Advanced Robot Control and Learning

Munich Institute of Robotics and Machine Intelligence

Peng,Xie

Technische Universität München Arcisstraße 21, 80333 München

February, 2023

Abstract

Advanced Robot Control and Learning is a course that focuses on the design, implementation and analysis of control algorithms and learning techniques for robots. The course covers a range of topics including robot kinematics and dynamics, control systems, and reinforcement learning. The article is split into two sections. The first section provides an overview of the theoretical knowledge of robot kinematics and dynamics, which includes the topics of space conversion and various control methods. The latter section focuses primarily on machine learning.

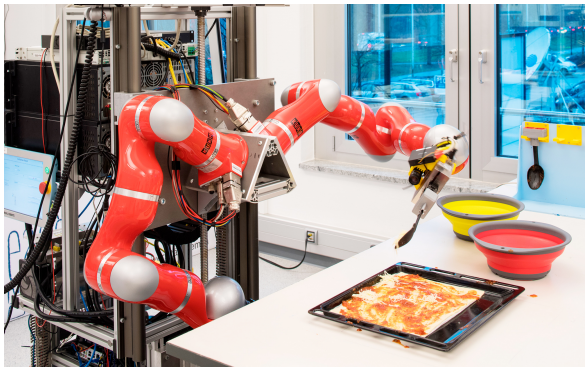


Figure 1: Artificial intelligence: degree courses in Germany

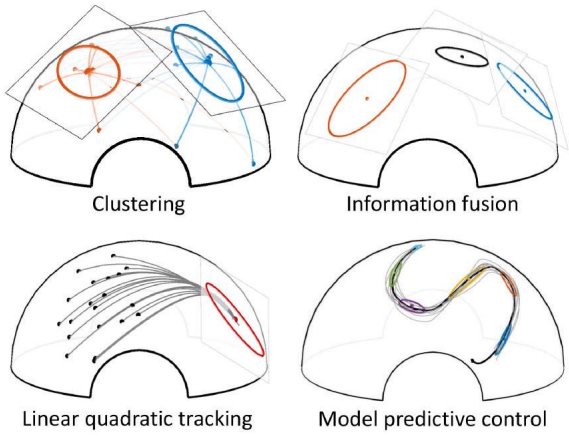


Figure 2: Gaussian-based representation on Riemannian manifolds

1. Differential Geometry

Differential geometry is an important tool in robotics, as it allows robots to navigate complex surfaces with ease. It can be used to model the shape of objects, determine the motion of the robot, and even provide information about the environment.

1.1 Riemannian manifolds

Two basic notions of Riemannian geometry are crucial for robot learning and adaptive control applications

Geodesics: Geodesics refer to the shortest possible paths between two points on a Riemannian manifold. Just like straight lines in Euclidean space, the second derivative of a geodesic is constant and equal to zero along its entire length.The exponential map $\text{Exp}_{\boldsymbol{x}_0} : \mathcal{T}_{\boldsymbol{x}_0}\mathcal{M} \to \mathcal{M}$ maps a point $\boldsymbol{u}$ in the tangent space of $\boldsymbol{x}_0$ to a point $\boldsymbol{x}$ on the manifold, so that $\boldsymbol{x}$ lies on the geodesic starting at $\boldsymbol{x}_0$ in the direction $\boldsymbol{u}$. The norm of $\boldsymbol{u}$ is equal to the geodesic distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}$. The inverse map is called the logarithmic map $\log_{\boldsymbol{x}_0} : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}_0}\mathcal{M}$.

Parallel transport:$\Gamma_{g\to h} : \mathcal{T}_g\mathcal{M} \to \mathcal{T}_{\boldsymbol{h}}\mathcal{M}$The process of moving vectors between tangent spaces while maintaining their inner product is facilitated by the use of a connection. A connection defines the relationship between vectors in tangent spaces that are infinitesimally close to each other. This connection enables a smooth transport of a vector from one tangent space to another by sliding it along a curve with tiny, incremental moves.

1.2 Manifolds in robot applications

The most common manifolds in robotics Two Lie groups: the special orthogonal group SO(3) and the special Euclidean group SE(3)

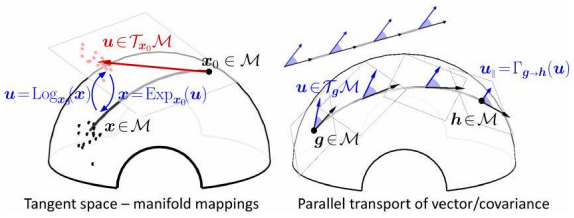


Figure 3: Applications in robotics using Riemannian manifolds

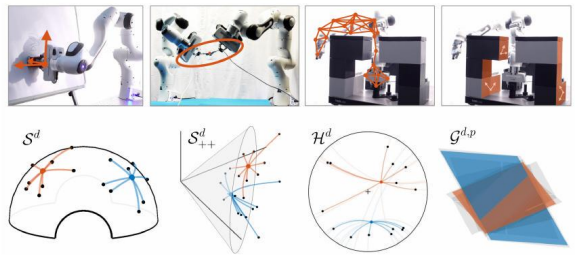


Figure 4: Structured manifolds in robotics

- **The sphere manifold** $\mathcal{S}^d$ is a key concept in robotics for encoding directions and orientations. This is achieved by using unit quaternions $\mathcal{S}^3$ to depict the orientation of the end effector (tool tip). Surfaces' perpendicular unit directional vector, required for contact planning, can be represented through $\mathcal{S}^2$. Articulatory joints in robotics can be symbolized on the torus $\mathcal{S}^1 \times \mathcal{S}^1 \times \ldots \times \mathcal{S}^1$. The Kendall shape space, utilized to encode 3D skeletal motion capture data, is also based on unit spheres.
- **The special orthogonal group** $\mathrm{SO}(d)$ represents the group of rotations around the origin in a d-dimensional space. The groups $\mathrm{SO}(2)$ and $\mathrm{SO}(3)$ are widely used in robotics.
- **The special Euclidean group** $\mathrm{SE}(3)$ is the group of rigid body transformations, which is comprised of rotations and translations. The geometry of $\mathrm{SE}(3)$ can be used to describe the kinematics and the Jacobian of robots, making it a popular choice for describing robot motion and pose estimation. $\mathbb{R}^d$.
- **The manifold of symmetric positive definite (SPD) matrices** $\mathcal{S}^d_{++}$ is utilized in a variety of ways within robotics.
- **Hyperbolic manifolds** $\mathcal{H}^d$ are the analogues of spheres with constant negative curvature instead of constant positive curvature.
- **Grassmannian** $\mathcal{G}^{d,p}$ is the manifold of all p-dimensional subspaces of $\mathbb{R}^d$. [2]

1.3 Twists and Wrenches

In robotics, twists and wrenches are two related concepts used to describe the motion and forces of rigid bodies.

A twist is a mathematical object that describes the instantaneous velocity and direction of a rigid body's motion. It consists of a rotational component and a translational component, represented by a six-dimensional vector.

A wrench, on the other hand, is a mathematical object that describes the forces and torques acting on a rigid body. Like a twist, it consists of a rotational component and a translational component, represented by a six-dimensional vector.

