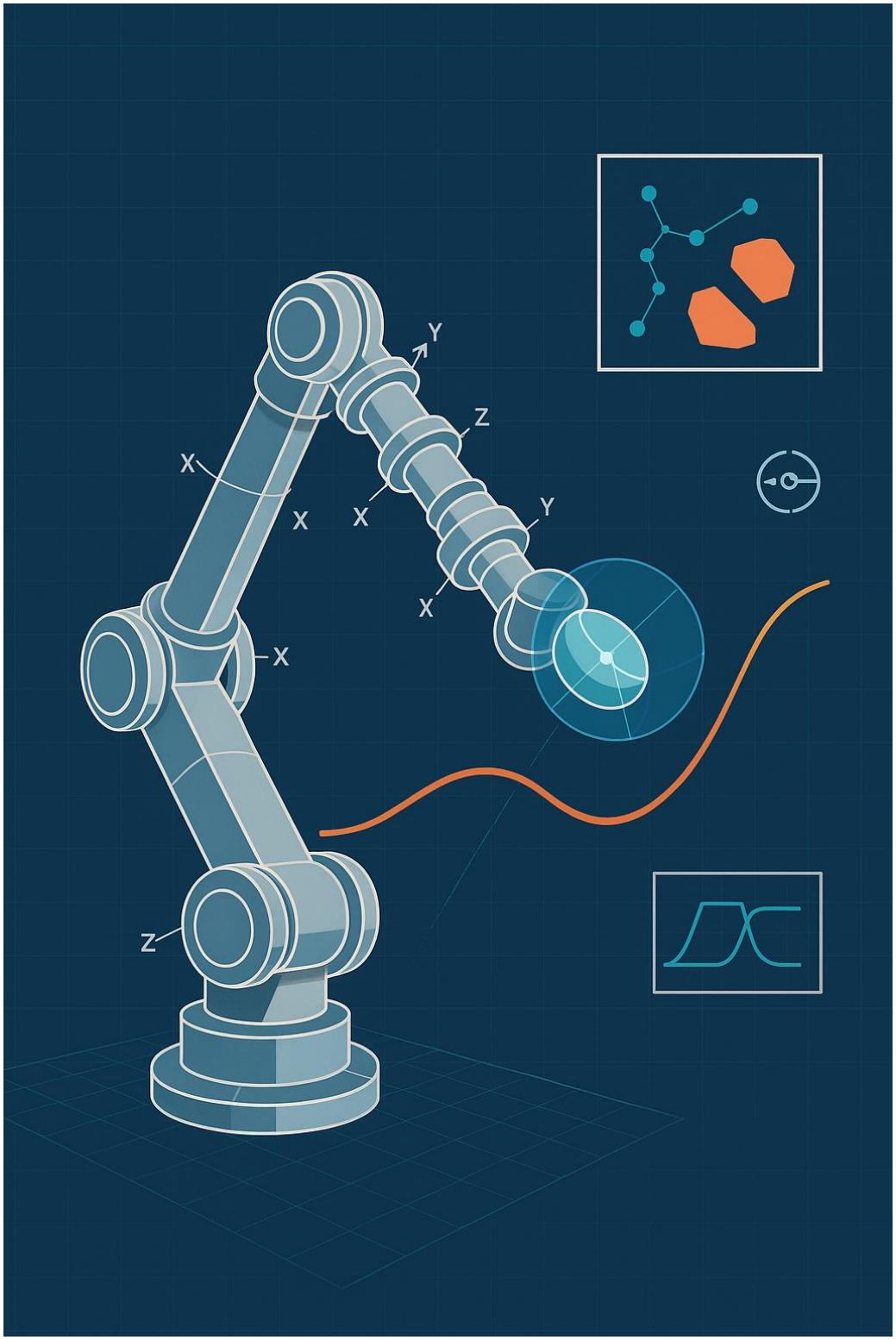
CSIC6843 – Lecture Unit 4

Kinematics & Motion Planning in Robotics



From Kinematics to Collision-Free Trajectories

Table of Contents

Introduction 1
4.1. Fundamental Concepts and Mathematical Foundations 2
Degrees of Freedom (DoF) and Redundancy 2
Forward Kinematics (FK) 3
Inverse Kinematics (IK) 3
Differential Kinematics and Jacobians 4
Singularity Analysis 5
4.2. Trajectory Generation and Path Planning 6
Path vs. Trajectory 6
Trajectory Parameterisation and Constraints 7
Trajectory Optimization 8
4.3. Configuration Space (C-Space) and Collision Detection 9
Configuration Space 9
Collision Detection 9
4.4. Motion Planning Algorithms 10
Graph- and Grid-Based Planners 10
Sampling-Based Planners 10
Optimization-Based Planners 10
Learning-Assisted Planning 10
4.5. Reactive and Hierarchical Planning 11
4.6. Multi-Robot Coordination and Human-Robot Interaction 12
Multi-Robot Coordination 12
Human-Robot Interaction (HRI) 12
4.7. Integration with Sensors and Control 13
Sensor Integration 13
Control Algorithms 13
4.8. Energy and Safety Considerations 14
4.9. Emerging Trends and Future Directions 15
4.10. Summary & Further Reading 16
Appendix: Glossary (Contextualised Terms) 17

