratively for the following links)
- as an alternative it is possible to post-multiply for the inverse of the transformation matrix of the last link
Example of Paul's method: RR

\[
T = \begin{bmatrix}
\cos\theta_1 \cos\theta_2 & -\cos\theta_1 \sin\theta_2 & \sin\theta_1 & l_2 \cos\theta_1 \cos\theta_2 \\
\sin\theta_1 \cos\theta_2 & -\sin\theta_1 \sin\theta_2 & -\cos\theta_1 & l_2 \sin\theta_1 \cos\theta_2 \\
\sin\theta_2 & \cos\theta_2 & 0 & d_1 + l_2 \sin\theta_2 \\
0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
n_x & o_x & a_x & p_x \\
n_y & o_y & a_y & p_y \\
n_z & o_z & a_z & p_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Select \( p_x \) e \( p_y \)

\[
\frac{p_y}{p_x} = \frac{l_2 \cdot \sin(\theta_1) \cdot \cos(\theta_2)}{l_2 \cdot \cos(\theta_1) \cdot \cos(\theta_2)} \Rightarrow \theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right)
\]

Pre-multiply by \( (A_{0,1})^{-1} \)

\[
\frac{p_x \cdot \cos(\theta_1) + p_y \cdot \sin(\theta_1)}{p_z - d_1} = \frac{l_2 \cdot \cos(\theta_2)}{l_2 \cdot \sin(\theta_2)} \Rightarrow \theta_2 = \tan^{-1}\left(\frac{p_z - d_1}{p_x \cdot \cos(\theta_1) + p_y \cdot \sin(\theta_1)}\right)
\]
Example of Paul's method: TT

\[
T = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & -1 & -d_2 \\
0 & 1 & 0 & d_1 \\
0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
n_x & o_x & a_x & p_x \\
n_y & o_y & a_y & p_y \\
n_z & o_z & a_z & p_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

- Select \( p_y \)
  \[ d_2 = -p_y \]

- Select \( p_z \)
  \[ d_1 = p_z \]
For robots with coupled geometries the closed form solution may not exist => iterative solutions

- m equations with n unknowns
- it starts with an initial estimation
- compute T matrix and error w.r.t. Cartesian values
- update the estimate to reduce the error
- indefinite execution time to get a definite error or indefinite error to get a definite execution time
- iterative methods may fail to find all the solutions to IK

While in the '80s these methods were not feasible, nowadays they represent an alternative.

Nevertheless, industrial robots are typically built to meet one of the Pieper's sufficient conditions in order to use closed form solutions
Redundancies and Degenerations

- A manipulator is redundant if it can reach the same position with several different configurations.
- Manipulators with more than 6 DOF are infinitely redundant.
- A point reachable with infinite configurations is called a degeneration point.
- Physical constraints are imposed to avoid degeneration points.
Static Precision

- **Accuracy**
  - difference between desired position and actually reached position
  - less tolerance => higher accuracy

- **Repeatability**
  - the variance of the reached position while repeating the same command
  - important when the robot is programmed on the field

- **Spatial resolution**
  - the minimum distance that can be measured or commanded
  - it depends by the resolution of the internal sensors
Manipulator Performances

- **Maximum payload**
  - the *maximum weight* that can be carried by a robot at low speed while keeping the accuracy
  - the *nominal payload* is measured at maximum speed while keeping the accuracy
  - the payload is any weight put on the wrist

- **Maximum speed**
  - the maximum velocity at which the robot end can be moved while extended

- **Cycle time**
  - The cycle time is the time required to execute a standard cycle of pick-and-place of 12 inches
  - For industrial robots ~ 1s