Together, twists and wrenches form a powerful mathematical framework for analyzing the motion and forces of robotic systems and are used extensively in fields such as robot kinematics and dynamics, control, and path planning.

2. Task Space Modeling and Control

Task Space Modeling and Control is a widely used approach in robotics for controlling the motion of robots in terms of tasks to be achieved in the workspace, rather than directly controlling the joint angles or velocities of the robot. Task Space Modeling involves defining the desired tasks or goals in terms of position and orientation of end-effectors or tools in the workspace, and modeling the relationship between the end-effector motion and the motion of the robot's joints. This typically involves the use of mathematical tools such as homogeneous transformations, Jacobian matrices, and differential kinematics. Task Space Control involves using the task space model to generate control commands that move the end-effector towards the desired task goal while avoiding obstacles or constraints. This can be done using various control techniques such as Proportional-Integral-Derivative (PID) control, Model Predictive Control (MPC), or other advanced control algorithms. Task Space Modeling and Control are widely used in robotics applications such as manipulation, grasping, and mobile robots, where the desired task goals are often specified in terms of position and orientation of end-effectors or tools in the workspace.

2.1 Joint Space Control VS Task Space Control

Joint space control and task space control are two common approaches for controlling the motion of robots, each with its own advantages and limitations.

- Joint space control involves controlling the motion of the robot by directly specifying the position, velocity, or torque of each joint. This approach is commonly used in robotics applications, such as pick-and-place tasks or joint-space obstacle avoidance. Joint space control is often simpler to implement than task space control, but can be more difficult to use for more complex tasks.

- Task space control, on the other hand, involves specifying the desired motion of the end-effector or tool in the workspace, rather than controlling the motion of the joints directly. This approach is more intuitive for users, as it allows them to define the task in terms of the desired position and orientation of the end-effector. Task space control is commonly used for more complex tasks, such as manipulation or grasping, as it allows the robot to move in a more natural way.

In summary, joint space control is suitable for simpler tasks and precise joint control, while task space control is better suited for more complex tasks and intuitive control of the end-effector. Both approaches have their own advantages and limitations, and the choice of control approach depends on the specific task and the needs of the application.

2.2 Inverse Kinematics

The inverse kinematics of the system translates position, velocity, and acceleration from Cartesian space backward to configuration space and string lengths. Therefore admissible control inputs for the system are obtained via the inverse kinematics of desired trajectories in Cartesian space. First, we define a vector for each space, collecting all variables in a specific order.

$$\underline{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix} \quad \underline{c} = \begin{bmatrix} r \\ \theta \\ \varphi \end{bmatrix} \quad \underline{e} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

The function mapping positions in Cartesian space to configuration space is called $\underline{c}(\underline{e})$ and is defined as

$$\underline{c}(\underline{e}) = \begin{bmatrix} r(\underline{e}) \\ \theta(\underline{e}) \\ \varphi(\underline{e}) \end{bmatrix} = \begin{bmatrix} \frac{x^2+y^2+z^2}{2\sqrt{x^2+y^2}} \\ \cos^{-1}\left(\frac{-x^2-y^2+z^2}{x^2+y^2+z^2}\right) \\ \operatorname{atan}2(y,x) \end{bmatrix}$$

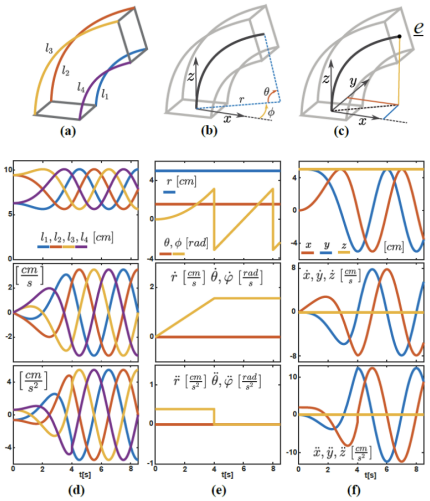


Figure 5: (a) actuator space. (b) configuration space. (c) Cartesian space. (d) kinematics in actuator space. (e) kinematics in configuration space. (f) kinematics in Cartesian space. rigid body dynamics equations

2.3 Control Theory

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \tau$$

where $\mathbf{q} \in \mathrm{Re}^n$ is the joint angle vector, $\mathbf{M}(\mathbf{q})$ is the inertia for complex humanoids. $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis/centrifugal vector. $\mathbf{g}(\mathbf{q})$ is the ultimate gravity vector and τ is the joint torque vector.

$$g(\mathbf{q}) = \frac{1}{2} \left(\mathbf{q} - \mathbf{q}_{\text{rest}}\right)^T \mathbf{K}_w \left(\mathbf{q} - \mathbf{q}_{\text{rest}}\right)$$

where $q \in \mathrm{Re}^n$ is the joint angle vector, $\mathbf{M}(\mathbf{q})$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis/centrifugal vector, $\mathbf{g}(\mathbf{q})$ is the gravity vector, and τ is the joint torque vector. The forward kinematics and differential relationship between the joint coordinates and the task coordinates are given as

$$\begin{aligned} \mathbf{x} &= \mathbf{f}(\mathbf{q}) \\ \dot{\mathbf{x}} &= \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \\ \ddot{\mathbf{x}} &= \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} \end{aligned}$$

where $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix and $\mathbf{x} \in \mathrm{Re}^m$, where $m < n$, is the task coordinate vector. The basic idea in redundancy resolution in the velocity and acceleration levels is to compute the desired joint velocities and accelerations respectively as

$$\begin{aligned} \dot{\mathbf{q}}_d &= \mathbf{J}^\dagger \dot{\mathbf{x}}_d + \left(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J}\right) \boldsymbol{\xi}_1 \\ \ddot{\mathbf{q}}_d &= \mathbf{J}^\dagger \left(\ddot{\mathbf{x}}_d - \dot{\mathbf{J}}\dot{\mathbf{q}}\right) + \left(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J}\right) \boldsymbol{\xi}_2 \end{aligned}$$

where $\mathbf{J}^\dagger$ is the pseudoinverse defined by $\mathbf{J}^\dagger = \mathbf{J}^T \left(\mathbf{J}\mathbf{J}^T\right)^{-1}$, ξ_1 and ξ_2 are arbitrary vectors, and

$\dot{\mathbf{x}}_d$ and $\ddot{\mathbf{x}}_d$ are the given desired task space velocities and accelerations, respectively. $\left(\mathbf{I}-\mathbf{J}^{\dagger}\mathbf{J}\right)$ projects arbitrary vectors ξ_1 and ξ_2 onto the null space of the Jacobian J. Note that by equating and the analytical differentiation, sufficient conditions for these two resolution schemes to be consistent are derived, respectively, as

$$\boldsymbol{\xi}_2 = \dot{\mathbf{J}}^{\dagger}\mathbf{J}\left(\dot{\mathbf{q}}-\boldsymbol{\xi}_1\right)+\dot{\xi}_1$$

and

$$\xi_2 = \dot{\mathbf{J}}^{T}\left(\mathbf{J}^{\dagger}\right)^{T}\left(\dot{\mathbf{q}}-\boldsymbol{\xi}_1\right)+\dot{\xi}_1$$

where $\mathbf{J}^{\dagger}=\frac{d}{dt}\left(\mathbf{J}^{\dagger}\right)$ for notational convenience. In the torque based redundancy resolution, the inertia weighted pseudoinverse $\overline{\mathbf{J}}=\mathbf{M}^{-1}\mathbf{J}^{T}\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}\right)^{-1}$ is used to compute the desired joint torque command exploiting the dynamical decoupling property of the joint torques that create task space forces, and the joint torques that act only in the null space of the task.[8]

