Movement generation for robot arms

Kinematics and Attractor Dynamics for manipulators
robotic arms

they aren’t vehicles
movement generation for vehicles

- the floor is a 2D environment
- vehicle treated as point
- task: reach goal
- task: avoid obstacles
- not much more vehicles can do
arms: what changes?

- where we move: environment: 3D
- what we do: more tasks are possible at the same time or in sequence: e.g. manipulation
- an interesting point on the arm is the **end-effector**
- what we move: chain-of-links or segments geometry (**kinematic chain**)
- but moving a link can affect other links. complication.
arms: what changes?

• different tasks active at different times: system needs to combine tasks that switch on/off all the time

• does Attractor Dynamics approach scale-up? what happens when multiple tasks are active at the same time? does it work? why?
rigid bodies

- cannot treat robot as single point in space, anymore
- connected links
- orientation and translation for each link: two times 3 dimensions
- we need a way to relate our task to the links translation and orientation
- note: not always require specific orientation and specific translation for link at the same time
kinematics and kinetics

- **kinematics**: movement *without* forces
- **kinetics**: (dynamics, not in the mathematical sense) movement *with* forces
- important acting forces: gravitation, interaction of links
- we push kinetics out to low-level controller. modern robots know their own dynamics.
how does the arm move?

- joints: revolute, spherical, cylindrical, prismatic
- how many DoF and what kind of joints does the human arm have?
- typically position controlled servo-motors
formal constraints

• **workspace**: either the environment or sometimes space of reachable positions \( p \) or \( x \) (vectors) of the end-effector. Euclidian.

• **configuration space**: space of all possible (here:) joint positions \( \theta \) (vectors). Also **joint space**.

• **task constraints**: equations (equalities or inequalities) that need to be successfully satisfied for the task. can be vector-valued.

• **holonomic** constraints: expressible purely via configuration (and time). Reduces dimension of workspace.

• **non-holonomic** constraints: Velocity-based constraints. Introduces path-dependency. Typically vehicles are non-holonomic robots (can’t move side-ways).

• **Degrees of Freedom**: dimensionality of configuration space.

• **Redundancy**: Compare DoF and dimensions of task contraints. More DoF than necessary? Infinite solutions to constraints possible.
Kinematics
where is the hand?

- **forward kinematics**
- example: single revolute joint
- \( p(\theta_1) = \begin{pmatrix} l_1 \cos \theta_1 \\ l_1 \sin \theta_1 \end{pmatrix} \)
- generally: \( p \) is a function of \( \theta \)
where is the hand?

- forward kinematics
- example: revolute joint and prismatic joint

\[
p(\theta) = p(\theta_1, \theta_2) = \begin{pmatrix} l_1 + \theta_1 + l_2 \cos \theta_2 \\ l_2 \sin \theta_2 \end{pmatrix}
\]
what happens if I move a joint?

- differential (forward) kinematics \( \dot{p} = J \dot{\theta} \)

- (kinematic) Jacobian matrix \( J \)

\[
J = \begin{pmatrix}
\frac{\partial p_1(\theta)}{\partial \theta_1} & \frac{\partial p_1(\theta)}{\partial \theta_2} \\
\frac{\partial p_2(\theta)}{\partial \theta_1} & \frac{\partial p_2(\theta)}{\partial \theta_2}
\end{pmatrix}
\]

\[
= \begin{pmatrix}
1 & -l_2 \sin \theta_2 \\
0 & l_2 \cos \theta_2
\end{pmatrix}
\]
what happens if I move a joint?

\[ J = \begin{pmatrix} \frac{\partial p_1(\theta)}{\partial \theta_1} & \frac{\partial p_1(\theta)}{\partial \theta_2} \\ \frac{\partial p_2(\theta)}{\partial \theta_1} & \frac{\partial p_2(\theta)}{\partial \theta_2} \end{pmatrix} \]

\[ = \begin{pmatrix} 1 & -l_2 \sin \theta_2 \\ 0 & l_2 \cos \theta_2 \end{pmatrix} \]

\[ \dot{p} = J \dot{\theta} \]
what happens if I move a joint?

\[ J = \begin{pmatrix} \frac{\partial p_1(\theta)}{\partial \theta_1} & \frac{\partial p_1(\theta)}{\partial \theta_2} \\ \frac{\partial p_2(\theta)}{\partial \theta_1} & \frac{\partial p_2(\theta)}{\partial \theta_2} \end{pmatrix} \]

\[ = \begin{pmatrix} 1 & -l_2 \sin \theta_2 \\ 0 & l_2 \cos \theta_2 \end{pmatrix} \]

\[ \dot{p} = J \dot{\theta} \]
what happens if I move a joint?

• differential (or instantaneous) kinematics provide a relationship between velocities

\[ \dot{p} = J \dot{\theta} \]

\[ J = \frac{\partial p(\theta)}{\partial \theta} \]

• note: J changes when \( \theta \) changes

• what happens when J is singular? kinematic singularity. rank changes

• since J changes, these singularities can appear and disappear (at certain configurations) while moving

• nullspace of J: space of all \( \dot{\theta} \) that project to a \( \dot{p} \) of 0.
how do I get the hand to where I want it?

• we now need to look at the inverse problem: what joints do I need to set to what values to reach a certain point in workspace?

• **closed form** solution (inverse of the forward kinematics)

• the forward kinematics $p(\theta)$ can in general not be analytically inverted

• geometrical construction. depends on geometry of robot!
how do I get the hand to where I want it?

\[ \theta_1 = \arctan_2(y, x) \pm \beta \]

\[ \theta_2 = \pi \pm \alpha \]

\[ \alpha = \cos^{-1} \left( \frac{l_1^2 + l_2^2 - r^2}{2l_1l_2} \right) \]

\[ \beta = \cos^{-1} \left( \frac{r^2 + l_1^2 - l_2^2}{2l_1l_2} \right) \]
how do I get the hand to where I want it?

