

Robotics
Theory

Christian Rossi

Academic Year 2023-2024

Abstract

The course is composed by a set of lectures on autonomous robotics, ranging from the main architectural patterns in mobile robots and autonomous vehicles to the description of sensing and planning algorithms for autonomous navigation. The course outline is:

- Mobile robots' kinematics.
- Sensors and perception.
- Robot localization and map building.
- Simultaneous Localization and Mapping (SLAM).
- Path planning and collision avoidance.
- Robot development via ROS.

Contents

1	Introduction	1
1.1	History	1
1.2	ISO definitions	2
1.3	Robot architecture	3
2	Sensors and actuators	4
2.1	Sensors	4
2.1.1	Encoder	4
2.1.2	Time of flight telemeter	6
2.1.3	Reflective optosensors	7
2.1.4	Light detection and ranging (LIDAR)	8
2.1.5	Position sensor	8
2.2	Inertial sensor	9
2.2.1	Tactile sensor	9
2.2.2	Proximity sensor	9
2.3	Actuators	10
2.3.1	Direct current motor	10
2.3.2	Stepper motor	11
2.3.3	Servo motor	12
3	Robot Odometry	13
3.1	Introduction	13
3.2	Direct kinematic	14
3.3	Differential drive	16
3.3.1	Odometry	17
3.3.2	Sensors	19
3.4	Synchronous drive	19
3.4.1	Odometry	20
3.5	Omnidirectional drive	20
3.6	Tricycle drive	21
3.6.1	Odometry	22
3.7	Ackerman steering drive	22
3.8	Skid steering drive	23
3.9	Inverse kinematics	24
3.9.1	Differential drive	24
3.9.2	Synchro drive	24

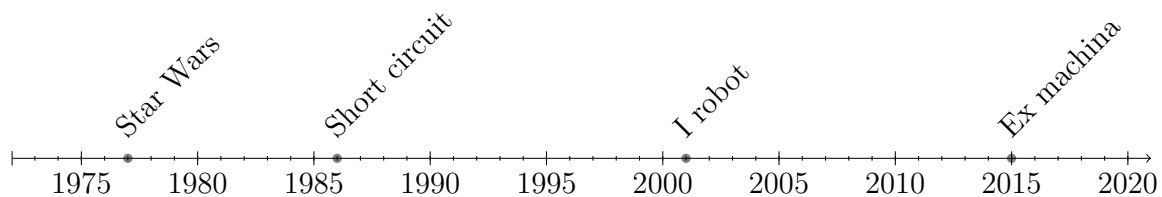
4	Robot localization	25
4.1	Introduction	25
4.2	Beam sensor model	25
4.2.1	Calibration	27
4.2.2	Model likelihood	28
4.3	Scan sensor model	28
4.4	Landmark model	29
4.4.1	Landmark detection	29
4.5	Bayesian filter	30
4.5.1	Bayes filter	31
4.5.2	Bayes filter algorithm	31
4.5.3	Discrete Bayes filter algorithm	32
4.6	Kalman filter	33
4.6.1	Discrete time Kalman filter	35
4.6.2	Extended Kalman filter	36
4.7	Particle filter	37

CHAPTER 1

Introduction

1.1 History

Filmography In the play "Rossum Universal Robots" from 1920, the term "robota" was introduced to refer to the first automatic robots. Several years later, Isaac Asimov penned the renowned science fiction series "I, Robot". Additionally, notable instances of robots in film include:



Robots evolution The mechanical era commenced in 1700 with the advent of the first automata, initially devised as specialized dolls for specific purposes. Transitioning from this era, the dawn of the 1920s saw a resurgence of interest in universal-purpose robots within the realm of fiction.

By 1940, the cybernetics era took root with the creation of the first turtles and telerobots. Grey Walters pioneered a significant development in this era by crafting a robotic tortoise that exhibited mechanical animal tropism (movement directed by stimuli).