Table of Figures

Figure 1 - "Illustration showing a 7-DoF robotic arm: the top highlights degrees of freedom through joint rotations, while the bottom shows redundancy—multiple possible arm paths to reach the same target point." 2
Figure 2 - A 2D arm diagram showing how joint positions determine the end-effector location... 3
Figure 3 - A manipulability ellipsoid, showing how easily the robot can move in different directions from a given pose—large axes indicate easy movement; small axes indicate harder movement. 4
Figure 4 - A diagram of the robot's workspace, with shaded "singularity zones" marked as problematic areas to avoid or navigate carefully. 5
Figure 5 - A 3D curve representing the path in space, with separate graphs showing velocity or acceleration versus time, illustrating how motion parameters change along the path..... 6
Figure 6 - Velocity profiles such as trapezoidal and S-curve, showing smooth acceleration and deceleration phases compared to abrupt motion changes..... 7
Figure 7 - A flowchart showing iterative refinement of a trajectory—starting from the initial plan, applying constraints, evaluating performance, and adjusting until objectives are met. 8
Figure 8 - Multiple robots navigating and coordinating in a shared space. 12

Unit 04: Kinematics & Motion Planning in Robotics

Introduction

Robotic motion is about turning a task like "pick up the part and place it here" into a sequence of joint movements that the hardware can execute safely and efficiently. This unit brings together the two pillars that make that possible—**kinematics** and **motion planning**. *Kinematics* models how joint motions produce end-effector positions and velocities, independent of forces. *Motion planning* chooses feasible, collision-free trajectories that respect the robot's physical limits and task constraints. Together, they connect high-level goals to low-level commands such as joint angles, velocities, and torques, drawing on rigid-body mechanics, control, optimisation, and increasingly, learning-based methods.

In this lecture unit, you will:

- Build core intuition for **degrees of freedom** and **redundancy**.
- Work through **forward** and **inverse kinematics**, then use the **Jacobian** to relate joint and end-effector velocities and to reason about **singularities**.
- Distinguish **paths** from **trajectories**, and apply time **parameterisation** under velocity, acceleration, and jerk limits; see how **trajectory optimisation** refines an initial guess.
- Move from workspace to **configuration space (C-space)** and understand **broad- vs narrow-phase** collision checking.
- Compare families of **motion planners**—graph/grid methods (e.g., A*), sampling-based methods (PRM, RRT, RRT*), optimisation-based planners, and learning-assisted variants.
- See how planners run in **hierarchies** (global/local/reflex) and how ideas extend to **multi-robot** settings and **human–robot interaction**.
- Connect planning to execution via **sensors and control** and factor in **energy** and **safety** considerations.
- Glimpse **emerging trends** such as language-guided planning and generative models for trajectory sampling.

4.1. Fundamental Concepts and Mathematical Foundations

Degrees of Freedom (DoF) and Redundancy

Degrees of Freedom (DoFs) describe each independent way a robot can move.

- A rotation counts as one DoF (like turning your wrist).
- A translation—moving in a straight line—also counts as one DoF (like sliding your hand forward).

Think of DoFs like control levers: the more you have, the more ways you can position or orient the robot.

For example, a robot arm that can rotate its shoulder, bend its elbow, and swivel its wrist might have **6 or 7 DoFs**, depending on how many separate motions each joint allows.

Redundancy
If a robot has more DoFs than the minimum needed to complete a task, we call it **redundant**.

This extra flexibility isn't wasted—it lets the robot:

- **Avoid obstacles**
- **Steer clear of joint limits** (the maximum bend or twist a joint can handle)
- **Optimise performance**, such as saving energy or avoiding awkward positions

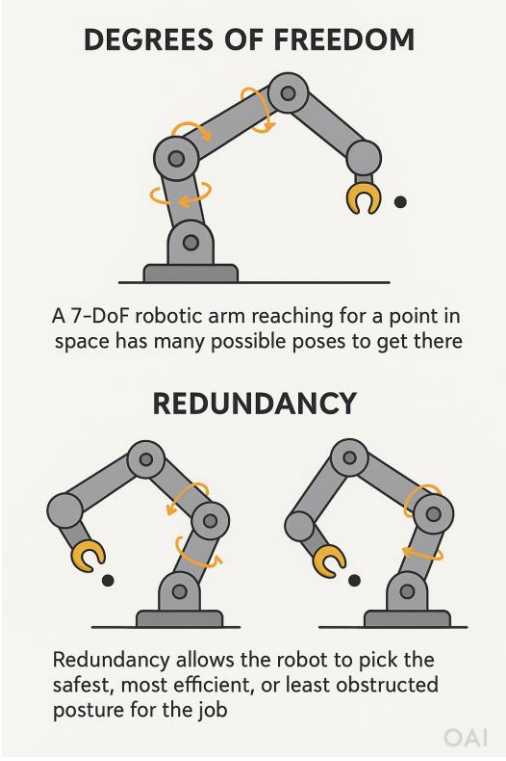


Figure 1 - "Illustration showing a 7-DoF robotic arm: the top highlights degrees of freedom through joint rotations, while the bottom shows redundancy—multiple possible arm paths to reach the same target point."

