Robot Kinematics

Simon Leonard
Department of Computer Science
Johns Hopkins University
Robot Manipulators

- A robot manipulator is typically moved through its joints
  - Revolute: rotate about an axis
  - Prismatic: translate along an axis

But we often prefer using Cartesian frames to program motions

SCARA

6 axes robot arm
Kinematics

Cartesian Space
Tool Frame (T)  
Base Frame (B)

\[
[B_R^T, B_t^T]
\]

\(B_R^T\): Orientation of T wrt B  
\(B_t^T\): Position of T wrt B

Rigid body motion  
Transformation between coordinate frames

Joint Space

Joint 1 = \(q_1\)  
Joint 2 = \(q_2\)  
...  
Joint \(N = q_N\)

Forward Kinematics

\([B_R^p B_t^p] = f(q)\)

Inverse Kinematics

\(q = f^{-1}( [B_R^p B_t^p] )\)

Linear algebra

FORWARD KINEMATICS

INVERSE KINEMATICS
Transformation Within Joint Space

Joint spaces are defined in $\mathbb{R}^N$.

Thus for a vector of joint values

$$ q = \begin{bmatrix} q_1 \\ \vdots \\ q_N \end{bmatrix} $$

we can add/subtract joint values

$$ q_c = q_A + q_B $$

How many joints do you need? It depends on the task. But ISO 8373 requires all industrial robots to have at least three or more axes.
**Kinematics**

**FORWARD KINEMATICS**

\[
^{B}R_{T} \quad ^{B}t_{T}
\]

\[\begin{bmatrix} ^{B}R_{T} \quad ^{B}t_{T} \end{bmatrix} = f(q)\]

\[q = f^{-1}( \begin{bmatrix} ^{B}R_{p} \quad ^{B}t_{p} \end{bmatrix})\]

**INVERSE KINEMATICS**

Joint Space

\[
\text{Joint 1} = q_1
\]
\[
\text{Joint 2} = q_2
\]
\[
\vdots
\]
\[
\text{Joint N} = q_N
\]

Rigid body motion
Transformation between coordinate frames

Linear algebra
2D Rigid Motion

- Combine position and orientation:
  - Special Euclidean Group: $SE(2)$

$$SE(2) = \{(t, R) : t \in \mathbb{R}^2, R \in SO(2)\} = \mathbb{R}^2 \times SO(2)$$

- $A_t_B \in \mathbb{R}^2$ is the translation between A and B
- $A_R_B \in SO(2)$ is the rotation between A and B

If $R \in SO(2)$, then $R \in \mathbb{R}^{2 \times 2}$, $R R^T = I$ and $\det(R) = 1$

$$A_R_B = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$
3D Rigid Motion

- Combine position and orientation:
  - Special Euclidean Group: SE(3)

\[
SE(3) = \{(t, R) : t \in \mathbb{R}^3, R \in SO(3)\} = \mathbb{R}^3 \times SO(3)
\]

- \(A t_B \in \mathbb{R}^3\) is the translation between A and B
- \(A R_B \in SO(3)\) is the rotation between A and B

If \(R \in SO(3)\), then \(R \in \mathbb{R}^{3 \times 3}\), \(R R^T = I\) and \(\det(R) = 1\)

\[
A R_B = \begin{bmatrix}
  r_{11} & r_{12} & r_{13} \\
  r_{21} & r_{22} & r_{23} \\
  r_{31} & r_{32} & r_{33}
\end{bmatrix}
\]
3D Rotations

\[ \mathbf{A} \mathbf{R}_B = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \]

Can be factorized into a product of elementary rotations

\[
R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}
\]

Be careful of Commutations

\[
R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}
\]

\[
R_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

Can be written as

\[
R = R_x R_y R_z \neq R_z R_y R_x
\]
3D Rotations

- Lots of different ways to represent 3D rotations:
  - Quaternion, Euler angles, axis/angle, Rodrigues
  - They all have strengths (i.e. less than 9 numbers) and weaknesses (i.e. singularities)
  - They represent a different way to represent the SAME concept:

\[
\begin{align*}
\text{A 3x3 matrix } R & \text{ such that} \\
(R^T) R &= R (R^T) = I \\
\det( R^T ) &= +1
\end{align*}
\]
Homogeneous Representation