• the differential kinematics may be simpler to invert?

\[ \dot{p} = \frac{dp}{dt} = \frac{dp}{d\theta} \frac{d\theta}{dt} = J \dot{\theta} \]

• ... if J is invertible.

• is J singular?

• is J even quadratic?

• iff invertible: \( \dot{\theta} = J^{-1} \dot{p} \)

we can calculate a commanded joint velocity

• integrate \( \dot{\theta} \) to \( \theta \) to send commands
inverse of differential kinematics

- if $J$ is not invertible, we can use the Moore-Penrose pseudo-inverse

$$J^+ = J^T (JJ^T)^{-1}$$

- a generalized matrix inverse

- $\dot{\theta} = J^+ \dot{p}$

- property: minimizes $|\dot{\theta}|$
Attractor Dynamics for robot arms
recap: tasks in Attractor Dynamics

- task as differential equation
- task is adhered-to if system is in a fixed-point
- move quickly into attractor state
- in reality: near attractor suffices
- avoidance: repellors
- task akin “forcelet”
generating complex movements
different tasks

• reach bottle
• grasp bottle
• pour the drink
• put bottle on table
• avoid obstacles

• hand position, bottle position
• hand orientation, hand opening, hand closing
• bottle orientation, glass position, glass filling
• obstacle positions, if any
different tasks

• different variables "\( \phi \)" are relevant for different tasks

• a task can be expressed as constraint on that variable (stable fixed-point in a dynamical system)

• but how do \( \phi \) and \( \dot{\phi} \) relate to \( \theta \) and \( \dot{\theta} \) in joints?

• task defines submanifold on configuration space

• different tasks live on different sub-manifolds of configuration space. how can this work?
independent stabilization

- independent forcelets
- each a (possibly different) relevant variable
- constraints expressed as attractors/repellors in dynamical system over that relevant variable only
- find joint space changes that realize this task (independently)
reintegration of independent tasks

- superposition of independent forcelets
- now new vector field realizes compromise of tasks

task constraint realized
reintegration of independent tasks

joint angles that realize task 1
(hand position)
reintegration of independent tasks

joint angles that realize task 2
(hand orientation)
reintegration of independent tasks

joint angles that realize task 1 and task 2
reaching
reaching

• deviation angle dynamics, analogously to heading angle dynamics (define a plane M)

\[ \dot{\phi} = f_{\text{dir}} = -\alpha_{\phi} \sin \phi \]

• angle: \( \phi = \angle(\dot{x}, k) \)

• insert a step: In workspace, what vector would realize the change?
reaching
reaching

- from geometry we can find:
  \[
  \mathbf{v}_\perp = \left( \mathbf{k} - \frac{\langle \mathbf{k}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \mathbf{v} \right) \frac{|\mathbf{v}|}{|\mathbf{k} - \frac{\langle \mathbf{k}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \mathbf{v}|}
  \]

- transformation of forcelet into workspace:
  \[
  \mathbf{f}_{dir} = \mathbf{f}_{dir} \cdot \mathbf{v}_\perp = -\alpha \phi \sin \phi \cdot \mathbf{v}_\perp
  \]
reaching

• transformation from workspace into joint-space:

• per inverse differential kinematics:

\[
J^+ = J^T (JJ^T)^{-1}
\]

\[
F_{dir} = J^+ \cdot f_{dir} = -\alpha \phi \sin \phi \cdot J^+ \cdot \mathbf{v}_\perp
\]

• we now have a “forcelet” in joint space
speed

- analogous to the vehicle scenario, speed treated as independent task:
  - $v = |\mathbf{v}|$
  - select a desired speed: $v_{des}$
  - $\hat{v} = \frac{\mathbf{v}}{|\mathbf{v}|}$

\[
\begin{align*}
  f_{vel} &= -\alpha_{vel}(v - v_{des}) \\
  f_{vel} &= f_{vel} \cdot \hat{v} \\
  F_{vel} &= J^+ \cdot f_{vel}
\end{align*}
\]
obstacle avoidance
obstacle avoidance

• finding a instantaneous joint change that enacts the required (instantaneous) task change: find direction that moves the relevant task variable into its attractor

• other take on it: find direction that moves the relevant task variable away from its repellor

• problem: all links must be able to avoid. but moving proximal links also moves distal ones (kinematic chain)
for every link:

• find closest points \( o \) on obstacle and \( s \) on link
• in what direction does link point \( s \) currently move?
• in what direction should it move?

note: \( s \) does not have the same forward kinematics and not the same Jacobian as the end-effector!
construction on normal plane

“shadow of obstacle” on plane N
avoidance with upwards bias (rotated)

direct avoidance
other parameters

- distance range
- angular range
gripper orientation

- angle dynamics
- different geometrical construction and Jacobian but same principle
- requires one DoF of the system, thus preferable only enforce when necessary. NOT ALWAYS ON

\[ f_{ori} = -\alpha \gamma \sin \gamma \]
superposition of tasks

• finally, superpose all independently stabilizing vector fields:

\[ \mathbf{F} = \mathbf{F}_{dir} + \mathbf{F}_{vel} + \sum_{obs,seg} \mathbf{F}_{obs} \]

• interpret the vector-field as acceleration command:

\[ \ddot{\theta} = \mathbf{F} \]
outlook: behavior
organization

sequences of tasks!