Example:
A 7-DoF robotic arm reaching for a point in space has many possible poses to get there. Just like you can touch your nose with your elbow tucked in or sticking out, the arm can twist its joints in different ways. Redundancy allows the robot to pick the **safest, most efficient, or least obstructed posture** for the job.

Forward Kinematics (FK)

Forward Kinematics (FK) answers the question:
"If I move each joint to these positions, where will my hand or tool end up?"

It calculates the **final position and orientation** of the robot's end-effector by combining the movement contributions of each joint in sequence. This process is essential for **robot control**, **path planning**, and **visualisation**.

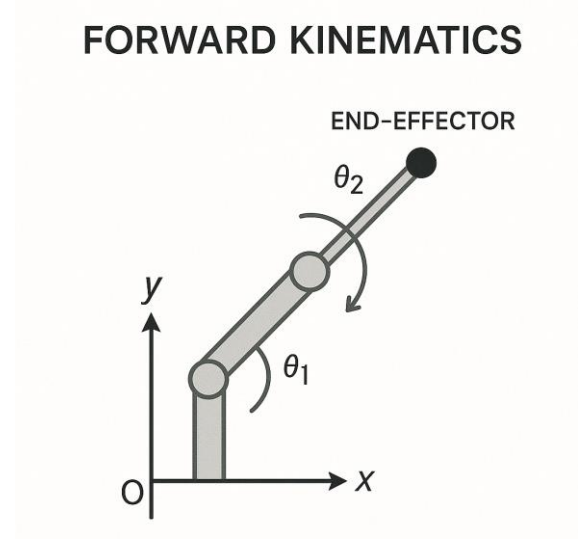


Figure 2 - A 2D arm diagram showing how joint positions determine the end-effector location.

Example:
Imagine a simple two-joint planar robot arm. By setting specific angles for the shoulder and elbow, FK tells you exactly where the hand will be in space—much like predicting where your fingertip will land if you bend your arm in a certain way.

Inverse Kinematics (IK)

Inverse Kinematics works backwards from the goal: instead of asking "*Where will the hand be if the joints move like this?*", it asks, "*How should each joint move so the hand ends up exactly here?*"

This problem is trickier than forward kinematics because:

- **Multiple solutions** can exist (different joint positions can put the hand in the same place).
- **No solution** might exist if the target is out of reach or blocked.

Robots typically solve IK using **iterative numerical methods**—step-by-step calculations that home in on an answer—balancing **accuracy** with **computational speed**.

When a robot has **redundancy** (more Degrees of Freedom than the task strictly needs), there can be many valid joint configurations. In those cases, the controller chooses the "best" one based on extra goals, such as:

- Avoiding collisions with the environment
- Steering clear of joint limits
- Minimising energy use or total movement

Example: A robotic arm reaching for the same point might adopt an "**elbow up**" or "**elbow down**" posture—both work, but one may be safer or more efficient.

Differential Kinematics and Jacobians

This area focuses on how small changes in **joint movements** translate into changes in the **velocity** and **direction** of the robot's end-effector (e.g., its hand or tool).

The **Jacobian** acts like a *mathematical translator*, converting **joint speeds** into **end-effector speeds**.

- If you know how fast each joint is moving, the Jacobian can predict the resulting speed and direction of the end-effector.
- Conversely, it can help determine how the joints should move to achieve a desired end-effector velocity.

The Jacobian is also useful for spotting **singularities**—robot poses where the math breaks down and precise control becomes impossible. At these points, the robot might lose the ability to move in certain directions or require infinite joint speed to achieve a small end-effector movement.

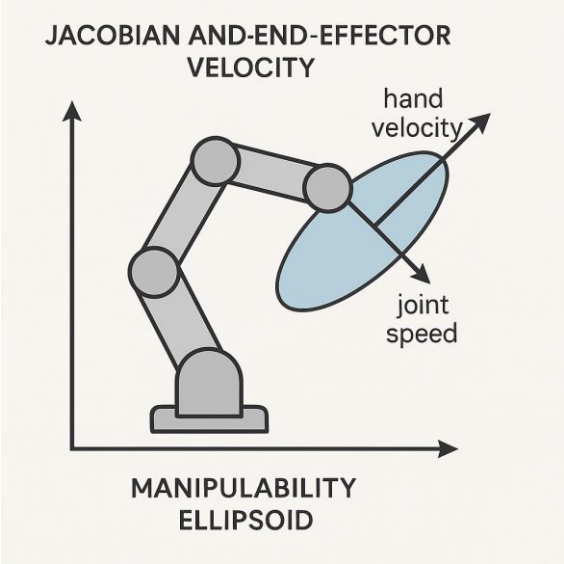


Figure 3 - A manipulability ellipsoid, showing how easily the robot can move in different directions from a given pose—large axes indicate easy movement; small axes indicate harder movement.

Example: Increasing the speed of just one joint and seeing how that change affects the overall velocity of the robot's hand.