• A 2D point is represented by appending a “1” to yield a vector in $\mathbb{R}^3$ $P=[x \ y \ 1]^T$
• A 3D point is represented by appending a “1” to yield a vector in $\mathbb{R}^4$ $P=[x \ y \ z \ 1]^T$
• They are called homogenous coordinates
• The affine transformation of a point

$$ A_P = A R_B B P + A t_B $$

is represented by a linear transformation using a homogeneous coordinates

$$ \begin{bmatrix} A P \\ 1 \end{bmatrix} = \begin{bmatrix} A R_B & A t_B \\ 0 & 1 \end{bmatrix} \begin{bmatrix} B P \\ 1 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} $$
Homogeneous Representation

\[ \mathbf{A} \mathbf{P} = \mathbf{A} \mathbf{R}_B \mathbf{B} \mathbf{P} + \mathbf{A} \mathbf{t}_B \]

\[ \mathbf{B} \mathbf{P} = \mathbf{B} \mathbf{R}_C \mathbf{C} \mathbf{P} + \mathbf{B} \mathbf{t}_C \]

Affine transformations

This is annoying

\[ \mathbf{A} \mathbf{P} = \mathbf{A} \mathbf{R}_B \left( \mathbf{B} \mathbf{R}_C \mathbf{C} \mathbf{P} + \mathbf{B} \mathbf{t}_C \right) + \mathbf{A} \mathbf{t}_B \]

\[ \begin{bmatrix} \mathbf{A} \mathbf{P} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} \mathbf{R}_B \\ 0 \end{bmatrix} \begin{bmatrix} \mathbf{A} \mathbf{t}_B \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{B} \mathbf{P} \\ 1 \end{bmatrix} = \mathbf{A} \mathbf{E}_B \begin{bmatrix} \mathbf{B} \mathbf{P} \\ 1 \end{bmatrix} \]

Linear transformations

This is convenient

\[ \mathbf{A} \mathbf{P} = \mathbf{A} \mathbf{E}_B \mathbf{B} \mathbf{E}_C \begin{bmatrix} \mathbf{C} \mathbf{P} \\ 1 \end{bmatrix} \]

\[ \mathbf{A} \mathbf{P} = \mathbf{A} \mathbf{E}_C \begin{bmatrix} \mathbf{C} \mathbf{P} \\ 1 \end{bmatrix} \]
Kinematics

Cartesian Space
Tool Frame ($T$)
Base Frame ($B$)

$[BR_T, Bt_T]$

$BR_T$: Orientation of $T$ wrt $B$
$Bt_T$: Position of $T$ wrt $B$

FORWARD KINEMATICS

$[BR_p, Bt_T] = f(q)$

INVERSE KINEMATICS

$q = f^{-1}( [BR_p, Bt_T] )$

Joint Space

Joint 1 = $q_1$
Joint 2 = $q_2$
...
Joint $N = q_N$

Rigid body motion
Transformation between coordinate frames

Linear algebra

G.D. Hager
S. Leonard

600.436/600.636
Cartesian Transformation

Kinematic Chain
Forward Kinematics

Guidelines for assigning frames to robot links:

• There are several conventions
  – Denavit Hartenberg (DH), modified DH, Hayati, etc.
  – They are “conventions” not “laws”
  – Mainly used for legacy reason (when using 4 numbers instead of 6 per link made a difference).

1) Choose the base and tool coordinate frame
   – Make your life easy!

2) Start from the base and move towards the tool
   – Make your life easy!
   – In general each actuator has a coordinate frame.

3) Align each coordinate frame with a joint actuator
   – Traditionally it’s the “Z” axis but this is not necessary and any axis can be use to represent the motion of a joint

Barrett WAM
Rigid Body Motion 2D

- We have the coordinates of a point $P$ in the coordinate frame “C”
- Given the following robot, what are the coordinates of $P$ in the coordinate frame “A”? 

![Diagram of a robot with coordinate frames](image)
Forward Kinematics 2D

- First, what is the position and orientation of coordinate frame “B” with respect to coordinate frame “A”?
  - The position of B with respect to A is constant $A_{t_B}$
  - The orientation of B with respect to A depends on the angle $q_1$

$$A_{R_B} = \begin{bmatrix} \cos(q_1) & -\sin(q_1) \\ \sin(q_1) & \cos(q_1) \end{bmatrix}$$
Forward Kinematics 2D

• Second, what is the position and orientation of coordinate frame “C” with respect to coordinate frame “B”?

