Robot Kinematics: *Planar Arms*
Robot Arms

• A robot arm (aka serial link manipulator) consists of a series of rigid links, connected by joints (motors), each of which has a single degree of freedom.
  • Revolute Joint: Single degree of freedom is rotation about an axis.
  • Prismatic joint: Single degree of freedom is translation along an axis.
Describing Serial Link Arms

• Number the links in sequence.
• For a robot with \( n \) joints:
  • Base (which does not move) is Link 0.
  • End-effector (tool) is attached to Link \( n \).
  • Joint \( i \) connects Link \( i - 1 \) to Link \( i \).
  • We define the joint variable \( q_i \) for joint \( i \) as:

\[
q_i = \begin{cases} 
\theta_i & \text{if joint } i \text{ is revolute} \\
 d_i & \text{if joint } i \text{ is prismatic}
\end{cases}
\]

Two-link Planar Arm:
• \( n = 2 \),
• both links are always coplanar (no rotation out of the plane).
• \( q_1 = \theta_1, \ q_2 = \theta_2 \)
Manipulator Kinematics

• Kinematics describes the position and motion of a robot, without considering the forces required to cause the motion.

• Forward Kinematics: *Given the value for each joint variable, $q_i$, determine the position and orientation of the end-effector (gripper, tool) frame.*

**The basic idea:**
➢ Assign lots of coordinate frames, and express these frames in terms of the joint variables, $q_i$. 
General Approach

• Each link is a rigid body.

• We know how to describe the position and orientation of a rigid body:
  • Attach a coordinate frame to the body.
  • Specify the position and orientation of the coordinate frame relative to some reference frame.

• If two links, say link $i - 1$ and link $i$ are connected by a single joint, then the relationship between the two frames can be described by a homogeneous transformation matrix $T_{i-1}^i$ which will depend only on the value of the joint variable!
Homogeneous Transformations

We can simplify the equation for coordinate transformations by augmenting the vectors and matrices with an extra row:

\[
\begin{bmatrix}
P_0^0 \\
P_1^0
\end{bmatrix} = \begin{bmatrix}
R_1^0 P_1^1 + d_0^0 \\
0_2
\end{bmatrix} = \begin{bmatrix}
R_1^0 & d_0^0 \\
0_2 & 1
\end{bmatrix} \begin{bmatrix}
P_1^1 \\
1
\end{bmatrix}
\]

\[\vec{P}^0 = [P_0^0, 1], \quad \vec{P}^1 = [P_1^1]\]

\[\vec{P}^0 = T_1^0 \vec{P}^1\]

- \(T_1^0\) is called a homogeneous transformation matrix
- \(\vec{P}^0\) are the homogeneous coordinates for \(P^0\)
Composition of Transformations

From our previous results, we know:

\[ \tilde{P}^0 = T_1^0 \tilde{P}^1 \]
\[ \tilde{P}^1 = T_2^1 \tilde{P}^2 \]

But we also know:

\[ \tilde{P}^0 = T_2^0 \tilde{P}^2 \]

\[ T_2^0 = T_1^0 T_2^1 \]

This is the composition law for homogeneous transformations.
What about robot arms??

- Attach a coordinate frame to each link of the robot!
- Frame 0 is attached to Link 0, which is merely the fixed mounting point to the environment.
- Now, the trick is to express $T_{i-1}^i$ as a function of $\theta_i$
A special case

Suppose the axis $x_i$ is collinear with the origin of Frame $i - 1$:
- $x_1$ is collinear with the origin of Frame 0
- $x_2$ is collinear with the origin of Frame 1

$T_i^{i-1} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 \\ \sin \theta_i & \cos \theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & a_i \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i & a_i \sin \theta_i \\ 0 & 0 & 1 \end{bmatrix}$

Use this to simplify link coordinate frames!
Assigning Coordinate Frames to Links

- Frame 0 (the base frame) has its origin at the center of Joint 1 (on the axis of rotation).
- Frame $i$ is **rigidly attached** to Link $i$, and has it’s origin at the center of Joint $i + 1$.
- The $x_i$-axis is collinear with the origin of Frame $i - 1$.
- The link length, $a_i$ is the distance between the origins of Frames $i$ and $i - 1$.
- The homogeneous transformation that relates adjacent frames is given by:

$$T_{i-1}^i = \begin{bmatrix}
\cos \theta_i & -\sin \theta_i & a_i \cos \theta_i \\
\sin \theta_i & \cos \theta_i & a_i \sin \theta_i \\
0 & 0 & 1
\end{bmatrix}$$
Assigning Link Frames

- $\theta_1$ is the angle from $x_0$ to $x_1$.
- $x_2$ is collinear with the origin of Frame 1.
- $\theta_2$ is the angle from $x_1$ to $x_2$.
- $x_1$ is collinear with the origin of Frame 0.