Singularity Analysis

A **singularity** is a robot pose where movement in certain directions becomes impossible or would require *infinite* joint speed. You can think of these as the robot's **"blind spots"** in motion control—positions where the math behind joint-to-end-effector movement breaks down.

At a singularity:

- The robot may lose one or more controllable directions of motion.
- Even small end-effector movements could demand extreme or unsafe joint movements.

Robots must be able to **detect** these poses and either:

- Plan paths that **avoid** them entirely, or
- **Slow down** when moving near them to maintain stability and safety.

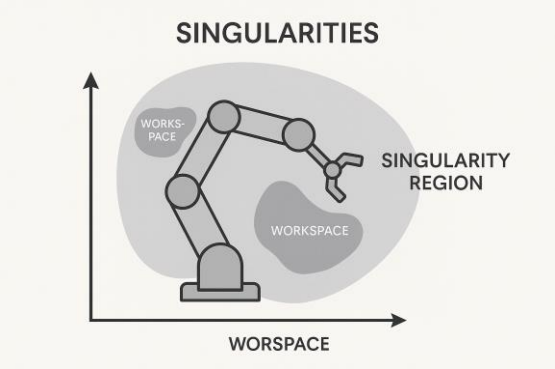


Figure 4 - A diagram of the robot's workspace, with shaded "singularity zones" marked as problematic areas to avoid or navigate carefully.

Example: A robot wrist aligned so that two rotational axes overlap. In this configuration, rotation around one axis becomes impossible without extreme joint movements.

4.2. Trajectory Generation and Path Planning

Path vs. Trajectory

A **path** describes *where* the robot moves—essentially the geometric route from a starting point to a goal. It only considers **position in space**, without worrying about timing or speed.

A **trajectory** builds on the path by adding the *when* and *how fast*. It specifies:

- **Velocity** at each point along the path
- **Acceleration** and **deceleration** phases
- **Smoothness** of motion to avoid jerks or instability

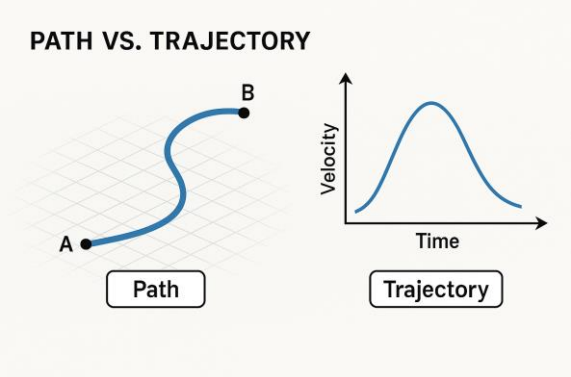


Figure 5 - A 3D curve representing the path in space, with separate graphs showing velocity or acceleration versus time, illustrating how motion parameters change along the path.

Example: A robotic arm moving from point A to point B along a smooth curve:

- The **path** is the curve itself.
- The **trajectory** includes timing profiles—how quickly the arm starts, speeds up, slows down, and stops.

Trajectory Parameterisation and Constraints

When planning a trajectory, the robot must operate within its **physical limits**—how quickly each joint can start, stop, or change speed without risking mechanical stress or instability.

Key considerations include:

- **Velocity limits** – maximum speed each joint can safely achieve
- **Acceleration limits** – how fast the joint can speed up or slow down
- **Jerk limits** – how quickly acceleration itself can change, to avoid sudden jolts

By limiting jerk, trajectories become smoother, reducing wear on components and improving control—especially important for tasks involving precision or delicate handling.

Orientation constraints may also apply, such as keeping a tool or sensor pointing in the same direction throughout motion.

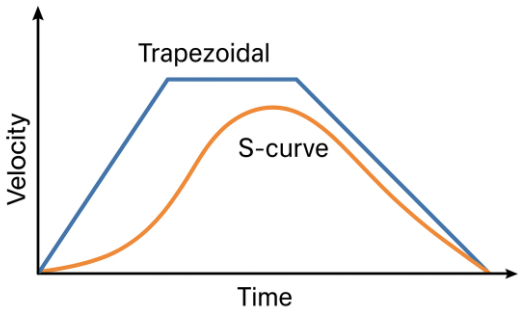


Figure 6 - Velocity profiles such as trapezoidal and S-curve, showing smooth acceleration and deceleration phases compared to abrupt motion changes.

Example: In a pick-and-place task with a fragile object, the robot uses a **smooth, jerk-limited trajectory** to move between points, avoiding sudden motions that could damage the object.

Trajectory Optimization

Once an initial trajectory is created, **optimisation** fine-tunes it to achieve the best possible performance while still meeting all **constraints** (like speed limits, joint ranges, or safety margins).

The process aims to **balance multiple objectives**, such as:

- **Minimising travel time**
- **Maximising energy efficiency**
- **Reducing wear** on mechanical components
- **Avoiding collisions** with the environment

Optimisation methods adjust the motion step by step—testing small changes, evaluating the results, and iterating—until they find a trajectory that is smoother, safer, and often faster.