   – The position of C with respect to B is constant \( B^t C \)
   – The orientation of C with respect to B depends on the angle \( q_2 \)

\[
B_R C = \begin{bmatrix}
\cos(q_2) & -\sin(q_2) \\
\sin(q_2) & \cos(q_2)
\end{bmatrix}
\]
Forward Kinematics 2D

Forward kinematics $\mathbf{A} \mathbf{t}_B$ and $\mathbf{A} \mathbf{t}_B$ are functions of $q_1$ and $q_2$.

\[
\begin{align*}
\mathbf{B}_C &= \begin{bmatrix} \mathbf{B}_R & \mathbf{B}_T \end{bmatrix} \\
\mathbf{B}_P &= \mathbf{B}_C \mathbf{C}_P \\
\mathbf{A}_B &= \begin{bmatrix} \mathbf{A}_R & \mathbf{A}_T \end{bmatrix} \\
\mathbf{A}_P &= \mathbf{A}_B \mathbf{B}_C \mathbf{C}_P = \mathbf{A}_C \mathbf{C}_P
\end{align*}
\]
Forward Kinematics 3D

\[ R_z(q) = \begin{bmatrix} \cos q & -\sin q & 0 \\ \sin q & \cos q & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

\[ A_p = A \cdot E_B \cdot B \cdot p \]
\[ = [R_z(q_1) \quad A\mathbf{t}_B] \]
\[ B_p = B \cdot E_C \cdot C \cdot p \]
\[ = [R_z(q_2) \quad B\mathbf{t}_C] \]

\[ A_p = A \cdot E_B \cdot B \cdot E_C \cdot p \]
\[ = [R_z(q_1)R_z(q_2) \quad A\mathbf{t}_B + R_z(q_1)B\mathbf{t}_C] \]
Kinematics

**Cartesian Space**
- Tool Frame ($T$)
- Base Frame ($B$)

$$[B R_T, B t_T]$$
- $B R_T$: Orientation of $T$ wrt $B$
- $B t_T$: Position of $T$ wrt $B$

**Joint Space**
- Joint 1 = $q_1$
- Joint 2 = $q_2$
- ... Joint N = $q_N$

FORWARD KINEMATICS
- $[B R_p, B t_T] = f(q)$

INVERSE KINEMATICS
- $q = f^{-1}( [B R_p, B t_T] )$

Rigid body motion
Transformation between coordinate frames

Linear algebra
Inverse Kinematics 2D

Given $^A R_B$ and $^A t_B$ find $q$

$q$ only appears in $^A R_B$ so solving $R$ for $q$ is pretty easy. With several joints, the inverse kinematics gets very messy.

$$^A R_B(\phi) = \begin{bmatrix} 0.7071 & 0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad q = 45 \text{ degrees}$$
Likewise, in 3D we want to solve for the position and orientation of the last coordinate frame: Find $q_1$ and $q_2$ such that

$$ A E_C = \begin{bmatrix} R_z(q_1) R_z(q_2) & A t_B + R_z(q_1) B t_C \end{bmatrix} $$

Solving the inverse kinematics gets messy fast!

A) For a robot with several joints, a symbolic solution can be difficult to get

B) A numerical solution (Newton’s method) is more generic

Note that the inverse kinematics is NOT the inverse of the forward kinematics $\left(A E_B^{-1}\right)$
Kinematics

Cartesian Space
Tool Frame \((T)\)
Base Frame \((B)\)

\[
\begin{bmatrix}
  Bv_T \\
  Bw_T
\end{bmatrix}
\]

\(Bv_T\): linear vel. of \(T\) wrt \(B\)
\(Bw_T\): angular vel. of \(T\) wrt \(B\)

Rigid body motion
Transformation between coordinate frames

Joint Space

Joint 1 = \(q_1^\prime\)
Joint 2 = \(q_2^\prime\)
... 
Joint \(N = q_N^\prime\)

JACOBIAN

\[
[ v, w ]^T = J(q) \dot{q}
\]

\(\dot{q} = J^{-1}(q) [ v, w ]^T\)

INVERSE JACOBIAN

Linear algebra

\(\checkmark\)
Rigid Body Transformation
Relates two coordinate frames

Rigid Body Velocity
Relate a 3D velocity in one coordinate frame to an equivalent velocity in another coordinate frame
Rotational Velocity