- Frame $n$ is the end-effector frame. It can be attached to link $n$ in any manner that is convenient.
- In this case, $n = 2$, so Frame 2 is the end-effector frame.
The Forward Kinematic Map

• The forward kinematic map gives the position and orientation of the end-effector frame as a function of the joint variables:

\[ T_n^0 = F(q_1, \ldots, q_n) \]

• For the two-link planar arm, we have

\[
T_2^0 = \begin{bmatrix}
\cos \theta_1 & -\sin \theta_1 & a_1 \cos \theta_1 \\
\sin \theta_1 & \cos \theta_1 & a_1 \sin \theta_1 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos \theta_2 & -\sin \theta_2 & a_2 \cos \theta_2 \\
\sin \theta_2 & \cos \theta_2 & a_2 \sin \theta_2 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & a_1 \cos \theta_1 + a_2 \cos(\theta_1+\theta_2) \\
\sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & a_1 \sin \theta_1 + a_2 \sin(\theta_1+\theta_2) \\
0 & 0 & 1
\end{bmatrix}
\]
Simple Geometry...

\[
\begin{align*}
\theta_1 + \theta_2 &= \theta_1 + \theta_2 \\
\cos \theta_1 &= a_1 \\
\sin \theta_1 &= a_1 \\
\cos (\theta_1 + \theta_2) &= a_2 \\
\sin (\theta_1 + \theta_2) &= a_2 \\
\end{align*}
\]
Simple Geometry...

\[
T_2^0 = \begin{bmatrix}
\cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & a_1 \cos \theta_1 + a_2 \cos(\theta_1+\theta_2) \\
\sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & a_1 \sin \theta_1 + a_2 \sin(\theta_1+\theta_2) \\
0 & 0 & 1
\end{bmatrix}
\]
We can parameterize the end effector frame by \((X_e, Y_e, \phi)\)

\[
T_3^0 = \begin{bmatrix}
C_{123} & -S_{123} & a_1 C_1 + a_2 C_{12} + a_3 C_{123} \\
S_{123} & C_{123} & a_1 S_1 + a_2 S_{12} + a_3 S_{123} \\
0 & 0 & 1
\end{bmatrix}
\]

\[
C_{123} = \cos(\theta_1 + \theta_2 + \theta_3), \text{ etc.}
\]

\[
T_3^0 = \begin{bmatrix}
\cos \phi & -\sin \phi & X_e \\
\sin \phi & \cos \phi & Y_e \\
0 & 0 & 1
\end{bmatrix}
\]
About the Forward Kinematic Map

• For the two-link arm, we can **position** the end-effector origin anywhere in the arm’s workspace: two inputs \((\theta_1, \theta_2)\) and two “outputs” \((X_e, Y_e)\).

• For the three-link arm, we can position the end-effector origin anywhere in the arm’s workspace, **and** we can choose the orientation of the frame: three inputs \((\theta_1, \theta_2, \theta_3)\) and three “outputs” \((X_e, Y_e, \phi)\).

• Suppose we had a four-link arm?
  • Infinitely many ways to achieve a desired end-effector configuration \((X_e, Y_e, \phi)\).
More General Robot Arms

- With a bit of work, this can be generalized to arbitrary robot arms.
- We shall not do this bit of work in CS3630.
Motion Control

• Trajectory following is important
  • Spray-painting
  • Sealing
  • Welding

• Three main approaches:
  • Trajectory replay
  • Joint-space Motion Control
  • Cartesian Motion Control
Trajectory Replay

• Teaching by demonstration
• Define a set of **waypoints** by “showing” the robot
• Similar to keyframe animation in graphics
• Still need to interpolate between waypoints
RRR example

\[ T^0_1 = \begin{bmatrix}
\cos \theta_1 & -\sin \theta_1 & 3.5 \cos \theta_1 \\
\sin \theta_1 & \cos \theta_1 & 3.5 \sin \theta_1 \\
0 & 0 & 1
\end{bmatrix} \]

\[ T^1_2 = \begin{bmatrix}
\cos \theta_2 & -\sin \theta_2 & 3.5 \cos \theta_2 \\
\sin \theta_2 & \cos \theta_2 & 3.5 \sin \theta_2 \\
0 & 0 & 1
\end{bmatrix} \]

\[ T^2_3 = \begin{bmatrix}
\cos \theta_3 & -\sin \theta_3 & 2 \cos \theta_3 \\
\sin \theta_3 & \cos \theta_3 & 2 \sin \theta_3 \\
0 & 0 & 1
\end{bmatrix} \]

• End-effector == frame 3

(a) \( \theta_1 = 112^\circ, \theta_2 = -52^\circ, \) and \( \theta_3 = -60^\circ \)
RRR example, cont’d

• Multiply 3 matrices
• Note R in upper left
• Check orientation!