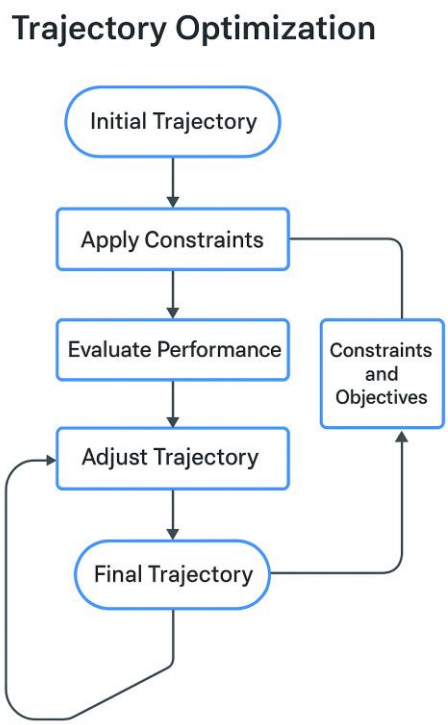


Figure 7 - A flowchart showing iterative refinement of a trajectory—starting from the initial plan, applying constraints, evaluating performance, and adjusting until objectives are met.

Example: A welding robot's arm path is optimised to complete welds in less time while maintaining precision and weld quality.

4.3. Configuration Space (C-Space) and Collision Detection

Configuration Space

Instead of describing the robot's movement in regular 3D space, **Configuration Space** (C-Space) represents every possible **pose** (position and orientation of all joints) as a single point in a higher-dimensional space.

In this abstract space:

- **Each axis** represents one joint's possible positions.
- **Obstacles** in the real world become **"forbidden zones"** in C-Space—regions where the robot would be in collision.

Motion planning in C-Space becomes a matter of finding a **continuous, collision-free path** from the start configuration to the goal configuration.

Example: For a robot arm with **two joints**, its C-Space can be shown as a 2D plot. Each point on the plot corresponds to a specific pair of joint angles. Areas marked as "blocked" represent joint combinations where the arm would collide with something in the workspace.

Collision Detection

To ensure safe operation, robots must check whether planned motions would cause **collisions** with obstacles or other parts of the robot itself. This process is usually performed in two main stages:

1. **Broad-phase detection** – A quick, coarse check that rules out obviously safe configurations.
    - Uses **simple bounding volumes** (like boxes, spheres, or capsules) around robot parts and obstacles.
    - Fast to compute, making it ideal for eliminating non-colliding cases early.
2. **Narrow-phase detection** – A precise, detailed check of the remaining cases.
    - Uses exact geometry (meshes or CAD models) to confirm whether contact actually occurs.
    - More computationally expensive, so it's only run on configurations that pass the broad-phase.

Emerging methods use **AI-based predictors** to speed up the process—learning from prior checks to quickly guess whether a configuration might cause a collision.

Example: First, bounding boxes detect that two robot links are far apart (safe). If they appear close, a detailed mesh-to-mesh intersection test confirms whether they truly collide.

4.4. Motion Planning Algorithms

Robotic motion planners differ in how they explore possible movements and find feasible, efficient paths. Four major categories are common:

Graph- and Grid-Based Planners

These planners **discretise** the robot's configuration space (C-space) or workspace into a **grid** or **graph** of possible positions.

- Use search algorithms like **A*** or **Dijkstra's** to find paths.
- **Strengths**: Precise and systematic.
- **Limitations**: Struggle with **high-dimensional spaces** due to exponential growth in possibilities.

Sampling-Based Planners

Rather than checking every possible position, these planners **randomly sample configurations** and connect them into a **roadmap** or **tree**.

- **Strengths:** Handle high Degrees of Freedom (DoF) efficiently.
- **Widely used today** for complex robots.
- **Examples:**
  - **PRM (Probabilistic Roadmap):** Builds a reusable roadmap for quick path queries.
  - **RRT (Rapidly-Exploring Random Tree):** Expands a tree rapidly toward the goal.
  - **RRT***: An improved version of RRT that refines the path over time for better quality.

Optimization-Based Planners

Start with an **initial path** and **iteratively refine it** to make it smoother, more efficient, and feasible.

- Directly handle **constraints** such as joint limits, velocity bounds, or obstacle clearance.
- Often combined with other planning methods for improved results.

Learning-Assisted Planning

Emerging AI-driven methods that **guide or speed up** planning.

- Learn from past experience to make planning **faster** and more **adaptive** to new environments.
- Can be combined with sampling or optimisation methods.

4.5. Reactive and Hierarchical Planning

Robots often face **dynamic environments** where unexpected obstacles or sudden task changes require on-the-fly adjustments. Planning systems are typically organised into **layers**, each operating at different **time scales**:

1. **Global Planner** –
    - Sets the **overall route or strategy** over longer time horizons.
    - Works with high-level goals, like mapping a delivery route across an entire building.
2. **Local Planner** –
    - Continuously **fine-tunes motion** in the short term.
    - Reacts to smaller, moving obstacles—like steering around a chair or a walking person.
3. **Reflex Layer** –
    - Provides **instant reactions** for safety.
    - Stops or alters motion immediately if a collision or hazard is imminent.