2.4 Nullspace Control and Task Prioritization

Nullspace control involves controlling the robot in a subspace that is orthogonal to the space of the primary task, while still maintaining the primary task objective. This approach allows the robot to achieve additional objectives or constraints, such as obstacle avoidance or joint limit avoidance, without interfering with the primary task. Task prioritization, on the other hand, involves assigning priority levels to different tasks or objectives, and then controlling the robot to achieve the objectives in order of priority. This approach allows the robot to perform multiple tasks simultaneously while still ensuring that the most important tasks are completed first. By combining nullspace control and task prioritization, robots can perform complex tasks with multiple objectives and constraints, such as manipulation or grasping tasks. For example, in a manipulation task, the robot might have a primary task of grasping an object, but also needs to avoid obstacles and joint limits. By using nullspace control, the robot can avoid obstacles and joint limits while still achieving the primary task of grasping the object. Task prioritization can then be used to ensure that the robot completes the primary task first before moving on to the secondary tasks of avoiding obstacles and joint limits.

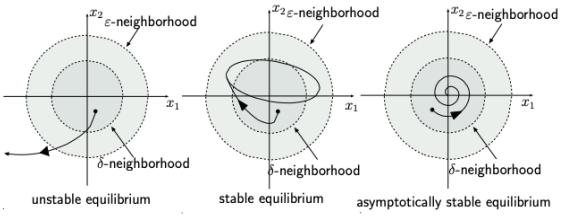


Figure 6: Lyapunov-stability

3. Modern Methods of Robot Control

3.1 Definition of Passivity

The system S with $m=q$ (equivalent number of input and output variables) is called passive if it is dissipative with respect to the special supply rate

$$s(\underline{u},\underline{y}) = \underline{y}^{T}\underline{u}.$$

In other words, there exists a positive semidefinite function $V(\underline{x})$, such that the integral passivity inequality

$$\int_0^t \underline{y}^{T}\underline{u}d\tau + V(\underline{x}(0)) \geq V(\underline{x}(t)), \quad \forall t \geq 0$$

holds for all initial values $\underline{x}_0$ and all input variables $\underline{u}(t)$.

3.2 Stability and Passivity – Lyapunov Stability

An equilibrium point $\underline{x}^{*}=\underline{0}$ of the system is - stable in the sense of Lyapunov (i.S.o.L) if for every $\varepsilon>0$, there exists a $\delta(\varepsilon,t_0)>0$ such that

$$\left\|\underline{x}(t_0)\right\| < \delta \Rightarrow \|\underline{x}(t)\| < \varepsilon, \forall t \geq t_0.$$

- - asymptotically stable in the sense of Lyapunov (i.S.o.L) if $\underline{x}^{*}$ is stable and $\delta(\varepsilon,t_0)>0$ is chosen in a way such that

$$\left\|\underline{x}(t_0)\right\| < \delta \Rightarrow \lim_{t\to\infty}\|\underline{x}(t)\| = 0.$$

- - uniformly stable in the sense of Lyapunov (i.S.o.L) if for every $\varepsilon>0$ there exists a $\delta(\varepsilon)>0$ independent of t_0 such that

$$\left\|\underline{x}(t_0)\right\| < \delta \Rightarrow \|\underline{x}(t)\| < \varepsilon, \forall t \geq t_0.$$

- - uniformly asymptotically stable in the sense of Lyapunov (i.S.o.L), if $\underline{x}^{*}$ is uniformly stable and $\delta(\varepsilon)>0$ is independent of t_0, such that $\left\|\underline{x}(t_0)\right\| < \delta \Rightarrow \lim_{t\to\infty}\|\underline{x}(t)\| = 0$ and $\underline{x}(t)$ converges uniformly in t_0. Uniform convergence of a trajectory $\underline{x}(t)$ in t_0 means that

there exists a time interval $T(R_1, R_2)$ for every pair R_1, R_2 with $0 < R_2 < R_1$ independent of t_0, such that from $\left\|\underline{x}(t_0)\right\| < R_1$ for all time $t \geq t_0 + T(R_1, R_2)$ it follows that $\|\underline{x}(t)\| < R_2$ holds.

- - unstable if $\underline{x}^*$ is not stable.

[6]

3.3 Passive Representation of a Robot

The passive representation of a robot refers to a mathematical model of the robot that characterizes the robot's mechanical behavior in response to external forces and torques applied to the robot's joints. It has the following features:

- It characterizes the robot's mass, inertia, and damping properties, as well as the relationship between the robot's joint torques and the resulting joint angles and velocities.
- It is distinct from the active representation, which describes how the robot's control system generates joint torques to achieve a desired motion or task.
- The passive and active representations are both important in robotics, as they provide a means to model and control the robot's behavior in different contexts.
- The passive representation is particularly useful for tasks such as trajectory planning, where a detailed understanding of the robot's mechanical properties is necessary to generate accurate and effective control commands.

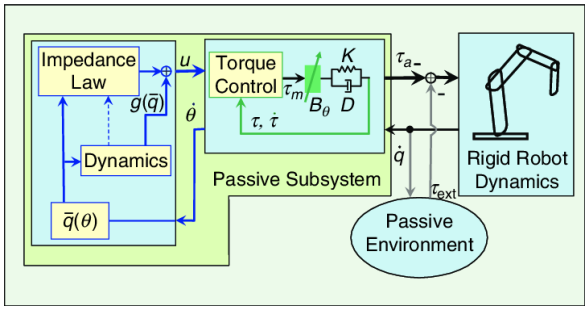


Figure 7: Papline of Passive Representation

3.4 Control Algorithm

3.4.1 Velocity based Control

Given the velocities and positions of the target in task space, we define the task space velocity command $\dot{\mathbf{x}}_r$ from the addion of a position error term with a target velocity feedforward term as

$$\dot{\mathbf{x}}_r = \dot{\mathbf{x}}_d + \mathbf{K}_p (\mathbf{x}_d - \mathbf{x})$$

where $\mathbf{K}_p$ is a positive definite gain matrix. Note that this scheme does not use the target acceleration information $\ddot{\mathbf{x}}_d$. The reference joint velocities are calculated based on a pseudoinverse solution with minimization of the cost function in the null space

$$\dot{\mathbf{q}}_r = \mathbf{J}^\dagger \dot{\mathbf{x}}_r - \left(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J}\right) \alpha \nabla g.$$