We note that a rotation relates the coordinates of 3D points with

\[ A^p(t) = A^R_B(t)^B p \]

Deriving on both sides with respect to time we get

\[ v_A^p(t) = \frac{d}{dt} A^p(t) = A^\dot{R}_B B^p \]

\[ v_A^p(t) = A^\dot{R}_B (A^{R^{-1}} A R_B) B^p \]

\[ v_A^p(t) = (A^\dot{R}_B A^{R^{-1}}) A^p \]

• Point P is attached to frame B
• Frame B moves wrt to frame A
• Frame A is inertial

This skew symmetric matrix defines the spatial angular velocity.
Rotational Velocity

\[ A \dot{R}_B A R_b^{-1} \]  is skew symmetric

\[ \hat{a} = \begin{bmatrix}
0 & -a_z & a_y \\
a_z & 0 & -a_x \\
-a_y & a_x & 0
\end{bmatrix} \]

The instantaneous spatial angular velocity is defined by

\[ A \hat{\omega}_B = \begin{bmatrix}
0 & -\omega_z & \omega_y \\
\omega_z & 0 & -\omega_x \\
-\omega_y & \omega_x & 0
\end{bmatrix} = A R_B A R_b^{-1} \]

\[ A \omega_B = \begin{bmatrix}
\omega_x \\
\omega_y \\
\omega_z
\end{bmatrix} \]
Rotational Velocity

\[ A \dot{\hat{R}}_B A R_b^{-1} \] is skew symmetric

\[ \hat{a} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \]

The instantaneous spatial angular velocity is defined by

\[ A \hat{\omega}_B = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = A \dot{\hat{R}}_B A R_b^{-1} \]

\[ A\omega_B = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \]
Rigid Body Velocity

We note that a rotation relates the coordinates of 3D points with

\[
{^A}_p(t) = \begin{bmatrix} \dot{A}_B(t) & \dot{t}_B(t) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A_E(t) \\ B \end{bmatrix} = {^A}_p(t) {^B}_p
\]

Just like we did for rotations, deriving on both sides with respect to time we get

\[
\dot{v}^A_p(t) = (\dot{A}_E \dot{A}_E^{-1}) {^A}_p
\]

and we expand the matrices to be

\[
\dot{A}_E \dot{A}_E^{-1} = \begin{bmatrix} \dot{A}_B & \dot{t}_B \\ 0 & 0 \end{bmatrix} \begin{bmatrix} A_R^T & -A_R^T \dot{A}_B \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \dot{A}_B & A_R^T \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \dot{A}_B & A_R^T \dot{A}_B + \dot{A}_B \\ 0 & 0 \end{bmatrix}
\]
Rigid Body Velocity

The “s”patial velocity is defined by

\[ A \hat{V}_B^s = A \dot{E}_B A E_B^{-1} \]

Where the linear velocity is defined by

\[ A v_B^s = -A \dot{R}_B A R_B^T A t_B + A \dot{t}_B \]

And the angular velocity is defined as before by

\[ A \omega_B^s = A \dot{R}_B A R_B^T \]

Combining these two we obtain the 6D vector

\[ A V_B^s = \begin{bmatrix} A v_B^s \\ A \omega_B^s \end{bmatrix} \]
Body Velocity

- If we have $^A E_B(t)$ but we want to know the velocity of frame B with respect to frame B?

$$^A p(t) = ^A E_B(t) B p$$

$$^A E_B^{-1} v_{^A p}(t) = ^A E_B^{-1} ^A \dot{E}_B B p$$

$$v_{^B p}(t) = \left(^A E_B^{-1} ^A \dot{E}_B \right) B p$$

Most intuitive: This is your velocity with respect to yourself.
Body Velocity

The “body velocity” is defined by

\[ A\hat{V}_B = A E_B^{-1} A\dot{E}_B = \begin{bmatrix} A R_B^T A\dot{R}_B & A R_B^T A\dot{i}_B \\ 0 & 0 \end{bmatrix} \]

Where the linear velocity is defined by

\[ A\nu_B^b = A R_B^T A\dot{i}_B \]

And the angular velocity is defined as before by

\[ A\omega_B^b = A R_B^T A\dot{R}_B \]

Combining these two we obtain the 6D vector

\[ A\nu_B = \begin{bmatrix} A\nu_B^b \\ A\omega_B^b \end{bmatrix} \]
Transform Body Velocity to Spatial Velocity