Example: A delivery robot uses its **global planner** to decide the best route to a destination, its **local planner** to navigate around a crowd, and its **reflex layer** to instantly stop if someone suddenly steps in front of it.

4.6. Multi-Robot Coordination and Human-Robot Interaction

Multi-Robot Coordination

When multiple robots work together, they must:

- **Avoid collisions** with each other
- **Coordinate and allocate tasks** efficiently
- Operate reliably even with **limited communication** or in the presence of **unexpected failures**

Common strategies include shared maps, priority-based movement rules, and distributed planning so robots can adapt if one fails or is delayed.

Human-Robot Interaction (HRI)

Robots working near people must prioritise **safety, predictability, and comfort**. This involves:

- Respecting **personal space** and **social norms**
- Moving in a way that humans can anticipate
- Using models that **predict human motion** to plan considerate and non-threatening trajectories

Example: A collaborative robot (**cobot**) slows its movement when a human approaches its workspace, ensuring safety and trust.

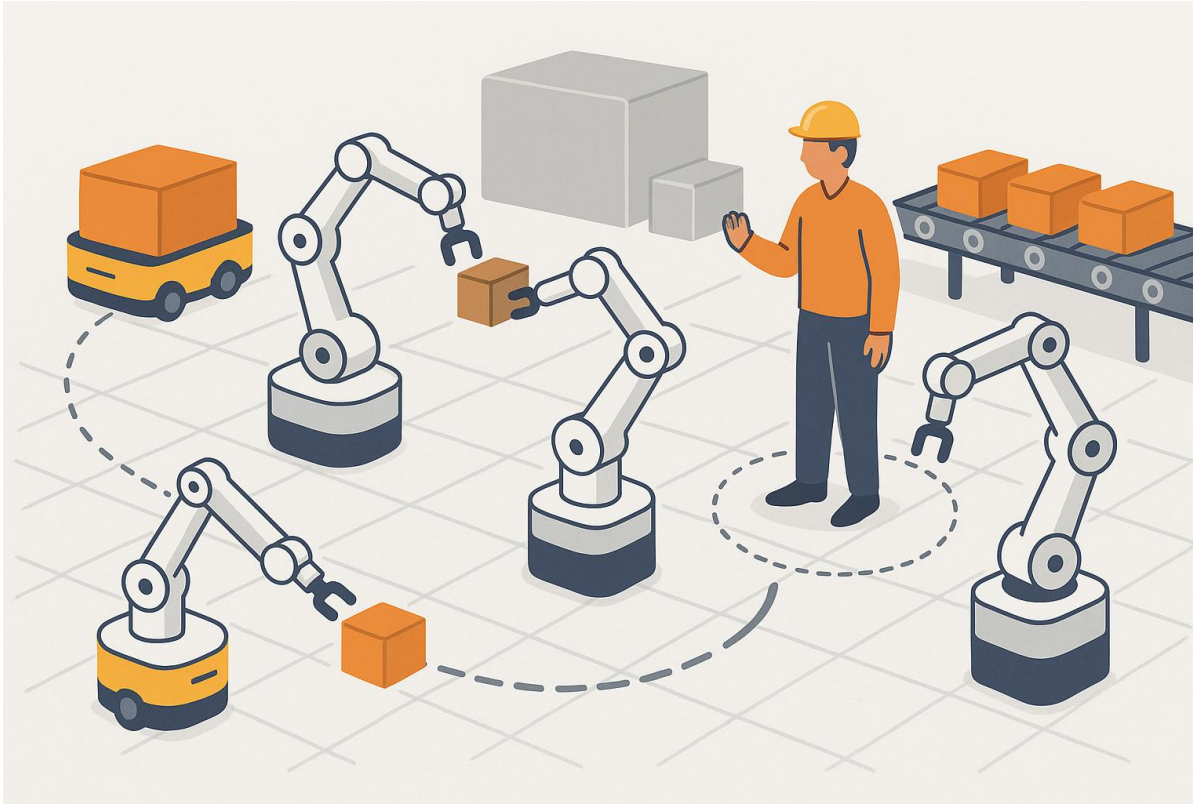


Figure 8 - Multiple robots navigating and coordinating in a shared space.

4.7. Integration with Sensors and Control

Sensor Integration

Robots rely on a variety of sensors—such as **cameras**, **LiDAR**, and **IMUs** (Inertial Measurement Units)—to perceive their surroundings and track their own movement.

- **Sensor fusion** combines data from multiple sources to improve accuracy and robustness.
- This real-time perception enables **safe navigation**, **precise manipulation**, and **quick responses** to environmental changes.

Control Algorithms

While **planning** decides *where* the robot should move, **controllers** ensure it *actually gets there*—smoothly, precisely, and safely.

- Controllers constantly compare the **planned motion** to the **actual motion**, making adjustments to correct errors.
- Advanced methods allow the robot to adapt to uncertainty or external forces.

Common methods include:

- **PID control** – A classic method that adjusts position, speed, and acceleration smoothly.
- **Impedance control** – Regulates force and motion together, enabling compliant, human-friendly interactions.