The null-space vector is chosen to be the negative gradient of the cost function as $\xi_1 = -\alpha \nabla g = -\alpha \left(\frac{\partial q}{\partial q}\right)^T$ where α is a positive constant. The reference joint accelerations and positions are obtained by numerical differentiation and integration of the reference joint velocities (11), respectively as

$$\ddot{\mathbf{q}}_r = \frac{d}{dt}\dot{\mathbf{q}}_r \simeq \frac{\dot{\mathbf{q}}_r(t) - \dot{\mathbf{q}}_r(t - \Delta t)}{\Delta t}$$
$$\mathbf{q}_r = \int_{t_0}^{t} \dot{\mathbf{q}}_r dt' \simeq \mathbf{q}_r(t - \Delta t) + \dot{\mathbf{q}}_r(t)\Delta t$$

where Δt is the sampling period. The final control signal is calculated using the computed torque control method with a PD controller as

$$\begin{aligned}\boldsymbol{\tau} =& \mathbf{M}(\mathbf{q}_r)\ddot{\mathbf{q}}_r + \mathbf{C}(\mathbf{q}_r, \dot{\mathbf{q}}_r) + \mathbf{g}(\mathbf{q}_r) \\ &+ \mathbf{K}_{q,d}(\dot{\mathbf{q}}_r - \dot{\mathbf{q}}) + \mathbf{K}_{q,p}(\mathbf{q}_r - \mathbf{q})\end{aligned}$$

where $\mathbf{K}_{q,d}$ and $\mathbf{K}_{q,p}$ are positive definite gain matrices.

3.4.2 Velocity-based Control without Joint Velocity Integration

Task space velocity and the reference joint velocities obtained from the Liegeois' null-space optimization,

$$\dot{\mathbf{x}}_r = \dot{\mathbf{x}}_d + \mathbf{K}_p (\mathbf{x}_d - \mathbf{x})$$
$$\dot{\mathbf{q}}r = \mathbf{J}^\dagger \dot{\mathbf{x}}_r - \alpha \left(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J}\right) \nabla g.$$

An inverse dynamics control law can be formulated with the addition of a joint velocity feedback term as

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_r + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \mathbf{K}_{q,d}(\dot{\mathbf{q}}_r - \dot{\mathbf{q}})$$

$$\begin{aligned}\boldsymbol{\tau} =& \mathbf{M}\ddot{\mathbf{q}}_r + \mathbf{C} + \mathbf{g} + \mathbf{K}_{q,d}(\dot{\mathbf{q}}_r - \dot{\mathbf{q}}) \\ =& \mathbf{M}\left(\ddot{\mathbf{q}}_d + \mathbf{J}^\dagger \mathbf{K}_p \dot{\mathbf{e}}\right) + \mathbf{C} + \mathbf{g} + \mathbf{K}_{q,d}(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) \\ &+ \mathbf{K}_{q,d}\mathbf{J}^\dagger \mathbf{K}_p \mathbf{e}\end{aligned}$$

with $\xi_1 = -\alpha \nabla g$, where $\mathbf{e} = \mathbf{x}_d - \mathbf{x}$.

3.4.3 Acceleration based Control

In this method, the control law is given by where

$$\ddot{\mathbf{x}}_r = \ddot{\mathbf{x}}_d + \mathbf{K}_d \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e}$$

3.4.4 *Torque based Control*

The dynamics which describe the motion of the task space coordinates are given by

$$\overline{\mathbf{M}}(\mathbf{x})\ddot{\mathbf{x}} + \overline{\mathbf{C}}(\mathbf{x}, \dot{\mathbf{x}}) + \overline{\mathbf{g}}(\mathbf{x}) = \mathbf{F}$$

where

$$\overline{\mathbf{M}} = \left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}$$
$$\overline{\mathbf{C}} = \left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{C} - \dot{\mathbf{J}}\dot{\mathbf{q}}\right)$$
$$\overline{\mathbf{g}} = \left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}\mathbf{J}\mathbf{M}^{-1}\mathbf{g}.$$

From these equations, a control law for the task space dynamics is designed to achieve asymptotic tracking of the desired end-effector trajectory $\mathbf{x}_d$ as

$$\mathbf{F} = \overline{\mathbf{M}}\ddot{\mathbf{x}}_r + \overline{\mathbf{C}} + \overline{\mathbf{g}}$$

where

$$\ddot{\mathbf{x}}_r = \ddot{\mathbf{x}}_d + \mathbf{K}_p\left(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}\right) + \mathbf{K}_p\left(\mathbf{x}_d - \mathbf{x}\right).$$

$$\ddot{\mathbf{e}} + \mathbf{K}_p\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e} = 0$$

3.5 Controller Example

3.5.1 *Open-Loop Controller*

In the beginning, we implemented an open-loop version of the computed torque controller and tested the impact of friction on the system. The function of the controller is given as while the friction is a constant term acting against the motion. Therefore the system equation is

$$\ddot{\underline{l}} = M(\underline{l})^{-1}\left(\tau_l - C(\underline{l}, \dot{\underline{l}})\dot{\underline{l}} - G(\underline{l})\right) - K\,\mathrm{sign}(\dot{\underline{l}}),$$

where K is a constant, chosen to be 0.4 N.

3.5.2 *Closed-Loop Controller*

To counteract deviations in motion created by unaccounted forces and dampen the system, we implemented a closed-loop version of the computed torque controller. By adding a PD controller to the computed torque controller. The controller function is extended to

$$\tau_l = M(\underline{l})\ddot{\underline{l}}_d + C(\underline{l}, \dot{\underline{l}})\dot{\underline{l}} + G(\underline{l}) + K_P\left(\underline{l}_d - \underline{l}\right) + K_D\left(\dot{\underline{l}}_d - \dot{\underline{j}}\right),$$

Once we add friction to the system, the open-loop system outputs are not able to follow the system inputs anymore. This can be seen in Fig 3(c). On the other side, the closed loop controller can balance out the friction and the system outputs follow the inputs, which can be seen in (d). Therefore the closed-loop system is stable and can reject small forces generated by friction.[7]

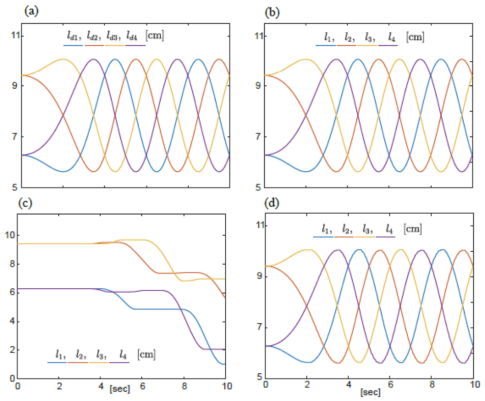


Figure 8: (a) Desired Trajectory, (b) System output of open loop controller with no friction, (c) System output of open loop controller with friction, (d) System output of closed-loop controller with friction.