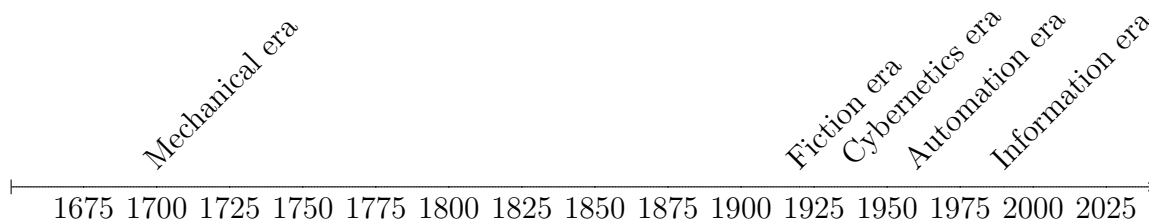
Two decades later, the automation era commenced with the inception of the first industrial robots, marking a shift towards mechanized processes. In 1961, UNIMATE, the inaugural industrial robot, initiated operations at General Motors, executing programmed tasks with precision and efficiency. In 1968, Marvin Minsky introduced the Tentacle Arm, a groundbreaking innovation resembling the movements of an octopus. This hydraulic-powered arm, controlled by a PDP-6 computer, featured twelve flexible joints facilitating maneuverability around obstacles.

In 1972, Shakey pioneered mobility in robotics with the creation of the Stanford cart, heralding advancements in mobile robotics.

The year 1980 witnessed the establishment of the first comprehensive definition of a robot as a reprogrammable, multifunctional manipulator designed for diverse tasks involving material, parts, tools, or specialized devices.

The onset of the information era in 1990 saw robots evolving to possess autonomy, cooperation, and intelligence, marking a significant leap in their capabilities.

Finally, in 2012, the International Organization for Standardization (ISO) established the standard definition for robots, consolidating their diverse functionalities and characteristics into a unified framework.



1.2 ISO definitions

Definition (*Robot*). A robot is an actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform intended tasks. Autonomy in this context means the ability to perform intended tasks based on current state and sensing, without human intervention.

Definition (*Service robot*). A service robot is a robot that performs useful tasks for humans or equipment excluding industrial automation application.

Definition (*Personal service robot*). A personal service robot or a service robot for personal use is a service robot used for a noncommercial task, usually by lay persons.

Examples of personal service robots include domestic servant robots, automated wheelchairs, personal mobility assist robots, and pet exercising robots.

Definition (*Professional service robot*). A professional service robot or a service robot for professional use is a service robot used for a commercial task, usually operated by a properly trained operator.

Examples of professional service robots encompass cleaning robots for public spaces, delivery robots in offices or hospitals, fire-fighting robots, rehabilitation robots, and surgical robots in hospitals. In this context, an operator is an individual designated to initiate, oversee, and terminate the intended operation of a robot or robot system.

Notes A robot system is defined as a system comprising robots, end-effectors, and any machinery, equipment, or sensors that support the robot in performing its tasks.

According to this definition, service robots require a degree of autonomy, which can range from partial autonomy involving human-robot interaction to full autonomy without active human intervention. Human-robot interaction involves information and action exchanges between humans and robots via a user interface to accomplish tasks.

Industrial robots, whether fixed or mobile, can also be considered service robots if they are utilized in non-manufacturing operations. Service robots may or may not feature an arm structure, which is common in industrial robots. Additionally, service robots are often mobile, but this is not always the case.

Some service robots consist of a mobile platform with one or several arms attached, controlled similarly to industrial robot arms. Unlike their industrial counterparts, service robots do not necessarily need to be fully automatic or autonomous. Many of these machines may assist a human user or operate via teleoperation.

1.3 Robot architecture

A machine gathers information from a set of sensors and utilizes this data to autonomously execute tasks by controlling its body parts.

One commonly employed model in robotics is the sense plan act paradigm, which forms the foundation of cognitive robotics. In this model, the sensing phase involves collecting data from sensors, the planning phase utilizes algorithms to process this data, and the action phase involves executing commands through actuators. This architecture is illustrated in the following diagram.