Example: A robot uses tactile sensors to detect vibrations from a fragile object it's holding. Its **impedance controller** adjusts grip pressure in real time, preventing damage while maintaining a secure hold.

4.8. Energy and Safety Considerations

Energy-Efficient and Safe Motion Planning

Energy-efficient motion planning aims to **reduce power consumption** and **extend operational life**, which is especially important for **mobile robots** and **drones** with limited battery capacity.

Planners consider **energy costs** alongside **safety constraints**, such as:

- **Speed limits** to maintain stability and reduce wear
- **Force limits** to ensure safe operation near humans
- **Smooth trajectories** to avoid sudden power spikes or mechanical stress

For robots working around people, these strategies help balance **efficiency**, **safety**, and **predictable behaviour**.

Before deployment, robots undergo **rigorous testing** to verify compliance with safety standards and ensure reliable, safe performance in real-world conditions.

Example: An autonomous warehouse robot selects a route that minimises battery usage while navigating around workers and avoiding congested areas.

4.9. Emerging Trends and Future Directions

Natural-Language Task and Motion Planning with LLMs

- Large Language Models (LLMs) are being integrated into planning pipelines to allow robots to understand and execute **natural-language commands**.
- This enables intuitive human–robot interaction, where a user can say *"Pick up the red box and place it on the top shelf"* and the system translates it into precise motion plans.

Diffusion and Generative Models for Trajectory Sampling

- Generative AI methods, including **diffusion models**, are being explored for **sampling complex, high-dimensional trajectories**.
- These models can produce diverse, high-quality motion samples, which are especially useful in cluttered or dynamic environments.

Cloud–Edge Hybrid Architectures

- Combining **edge computing** (on-robot) with **cloud resources** allows heavy computation—such as large-scale motion planning or AI inference—to be offloaded.
- Low-latency communication ensures the robot can still react quickly to real-world changes while benefiting from cloud-scale processing.

Data-Driven Learned C-Space Metrics

- Machine learning is being applied to learn **configuration space (C-space) distance metrics** from data.
- These learned metrics speed up **nearest-neighbour queries** in sampling-based planners, significantly accelerating pathfinding in complex spaces.

4.10. Summary & Further Reading

This lecture unit connected **kinematics** to **motion planning**, showing how we convert high-level goals into safe, feasible robot motions: **FK/IK**, **Jacobians**, and **singularity** reasoning; **path vs trajectory** with time **parameterisation** and constraints; **C-space** modelling with **collision checking**; families of **planners**; **reactive/hierarchical** execution; **multi-robot** and **HRI** considerations; **sensor–control** integration; plus **energy & safety** and current **trends**.

By now, you should be able to:

- **Model and analyse robot motion:** apply FK/IK, use the Jacobian to relate joint and task-space velocities, interpret manipulability, and recognise/mitigate singularities.
- **Plan time-feasible motion:** distinguish paths from trajectories; perform basic time parameterisation under velocity/acceleration/jerk and orientation constraints; understand when optimisation refines a seed trajectory.
- **Work in configuration space:** reason about obstacles as C-space regions; use broad- then narrow-phase collision checks (and self-collision) to validate motion.
- **Choose planning methods appropriately:** compare graph/grid (e.g., A*, Dijkstra), sampling (PRM/RRT/RRT*), optimisation-based, and learning-assisted planners, with their strengths/trade-offs.
- **Execute robustly:** understand global–local–reflex layers for reactivity; integrate perception via sensor fusion; use controllers (e.g., PID, impedance) to track plans safely and accurately.
- **Design with constraints:** plan for energy efficiency and safety during deployment, especially around people.
- **Track emerging directions:** language-guided task & motion planning, generative trajectory sampling, cloud–edge computation, and learned C-space metrics.

Further reading & practice

- **Lynch & Park — *Modern Robotics: Mechanics, Planning, and Control***: a balanced, rigorous treatment spanning kinematics through planning and control.
- **LaValle — *Planning Algorithms* (free online)**: canonical reference for graph- and sampling-based methods.
- **Journals:** *IEEE Robotics & Automation Letters* (RA-L) and *The International Journal of Robotics Research* (IJRR) for current research on motion planning.

Optional practitioner resources: OMPL and MoveIt tutorials for hands-on exposure to PRM/RRT pipelines and collision checking; try reproducing simple scenes from this unit and experimenting with velocity/jerk limits and obstacle layouts.