3.6 Joint Control Summary – Performance vs. Robustness

Joint control is an important aspect of robotics, as it allows the robot to move its joints in a controlled manner in order to achieve various tasks. There are several different types of joint controllers, each with its own trade-offs between performance and robustness. Here's a summary of some common types of joint controllers:

- PD controller: This is a simple joint controller that uses proportional and derivative feedback to regulate the position and velocity of the joints. The PD controller is easy to implement and can provide good performance, but is not robust to changes in the robot's environment or model uncertainties.
- PD+ controller: This is an extension of the PD controller that adds a feedforward term to the controller to compensate for model uncertainties. The PD+ controller can provide improved performance over the PD controller, while still being relatively easy to implement.
- Computed torque controller: This is a more sophisticated joint controller that uses a mathematical model of the robot to calculate the desired joint torques necessary to achieve a desired motion or task. The computed torque controller can provide excellent performance and robustness to changes in the robot's environment or model uncertainties but is more complex to implement than simpler controllers like the PD controller.

4. Modern Methods of Robot Control II

4.1 Impedance/Admittance Control

Impedance and admittance control are two related methods for controlling the behavior of robots in contact with their environment. Here's a summary of each method:

- Impedance control: This method regulates the interaction between the robot and its environment by controlling the impedance of the robot's end-effector. The impedance represents the robot's resistance to motion when in contact with the environment, and is controlled using a combination of position and force feedback. The goal of impedance control is to provide a compliant behavior that allows the robot to adapt to the environment and avoid damage or injury. Impedance control is commonly used in applications such as force sensing and assembly tasks.
- Admittance control: This method regulates the interaction between the robot and its environment by controlling the admittance of the robot's end-effector. The admittance represents the robot's responsiveness to external forces, and is controlled using a combination of position and force feedback. The goal of admittance control is to provide a responsive behavior that allows the robot to adapt to external forces and disturbances. Admittance control is commonly used in applications such as teleoperation and haptic feedback.

Both impedance and admittance control are useful methods for controlling robots in contact with their environment. The choice of method depends on the specific application and the desired behavior of the robot. In general, impedance control is better suited for tasks where the robot needs to apply force to the environment, while admittance control is better suited for tasks where the robot needs to respond to external forces.

4.2 Port-Hamiltonian Systems in Robotics

The port-Hamiltonian framework is a methodology for modeling physical systems based on a known energy function, known as the Hamiltonian ($\mathcal{H}$). This framework provides an energy-consistent description of a system and is particularly useful for modeling complex, interconnected systems. Each individual element in the system is defined by its own Hamiltonian energy function and interacts with other elements by exchanging energy through a "port". These ports are described by a pair of variables, referred to as efforts (e.f) and flows, that represent the input and output of the system respectively. The product of these two variables is referred to as power. A port-Hamiltonian system is described in an input-state-output form, and its behavior is defined by the Hamiltonian function is:

$$\dot{\boldsymbol{x}} = [J(\boldsymbol{x}) - D(\boldsymbol{x})]\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}(\boldsymbol{x}) + G(\boldsymbol{x})\boldsymbol{u}$$
$$\boldsymbol{y} = G^T(\boldsymbol{x})\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}(\boldsymbol{x}),$$

where $J(\boldsymbol{x})$ is a skew-symmetric structure matrix and $D(\boldsymbol{x}) \succeq 0$ is a symmetric dissipation matrix. Inputs to the system are flows, $\boldsymbol{u}$, and outputs are efforts $\boldsymbol{y}$. The matrix $G(\boldsymbol{x})$ is a mapping matrix and $\boldsymbol{x}$ is the state vector of the system. The Hamiltonian of the system is a positive, semidefinite function $\mathcal{H} \geq 0$, which represents the total energy stored in the system. The rate of the energy change of is:

$$\dot{\mathcal{H}} = \boldsymbol{y}^T\boldsymbol{u} - \frac{\partial^T \mathcal{H}}{\partial \boldsymbol{x}} D(\boldsymbol{x}) \frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}$$

Every port-Hamiltonian system is passive.

And Port-based modeling includes:

- Ports: In a port-based model, ports represent the interfaces through which a system exchanges energy, mass, and/or information with its environment. For a manipulator, these ports might include the robot's joints, actuators, and sensors, as well as any external forces or torques acting on the robot.
- Bond graph: A bond graph is a graphical representation of the energy and information flows between different ports in a system. In the context of a manipulator, a bond graph might represent the flow of energy from the robot's actuators to its joints, the flow of force from the environment to the robot's end-effector, and the flow of position and velocity information from the robot's sensors to its controllers.
- Port-Hamiltonian system: A Port-Hamiltonian system is a type of dynamic system that describes the energy and information flows between different ports in a system. In the context of a manipulator, a Port-Hamiltonian system might describe the dynamics of the robot's joints and actuators, as well as the forces and torques acting on the robot from the environment.

By using port-based modeling to describe the behavior of a manipulator, engineers and researchers

can design controllers that can regulate the energy and information flows within the robot, and thus achieve desired behaviors such as stable motion, force control, or energy efficiency. This framework provides a powerful tool for modeling and controlling the behavior of manipulators, and is widely used in the field of robotics.

5. Linear Parametrization and Identification of Robot Dynamics

Linear parametrization of a manipulator's dynamic model involves representing the dynamic behavior of the manipulator as a linear combination of its physical parameters. These parameters may include the mass, center of mass, and inertia of the manipulator's links, as well as the friction and damping coefficients of its joints. The resulting model is often referred to as the Linear Parametrized Dynamic Model (LPDM), and it is a fundamental tool for controlling the behavior of the manipulator. Both the kinetic energy and the gravity potential energy can be rewritten so that a new set of dynamic parameters appears only in a linear way

5.1 fundamental kinematic relation

$v_{ci} = v_i + \omega_i \times r_{ci} = v_i + S(\omega_i) r_{ci} = v_i - S(r_{ci}) \omega_i$

kinetic energy of link i

$$
\begin{aligned}
T_i &= \frac{1}{2} m_i v_{ci}^T v_{ci} + \frac{1}{2} \omega_i^T I_{ci} \omega_i \\
&= \frac{1}{2} m_i \left(v_i - S(r_{ci})\omega_i\right)^T \left(v_i - S(r_{ci})\omega_i\right) + \frac{1}{2} \omega_i^T I_{ci} \omega_i \\
&= \frac{1}{2} m_i v_i^T v_i + \frac{1}{2} \omega_i^T \left(I_{ci} + m_i S^T(r_{ci}) S(r_{ci})\right) \omega_i - v_i^T S(m_i r_{ci}) \omega_i
\end{aligned}
$$

Steiner theorem $\longrightarrow I_i = \begin{pmatrix} I_{i,xx} & I_{i,xy} & I_{i,xz} \\ & I_{i,yy} & I_{i,yz} \\ \text{symm} & & I_{i,zz} \end{pmatrix}$