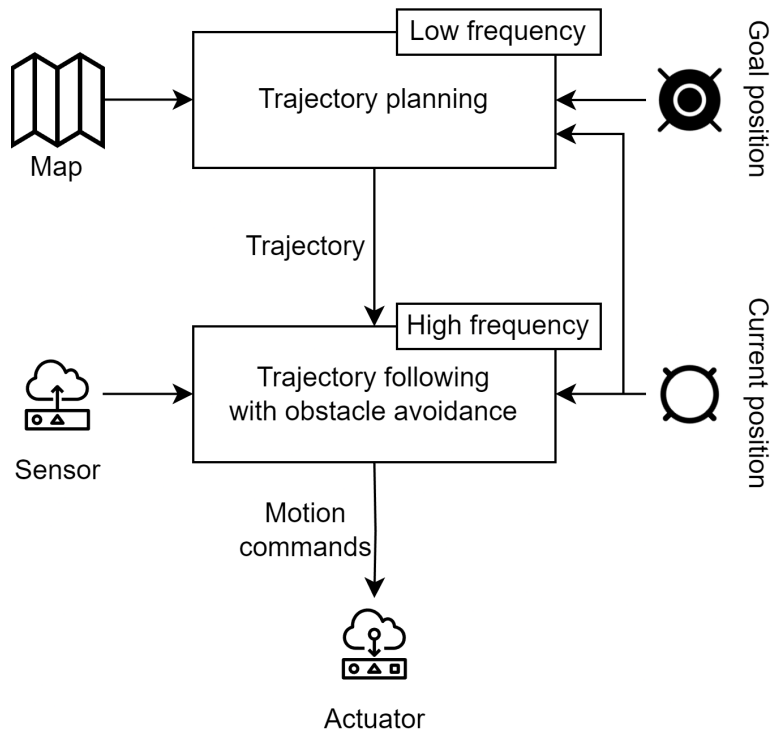


Figure 1.1: Sense plan act architecture

CHAPTER 2

Sensors and actuators

2.1 Sensors

Sensors serve to detect both the internal condition of the robot (proprioceptive sensors) and the external state of the environment (exteroceptive sensors). Another classification for sensors can be based on whether they are passive, which measure physical properties, or active, which involve an emitter and a detector.

2.1.1 Encoder

An encoder translates rotary motion or position of a motor/joint into electronic pulses. Encoders come in two primary types:

- *Linear encoder*: comprising a lengthy linear read track and a compact read head, linear encoders are designed for linear motion measurement.
- *Rotary encoder*: suitable for both rotary and linear motion, rotary encoders convert rotary motion into electrical signals. They are further categorized as incremental or absolute.

Their operation proceeds as follows:

1. A light-emitting diode (LED) projects a light beam onto a tape striped with red and black segments.
2. The reflected light is captured by a photodetector.
3. The photodetector generates a periodic wave whose frequency varies based on the speed of the tape.

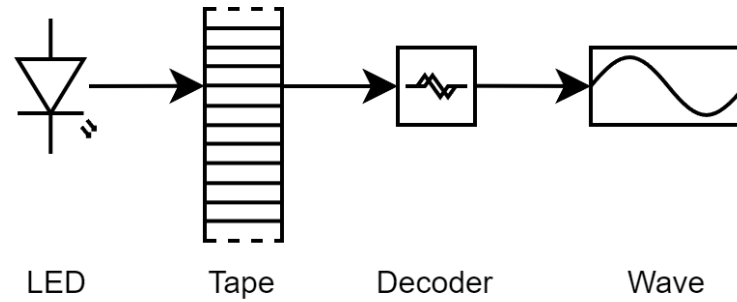


Figure 2.1: Linear encoder structure

Incremental rotary encoders Incremental rotary encoders operate based on the photoelectric principle, employing a disk with alternating transparent and opaque zones containing two traces or sensors. These traces facilitate the identification of rotation direction and enhance resolution through quadrature.

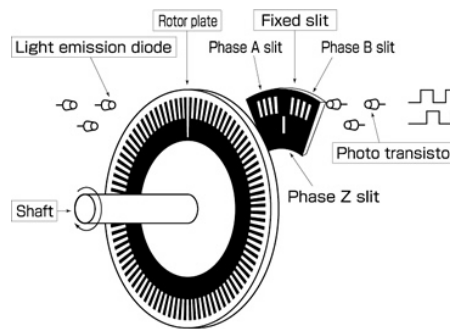


Figure 2.2: Incremental rotary encoder

To determine speed and direction, the quadrature technique is employed, where the two signals are shifted by $\frac{1}{4}$ step. Denoting N as the number of steps of light/dark zones per turn, the resolution is given by:

$$\text{resolution} = \frac{360^\circ}{4N}$$

Using this technique:

- If a transition from 11 to 10 occurs, it indicates a counterclockwise rotation.
- Conversely, if a transition from 11 to 01 occurs, it indicates a clockwise rotation.

The encoders' limitation lies in their inability to determine the actual position relative to the starting point. The only feasible solution involves resetting to the starting position and then incrementally counting until reaching the desired position.