Appendix: Glossary (Contextualised Terms)

Key Term	**Description**
Acceleration Limits	The maximum rate at which a robot joint or end-effector can increase or decrease its speed.
A (A-star)*	A popular graph traversal and pathfinding algorithm used in grid-based motion planning.
Broad-phase Detection	The initial, quick stage of collision detection that uses simplified bounding volumes to identify potential collisions, ruling out non-colliding objects efficiently.
C-Space (Configuration Space)	An abstract, higher-dimensional space where every possible pose (position and orientation of all joints) of a robot is represented as a single point. Obstacles in the real world become "forbidden zones" in C-Space.
Collision Detection	The process of determining whether a robot or its planned motion would intersect with obstacles in the environment or with itself.
Control Algorithms	Mathematical procedures that compare a robot's planned motion to its actual motion and make adjustments to correct errors, ensuring smooth, precise, and safe execution.
Degrees of Freedom (DoF)	Each independent way a robot can move, such as a rotation around a joint or a translation along an axis.
Differential Kinematics	The study of how small changes in joint positions translate into changes in the end-effector's position and orientation, particularly concerning velocities.
Dijkstra's Algorithm	A graph search algorithm that finds the shortest paths between nodes in a graph, often used in grid-based motion planning.
End-effector	The part of the robot that interacts with the environment or performs the task (e.g., a gripper, tool, or hand).
Forward Kinematics (FK)	The calculation of the end-effector's position and orientation given the angles or positions of all the robot's joints.
Generative Models	AI models, such as diffusion models, that can generate new data instances, are being explored for creating diverse and high-quality trajectory samples.
Global Planner	The highest level of a hierarchical planning system, responsible for setting the overall route or strategy over longer time horizons.
Graph-Based Planners	Motion planning methods that discretise the robot's configuration space or workspace into a graph and use search algorithms to find paths.
Human-Robot Interaction (HRI)	The study and design of interactions between humans and robots, focusing on safety, predictability, and comfort.
IMUs (Inertial Measurement Units)	Electronic devices that measure a robot's orientation, velocity, and gravitational forces, often used for sensing motion.
Impedance Control	A control method that regulates the force and motion of a robot simultaneously, enabling compliant and human-friendly interactions.
Inverse Kinematics (IK)	The calculation of the required joint angles or positions for a robot to achieve a desired end-effector position and orientation.

Jacobian	A matrix used in differential kinematics that relates joint velocities to end-effector velocities, also crucial for singularity analysis.
Jerk Limits	Constraints on how quickly acceleration itself can change, used to ensure smoother trajectories, reduce wear, and improve control stability.
Kinematics	The study of motion without considering the forces that cause it, focusing on the geometry of movement.
Learning-Assisted Planning	Emerging AI-driven methods that use machine learning to guide, speed up, or adapt motion planning processes.
LiDAR	Light Detection and Ranging, a remote sensing method that uses pulsed laser light to measure distances and create detailed 3D maps of the environment.
Local Planner	A layer in hierarchical planning that continuously fine-tunes motion in the short term, reacting to smaller or moving obstacles.
LLMs (Large Language Models)	Advanced AI models capable of understanding and generating human-like text, being integrated into robotic systems for natural-language control.
Motion Planning	The process of determining a sequence of movements (a path or trajectory) for a robot to achieve a task safely and efficiently, often avoiding obstacles and respecting physical limits.
Narrow-phase Detection	The second, precise stage of collision detection that uses exact geometric models to confirm whether an actual collision occurs, typically run only after a broad-phase check.
Optimisation-Based Planners	Motion planning methods that start with an initial path and iteratively refine it to achieve desired objectives (e.g., smoothness, efficiency) while satisfying constraints.
Path	A purely geometric route or sequence of positions that a robot follows, without specifying timing or speed.
PID Control (Proportional-Integral-Derivative Control)	A widely used feedback control loop mechanism that adjusts a system's output based on the error between a desired setpoint and a measured process variable.
PRM (Probabilistic Roadmap)	A sampling-based motion planning algorithm that builds a reusable graph (roadmap) of collision-free configurations and their connections.
Redundancy	A characteristic of a robot having more degrees of freedom than the minimum necessary to complete a given task, providing extra flexibility.
Reflex Layer	The lowest and fastest layer in hierarchical planning, providing instant, automatic reactions for immediate safety or hazard avoidance.
Rigid-body Mechanics	The study of the motion of interconnected rigid bodies, foundational to understanding robot kinematics.
RRT (Rapidly-Exploring Random Tree)	A sampling-based motion planning algorithm that builds a tree of possible robot configurations by rapidly exploring the configuration space towards a goal.
*RRT**	An improved version of RRT that aims to find optimal or near-optimal paths by continuously rewiring the tree to reduce path costs.

Sampling-Based Planners	Motion planning methods that explore the configuration space by randomly sampling configurations and connecting them, efficient for high-dimensional robots.
Sensor Fusion	The process of combining data from multiple sensors to achieve a more accurate and robust understanding of the environment and the robot's state.
Singularity	A robot pose where the Jacobian loses rank, meaning the robot loses one or more controllable directions of motion or requires infinite joint speeds to achieve a desired end-effector movement.
Time Parameterisation	The process of adding timing information (velocities, accelerations) to a geometric path to create a trajectory.
Trajectory	A path with added timing information, specifying the robot's velocity, acceleration, and deceleration at each point, ensuring smooth and controlled motion.
Trajectory Optimisation	The process of refining an initial trajectory to achieve better performance (e.g., minimum time, energy efficiency) while still meeting all constraints.
Translation	A type of robot movement that involves moving in a straight line without rotation.
Velocity Limits	The maximum speed each robot joint or the end-effector can safely achieve.
Workspace	The physical space in which the robot operates and interacts with its environment.