5.2 gravity potential energy

the gravitational potential energy of link i

$$U_i = -m_i g_0^T r_i - g_0^T (m_i r_{ci})$$

5.3 Overall Energies and Robot Regressor Formulation

using a $N \times 10N$ regression matrix Y_π that depends only on kinematic quantities, the robot dynamic equations can always be rewritten linearly in the standard dynamic parameters as

$$M(q)\ddot{q} + c(q,\dot{q}) + g(q) = Y_\pi(q,\dot{q},\ddot{q})\pi = u$$

$$\pi^T = \begin{pmatrix} \pi_1^T & \pi_2^T & \cdots & \pi_N^T \end{pmatrix}$$

the open kinematic chain structure of the manipulator implies that the i-th dynamic equation can depend only on the dynamic parameters of links from i to $N \Rightarrow Y_\pi$ has a block upper triangular structure

$$
\boldsymbol{Y}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \begin{bmatrix}
\boldsymbol{Y}_{1,1} & \boldsymbol{Y}_{1,2} & \boldsymbol{Y}_{1,3} & \cdots & \boldsymbol{Y}_{1,n-1} & \boldsymbol{Y}_{1,n} \\
\boldsymbol{0} & \boldsymbol{Y}_{2,2} & \boldsymbol{Y}_{2,3} & \cdots & \boldsymbol{Y}_{2,n-1} & \boldsymbol{Y}_{2,n} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Y}_{3,3} & \cdots & \boldsymbol{Y}_{3,n-1} & \boldsymbol{Y}_{3,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{Y}_{n-1,n-1} & \boldsymbol{Y}_{n-1,n} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{Y}_{n,n}
\end{bmatrix}
$$

Property: element m_{ij} of $M(q)$ depends at most on $(q_{k+1}, \cdots, q_N)$, with $k = \min\{i,j\}$, and on the dynamic parameters of at most links h to N, with $h = \max\{i,j\}$

In summary, linear parametrization of a manipulator's dynamic model involves expressing the dynamic behavior of the manipulator as a linear combination of its physical parameters, and results in a Linear Parametrized Dynamic Model (LPDM) that can be used for control design. The LPDM is a powerful tool for controlling the behavior of manipulators, and is widely used in the field of robotics.

5.4 Identification of Robot Dynamics

Identification of robot dynamics can be approached from various aspects, including:

- Parameter identification: This involves estimating the values of the physical parameters that govern the robot's behavior, such as the mass, inertia, and friction coefficients. This is typically done by performing experiments and using data-driven methods to estimate the parameters, such as the least-squares method or the maximum likelihood method.
- Trajectory optimization: This involves finding optimal trajectories for the robot to follow, taking into account its dynamic model and physical limitations. This can be done using numerical optimization techniques, such as gradient-based methods or trajectory optimization methods.
- Modeling of friction: Friction plays an important role in the dynamics of robotic systems, and its accurate modeling is critical for control design. Different approaches can be used to model friction, such as the Coulomb friction model or the viscous friction model.
- Identification procedure: The process of identifying robot dynamics typically involves a

number of steps, such as data collection, preprocessing, parameter estimation, and model validation. The specific procedure may vary depending on the application and the available data.

- Adaptive control: Adaptive control is a control technique that uses online parameter estimation to continuously update the controller parameters in response to changes in the system dynamics. This approach can be used to compensate for modeling errors or uncertainties in the robot's dynamics, and can improve the performance and robustness of the control system. Identification of robot dynamics is a critical component of adaptive control, as it provides the necessary information for online parameter estimation.

6. Bio-Inspired Robot Control

One way to enhance the adaptability of robots is through the implementation of bio-inspired learning algorithms that emulate the methods used by living organisms to adapt to new situations. [3] The capability for adaptation to new dynamics is a crucial area of study in robotics, as it holds the potential to transform the field by enabling robots to carry out tasks that are currently not possible. [4]

6.1 Bio-Inspired Formulation

Bio-inspired formulation refers to the development of robotic control strategies that are inspired by biological systems, such as animals and humans. Some key areas of research in bio-inspired control include:

- Adaptation to novel dynamics: Biological systems are able to adapt to changes in their environment and in the dynamics of their bodies. Robotic systems can benefit from similar adaptive capabilities, which can be achieved through machine learning and other techniques.

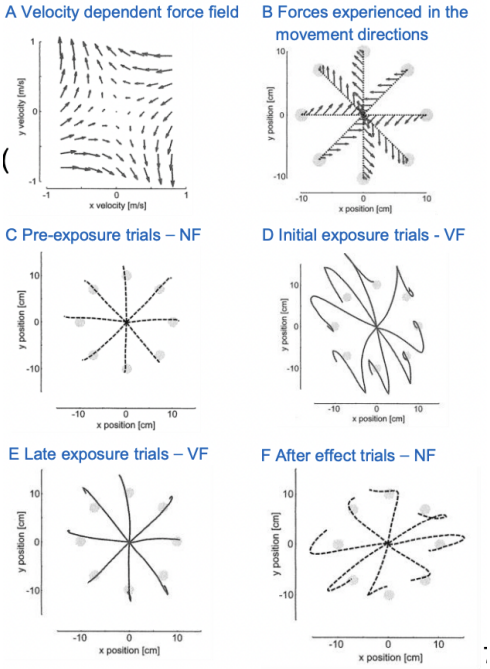
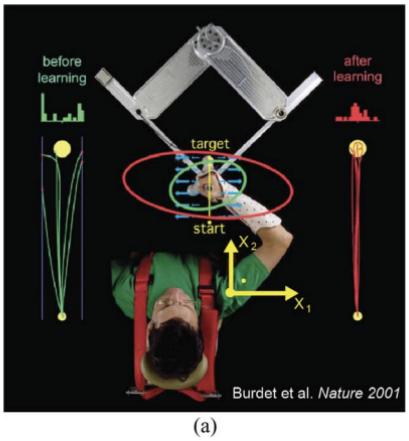


Figure 9: Adaption to Novel Dynamics

- Motor learning under unstable and unpredictable conditions: Biological systems are able to learn and refine their motor skills even under challenging conditions, such as unstable surfaces or unexpected disturbances. Robotic systems can also benefit from such learning capabilities, which can be achieved through reinforcement learning, imitation learning, or other approaches.



(a)

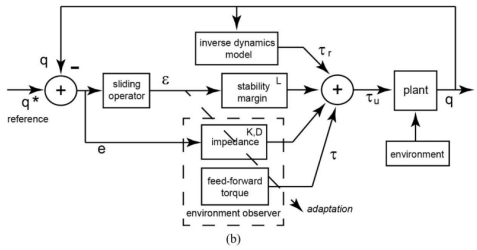


Figure 10: (a) Investigation of point-to-point arm movements with lateral instability. (b) Control diagram of the biomimetic controller that is presented in this paper and was derived from the results of this investigation.

[9]

- Adapted impedance: The impedance of a robotic system refers to its response to external forces or disturbances. Biological systems are able to adjust their impedance in response to changing task demands or environmental conditions. Robotic systems can also benefit from adapted impedance, which can improve their stability, safety, and efficiency.
- Change of impedance and force during learning: As biological systems learn new skills, they may change their impedance and force profiles to optimize their performance. Robotic systems can also benefit from similar adaptive strategies, which can be achieved through online learning and control.
- Error-based learning algorithm: Biological systems often use error-based learning algorithms to refine their motor skills and adapt to new situations. These algorithms involve comparing the desired output with the actual output and adjusting the control parameters to reduce the error. Robotic systems can also benefit from similar error-based learning algorithms, which can be used to improve their performance and adapt to changing conditions.