(b) $\theta_1 = 60^\circ$, $\theta_2 = -45^\circ$, and $\theta_3 = -90^\circ$

$$T_i^s(q) = \begin{pmatrix}
\cos \beta & -\sin \beta & 3.5 \cos \theta_1 + 3.5 \cos \alpha + 2 \cos \beta \\
\sin \beta & \cos \beta & 3.5 \sin \theta_1 + 3.5 \sin \alpha + 2 \sin \beta \\
0 & 0 & 1
\end{pmatrix}$$

with $\alpha = \theta_1 + \theta_2$ and $\beta = \theta_1 + \theta_2 + \theta_3$, the latter being the tool orientation.
Proportional Feedback Control

• Feedback law:

\[ q_{t+1} = q_t + K_p(q_d - q_t) \]

• At every time step:
  • Calculate **joint space error** \( e_t = q_d - q_t \)
  • Increase of decrease proportional to \( e_t \)
  • \( K_p \) is proportional gain parameter
Proportional Feedback Control

- Properties:
  - Closer to goal -> smaller steps
  - Automatically reverses sign if we overshoot
  - Generalizes to vector-valued control

- Value of $K_p$ really matters:
  - too high: overshoot
  - too low: slow convergence

- Special case of PID control

https://arduinoplusplus.wordpress.com/2017/06/21/pid-control-experiment-tuning-the-controller/
The Manipulator Jacobian

• Velocity of end-effector if we move any given joint?
• Given by arrows:
  • R=joint 1
  • G=joint 2
  • B=joint 3
Jacobian = linear map

- Linear relationship between joint space velocity and cartesian velocity (pose space!)

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix}^T = J(q) \dot{q}
\]

- $J$ is $3 \times n$ matrix:

\[
J(q) \triangleq \begin{bmatrix}
J_1(q) & J_2(q) & \ldots & J_n(q)
\end{bmatrix}
\]

- Each $J_i(q)$ column corresponds to arrow.
- Partial derivative of pose wrt $q_i$
Worked Example: RRR manipulator

• Remember:

\[
T_t^s(q) = \begin{pmatrix}
\cos \beta & -\sin \beta & 3.5 \cos \theta_1 + 3.5 \cos \alpha + 2 \cos \beta \\
\sin \beta & \cos \beta & 3.5 \sin \theta_1 + 3.5 \sin \alpha + 2 \sin \beta \\
0 & 0 & 1
\end{pmatrix}
\]

• Extracting x, y, theta:

\[
\begin{pmatrix} x(q) \\ y(q) \\ \theta(q) \end{pmatrix} = \begin{pmatrix} 3.5 \cos \theta_1 + 3.5 \cos \alpha + 2 \cos \beta \\ 3.5 \sin \theta_1 + 3.5 \sin \alpha + 2 \sin \beta \\ \beta \end{pmatrix}
\]

• So what is Jacobian???
Worked Example: RRR manipulator

• \(x, y, \theta\):

\[
\begin{bmatrix}
  x(q) \\
  y(q) \\
  \theta(q)
\end{bmatrix} =
\begin{bmatrix}
  3.5 \cos \theta_1 + 3.5 \cos \alpha + 2 \cos \beta \\
  3.5 \sin \theta_1 + 3.5 \sin \alpha + 2 \sin \beta \\
  \beta
\end{bmatrix}
\]

• Jacobian:

\[
\begin{pmatrix}
  -3.5 \sin \theta_1 - 3.5 \sin \alpha - 2.5 \sin \beta & -3.5 \sin \alpha - 2.5 \sin \beta & -2 \sin \beta \\
  3.5 \cos \theta_1 + 3.5 \cos \alpha + 2.5 \cos \beta & 3.5 \cos \alpha + 2.5 \cos \beta & 2 \cos \beta \\
  1 & 1 & 1
\end{pmatrix}
\]
Cartesian Motion Control

- Convert direction in cartesian space to direction in joint space
- Yields straight-line paths
How do we convert?

- We want a straight line!
- Calculate (scaled) direction of the line
- Error in cartesian space:

\[
E_t(q) = \begin{bmatrix}
e_x \\
e_y \\
e_\theta
\end{bmatrix} = \begin{bmatrix}
x_d - x(q_t) \\
y_d - y(q_t) \\
\theta_d - \theta(q_t)
\end{bmatrix}
\]

- Then, simple proportional control:

\[
q_{t+1} = q_t + K_p J(q_t)^{-1} E_t(q)
\]

Small print: we have to take care when subtracting angles, as they are not unique
1. **Forward Kinematics** is just multiplying transforms
2. We went through an **RRR Worked Example**
3. **Joint-Space Motion Control** creates paths that minimize distance in joint space
4. The **Manipulator Jacobian** provides a relationship between cartesian and joint-space velocities/displacements
5. **Cartesian Motion Control** exploits this relationship to provide predictable paths in cartesian space