If you are given a body velocity, for example say you want to:

1) Rotate the tool about a given axis (in the tool frame)
2) Drive the tool along a given axis (in the tool frame)

Then you can compute the equivalent velocity in the base frame according to

\[
\begin{bmatrix}
A\nu_B^s \\
A\omega_B^s
\end{bmatrix}
= \begin{bmatrix}
A\ R_B & A\hat{t}_B & A\ R_B \\
0 & A\ R_B & A\ \omega_B
\end{bmatrix}
\begin{bmatrix}
A\nu_B^b \\
A\omega_B^b
\end{bmatrix}
\]
Kinematics

Cartesian Space Velocity

$$[v, w]$$

- $v$: linear velocity
- $w$: angular velocity

Joint Space Velocity

Joint 1 = $q_1$
Joint 2 = $q_2$
... 
Joint N = $q_N$

JACOBIAN

$$[v, w]^T = J(q) \dot{q}$$

INVERSE JACOBIAN

$$\dot{q} = J^{-1}(q) [v, w]^T$$

Rigid body motion
Transformation between coordinate frames

Linear algebra

G.D. Hager
S. Leonard
Manipulator Jacobian

Spatial velocity of the “T”ool frame in the “B”ase frame is

\[ \dot{\hat{B}_T^s} = \dot{B}_T^s(q(t)) B E_T^{-1}(q(t)) \]

Let’s change the time varying trajectory \( B E_T(t) \) to be a time varying joint trajectory \( q(t) \)

Applying the chain rule

\[ \frac{\partial E(q(t))}{\partial t} = \frac{\partial E(q(t))}{\partial q} \frac{\partial q(t)}{\partial t} \]

\[ B V_T^s = \sum_{i=1}^{N} \left( \frac{\partial^B E_T}{\partial q_i} \dot{q_i} \right) B E_T^{-1}(q(t)) \]
Manipulator Jacobian

\[ B \dot{V}_T^s = \sum_{i=1}^{N} \left( \frac{\partial^B E_T}{\partial q_i} B E_T^{-1}(q(t)) \right) \dot{q}_i \]

Let's rewrite this result as

\[
\begin{bmatrix}
  B \dot{v}_T^s \\
  B \dot{w}_T^s
\end{bmatrix} = J(q) \dot{q}
\]

Where \( J(q) \) is a 6xN matrix called the **manipulator Jacobian** that relates joint velocities to the Cartesian velocity of the tool. Note that \( J(q) \) depends on “\( q \)” and, therefore, on the robot’s configuration.
Rigid body motion
Transformation between coordinate frames

Cartesian Space
Tool Frame (\(T\))
Base Frame (\(B\))

\[
[ Bv_T, \ Bw_T ]
\]

- \(Bv_T\): linear vel. of \(T\) wrt \(B\)
- \(Bw_T\): angular vel. of \(T\) wrt \(B\)

Joint Space

\[
[ \dot{v}, \dot{w} ]^T = J(q) \dot{q}
\]

- Joint 1 = \(q_1^*\)
- Joint 2 = \(q_2^*\)
- ... \(N = q_N^*\)

Linear algebra

JACOBIAN

INVERSE JACOBIAN
Manipulator Jacobian

We just derived that given a vector of joint velocities, the velocity of the tool as seen in the base of the robot is given by

$$
\begin{bmatrix}
\dot{B}v^s_T \\
B\omega^s_T
\end{bmatrix} = J(q)\dot{q}
$$

If, instead we want the tool to move with a velocity expressed in the **base** frame, the corresponding joint velocities can be computed by

$$
\dot{q} = J^{-1}(q)\begin{bmatrix}
\dot{B}v^s_T \\
B\omega^s_T
\end{bmatrix}
$$

Inverting a matrix is much easier than computing the inverse kinematics!
Manipulator Jacobian

What if the Jacobian has no inverse?

A) No solution: The velocity is impossible
   B) Infinity of solutions: Some joints can be moved without affecting the velocity (i.e. when two axes are colinear)

The robot cannot move in this direction when the robot is in this configuration. Hence $J(q)$ is singular.

In this configuration, $q_1$ and $q_3$ can counter rotate. Hence $J(q)$ is singular.

\[ \dot{q}_1 = -\dot{q}_3 \]