Overall, bio-inspired formulation provides a promising approach to developing robotic control strategies that are more flexible, adaptive, and robust than traditional approaches. By taking inspiration from the biological world, researchers can develop control strategies that are better suited to the complex and dynamic environments in which robots are increasingly being deployed.

6.2 Adaptive Impedance Control for a Manipulator

Adaptive impedance control for a manipulator is a type of control strategy that allows the robot to adapt its impedance (i.e., its response to external forces and disturbances) in real time. This is achieved through an adaptive control algorithm that continuously monitors the robot's interactions with the environment and adjusts the impedance parameters accordingly. The goal of adaptive impedance control is to improve the robot's performance and safety by allowing it to respond more effectively to changing task demands and environmental conditions. For example, the robot can adjust its impedance to be more compliant when interacting with delicate objects or to be more rigid when performing tasks that require high precision. The adaptive impedance control algorithm typically involves a feedback loop that compares the desired and actual robot behavior and adjusts the impedance parameters based on the error. The algorithm may also incorporate machine learning techniques to improve the adaptation process over time. Overall, adaptive impedance control is a promising approach to developing robotic control strategies that are more flexible and responsive to the dynamic environments in which robots are increasingly being deployed.

7. ML in Robotics

7.1 Neural Networks

Neural networks are a type of artificial intelligence system modeled after the structure and function of the human brain. They are widely used in various applications, including image recognition, natural language processing, and control systems for robots. Neural networks are particularly effective for tasks that require pattern recognition and classification, as they can learn and identify complex relationships between inputs and outputs. In the field of robotics, neural networks are utilized to develop advanced control systems that allow robots to respond to environmental stimuli and operate more efficiently.

This is achieved by using neural networks to process sensory information and generate control signals that drive the robot's actuators. The network is trained using a large dataset of examples, and it continuously improves its performance over time.

One key advantage of using neural networks in robotics is their ability to learn from experience and adapt to changing conditions. This is crucial in real-world applications where robots must operate in unpredictable and dynamic environments. For example, robots equipped with neural networks can adapt to different terrains, obstacles, and other factors that may affect their movements, allowing them to perform tasks that are difficult or impossible for traditional robots.

Another application of neural networks in robotics is in the development of bio-inspired robots, which aim to replicate the behaviors and movements of living organisms. This is achieved by incorporating biologically-inspired features, such as sensory feedback, into the robot's control system. The robot's neural network then processes this information and generates control signals that drive its actuators, enabling it to respond to environmental stimuli in a manner similar to living organisms.
In conclusion, the integration of neural networks in robotics has the potential to revolutionize the field by enabling robots to perform tasks that are beyond their current capabilities. By incorporating advanced AI algorithms, robots can adapt to changing environments, navigate complex terrains, and perform delicate manipulations, leading to new possibilities and applications for these systems in various industries.

7.2 3 Categories

Machine Learning can be divided into three main categories of problems

7.2.1 Supervised Learning

Supervised learning is a type of machine learning that trains a model to make predictions based on labeled data. In robotics, this approach is used for tasks such as object recognition, scene understanding, and control.
Supervised learning is beneficial for robots operating in dynamic environments as they can learn from examples and improve their performance over time.
For example, a robot can learn to perform a task by observing a human demonstration and continuously improving its performance through training. Another application of supervised learning in robotics is in the development of autonomous robots, allowing them to perform tasks such as navigation, grasping, and manipulation more efficiently. The use of supervised learning in robotics has the potential to revolutionize the field and lead to new possibilities for these systems.

7.2.2 Unsupervised Learning

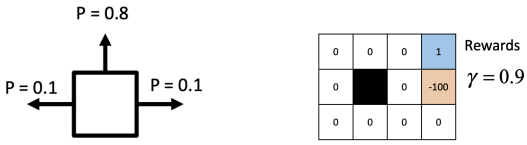
Unsupervised learning is a type of machine learning where the model is not provided with labeled data, but instead must find patterns and relationships in the data on its own. In robotics, unsupervised learning can be used for tasks such as dimensionality reduction, clustering, and anomaly detection.
One application of unsupervised learning in robotics is in the development of self-organizing maps, which can be used for visual recognition and navigation. In these systems, the robot uses unsupervised learning to map its environment and recognize objects and obstacles.
Another application of unsupervised learning in robotics is in the development of adaptive controllers, which can be used to improve the performance of robots in dynamic environments. By using unsupervised learning, robots can learn and adapt to new dynamics, allowing them to operate more effectively and efficiently in real-world settings.
In conclusion, unsupervised learning is an important tool in robotics, as it allows robots to learn from data without the need for labeled examples. This approach can be used for tasks such as visual recognition, navigation, and adaptive control, and has the potential to revolutionize the field and lead to new possibilities for these systems.

7.2.3 Reinforcement Learning

Reinforcement learning is a type of machine learning that trains a model to make decisions by learning from experiences. In this approach, the model receives rewards for certain actions and learns to optimize its behavior over time to maximize the reward. In robotics, reinforcement learning is used for tasks such as autonomous navigation, grasping, and manipulation.
One application of reinforcement learning in robotics is in the development of autonomous robots, which must be able to make decisions and take actions based on the state of their environment. By using reinforcement learning, robots can learn to perform tasks such as navigation and grasping by receiving rewards for successful actions and adjusting their behavior accordingly.
Another application of reinforcement learning in robotics is in the development of control systems, where the model can learn to control the robot's movements to achieve a desired goal. For example, a robot equipped with a reinforcement learning system can learn to balance on a moving platform by continuously adjusting its behavior based on the rewards it receives.
In conclusion, reinforcement learning is an important tool in robotics, allowing robots to learn and make decisions based on experiences. By incorporating reinforcement learning into their control systems, robots can adapt to changing conditions and perform tasks that are beyond their current capabilities. The use of reinforcement learning in robotics has the potential to revolutionize the field and lead to new possibilities for these systems. [5]

7.3 Optimal Control- Bellman's Equation

$$Q_i(x,u) = R\left(x_i,u_i\right) + \mathbb{E}_e\left[\max_{u'} Q_{i+1}\left(f\left(x_i,u_i,e_i\right),u'\right)\right]$$



- tool for solving optimal value problems
- the basis of policy iteration in RL
- It allows us to express the value of one state as the value of another state[6]
- the dynamic optimization problem is turned into simple sub-problems
- simplify the reinforcement learning or Markov decision problems. Just make complex problems simple!

Linear equations, theoretically the solution can be solved:

$$v = \mathcal{R} + \gamma \mathcal{P} v$$
$$(I - \gamma \mathcal{P}) v = \mathcal{R}$$
$$v = (I - \gamma \mathcal{P})^{-1} \mathcal{R}$$

- complexity is $O\left(n^3\right)$
- only suitable for small-scale MRPS.
- large-scale MRP usually needs to use other iterative methods.

Markov Decision Processes (MDP)

- Elements
- MDP problem
- Optimal Policy

Dynamic Programming (DP)

- Solving MDP using DP
- Policy Iteration
- Value Iteration

$V_{k+1}(s) = \max_a \left[r(s,a) + \gamma \sum_{s'} p\left(s' \mid s, a\right) V_k\left(s'\right) \right]$v

- immediate reward $r(s,a)$
- discounted value of successor state $\gamma \sum_{s'} p\left(s' \mid s, a\right) V_k\left(s'\right)$

Selected direction with probability 0.8 and in the perpendicular directions with prob. 0.1 and the rewords discount is 0.9[1]

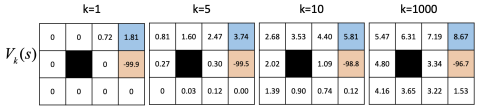


Figure 11: Example

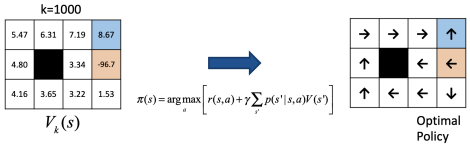


Figure 12: Final Result

7.4 Examples

7.4.1 Supervised Learning in Robotics (Computer Vision)

Figure 13: Autonomous Driving (Tesla)

Figure 14: space exploration (NASA)

Figure 15: Humanoid Locomotion (Boston Dynamics)

7.5 Reinforcement Learning in Robotics

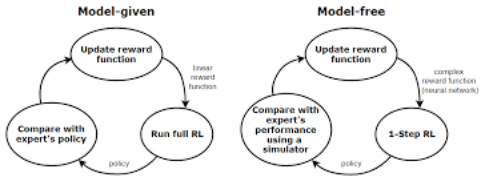


Figure 16: Imitation Learning

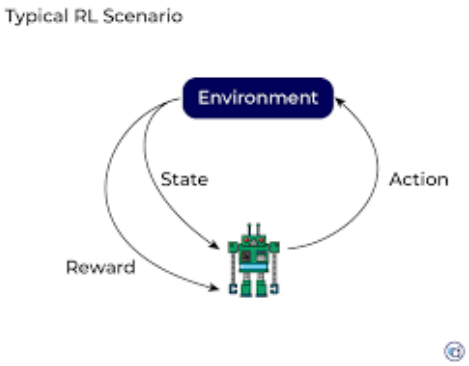


Figure 17: Reinforcement learning

8. Overview of Robot Perception

Robot perception refers to the process by which a robot collects, processes, and interprets sensory data from its environment in order to make informed decisions about its surroundings. This is a crucial aspect of robotics, as it enables robots to interact with their environment and perform tasks more effectively. There are several key components to robot perception, including:

- Sensors: Robots use various sensors, such as cameras, depth sensors, LIDAR, and microphones, to gather information about their environment. The type of sensor used will depend on the task the robot is performing and the information it needs to gather.
- Signal Processing: Once the data is collected by the sensors, it must be processed in order to extract useful information. This involves filtering the raw data, removing noise, and transforming the data into a format that can be used by the robot's control system.
- Feature Extraction: The processed data is then analyzed to extract features that are relevant to the task the robot is performing. These features may include the position of objects, their shape, size, and color, and any other relevant information.
- Object Recognition: The extracted features are used to identify objects in the environment and categorize them. This may involve comparing the features to a database of known objects, or using machine learning algorithms to identify patterns in the data.
- Scene Understanding: The final step in robot perception is to build a comprehensive understanding of the scene, including the relationships between objects and their interactions with each other. This information is then used by the robot's control system to make informed decisions about how to interact with the environment.

In conclusion, robot perception is a complex process that involves several components, including sensors, signal processing, feature extraction, object recognition, and scene understanding. These components work together to provide the robot with a detailed understanding of its environment, allowing it to make informed decisions and interact more effectively with its surroundings.

Reader Response

Advanced robot control and learning is a rapidly evolving field that holds great promise for revolutionizing the way robots operate and interact with their environment. The integration of technologies, such as the design, implementation and analysis of control algorithms and robot kinematics and dynamics, and machine learning technology like artificial intelligence, computer vision. I learn about various control techniques, such as PID control and adaptive control, and how they can be applied to robotic systems to improve stability, accuracy, and efficiency. Reinforcement learning, a type of machine learning, is also introduced as a method for robots to learn from experience and adapt to their environment. And I have an understanding of how to design, implement, and analyze control algorithms and learning techniques for robots. This knowledge can also be applied to a wide range of applications, including autonomous vehicles, industrial automation, and service robots. The technology allows robots to perform more efficiently and effectively in complex and dynamic environments. The continued development of this field will likely result in robots that are better equipped to handle a wide range of tasks and environments, and that can operate with greater autonomy and adaptability. As technology continues to advance,

it will likely play an increasingly important role in many areas of society, including manufacturing, healthcare, and transportation, to name just a few. Overall, the future of advanced robot control and learning is exciting and holds immense potential for improving the quality of life for people all around the world.

References

[1]R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, «Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction», Autonomous Robots **43**, 10.1007/s10514-018-9706-9 (2019) 10.1007/s10514-018-9706-9.

[2]S. Calinon, «Gaussians on riemannian manifolds: applications for robot learning and adaptive control», IEEE Robotics Automation Magazine **27**, 33–45 (2020) 10.1109/MRA.2020.2980548.

[3]S. Haddadin and E. Croft, «Erratum to: physical human–robot interaction», in *Springer handbook of robotics*, edited by B. Siciliano and O. Khatib (Springer International Publishing, Cham, 2016), E1–E1, ISBN: 978-3-319-32552-1, 10.1007/978-3-319-32552-1_81.

[4]S. Haddadin, A. De Luca, and A. Albu-Schäffer, «Robot collisions: a survey on detection, isolation, and identification», IEEE Transactions on Robotics **33**, 1292–1312 (2017) 10.1109/TRO.2017.2723903.

[5]S. Haddadin, S. Haddadin, A. Khoury, T. Rokahr, S. Parusel, R. Burgkart, A. Bicchi, and A. Albu-Schäffer, «On making robots understand safety: embedding injury knowledge into control», The International Journal of Robotics Research **31**, 1578–1602 (2012) 10.1177/0278364912462256.

[6]T. Liu, F. Liu, and M. Buss, «Indirect adaptive control of piecewise affine systems without common lyapunov functions», IEEE Control Systems Letters **6**, 1808–1813 (2022) 10.1109/LCSYS.2021.3133800.

[7]J. Nassour, P. Hénaff, F. B. Ouezdou, and G. Cheng, «Experience-based learning mechanism for neural controller adaptation: application to walking biped robots», in 2009 ieee/rsj international conference on intelligent robots and systems (2009), pp. 2616–2621, 10.1109/IROS.2009.5354797.

[8]J. Nassour, V. Hugel, F. Ouezdou, and G. Cheng, «Qualitative adaptive reward learning with success failure maps: applied to humanoid robot walking», Neural Networks and Learning Systems, IEEE Transactions on **24**, 10.1109/TNNLS.2012.2224370 (2012) 10.1109/TNNLS.2012.2224370.

[9]C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schäeffer, and E. Burdet, «Experimental safety study on soft-tissue injury in robotics», IEEE Transactions on Robotics **27**, 918–930 (2011).