Absolute rotary encoder The absolute rotary encoder addresses the limitation of determining absolute position by encoding it directly on the disk.

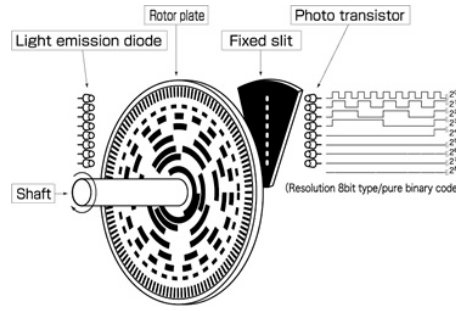


Figure 2.3: Incremental rotary encoder

The disk features transparent and opaque areas arranged in concentric rings. Each bit of position data is represented by a corresponding ring, offering an absolute resolution of:

$$\text{resolution} = \frac{360^\circ}{2^N}$$

In robotic applications, a minimum of 12 rings are typically employed. To prevent ambiguities, binary codes with single variations, such as Gray code, are utilized.

2.1.2 Time of flight telemeter

The time-of-flight telemeter records the duration between when the emitter generates a signal and when the detector detects its reflection. The signal travels a distance of $2d$, and the time of flight is given by:

$$\Delta t = \frac{2d}{c}$$

The initial type of sensor utilizing this principle in robotics was the sonar, which relies on sound waves. Sound waves, with their slower speed of approximately 340 m/s , and their relatively broad directionality ranging from 20° to 40° , offer an advantage in measuring shorter distances.

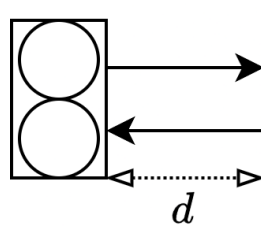


Figure 2.4: Sonar sensor

Sonar	
Range (m)	0.3 up to 10
Accuracy (m)	0.025
Cone opening ($^\circ$)	30
Frequency (Hz)	50000

The primary limitation is the susceptibility of the signal-to-noise, particularly from significant reflections. Selecting an appropriate range is crucial depending on the specific application. However, these sensors may not function optimally in all scenarios, due to factors such as:

- Balancing sampling frequency.
- Dealing with reflections off walls.
- Detecting small or soft objects.

Additionally, it's worth noting that room dimensions may appear distorted, especially around corners.

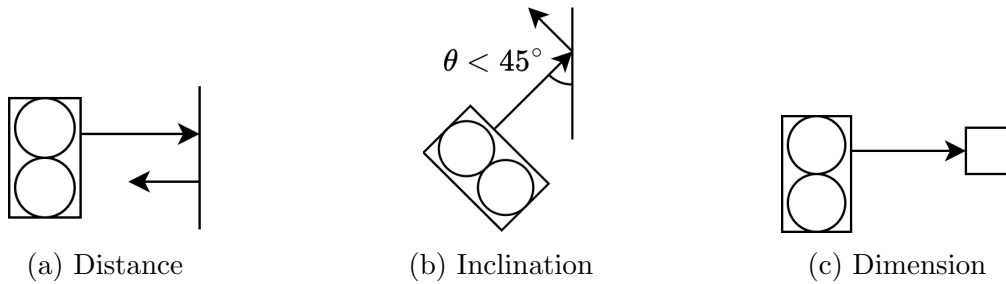


Figure 2.5: Possible problems for sonar sensors

2.1.3 Reflective optosensors

Reflective optosensors are active sensors where the emitter is a source of light and the detector is a light detector. This type of sensors uses triangulation to compute distance:

1. Emitter casts a beam of light on the surface.
2. The detector measures the angle corresponding to the maximum intensity of returned light.
3. Being s the distance between the emitter and the detector we have:

$$d = s \cdot \tan \alpha$$

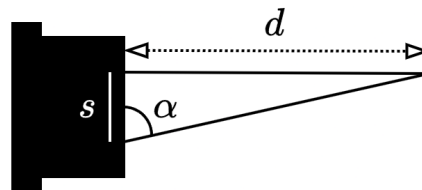


Figure 2.6: Reflective optosensor

Infrared sensors are relatively inexpensive and sturdy, but they have their drawbacks. They exhibit nonlinear characteristics that require calibration. Additionally, there can be ambiguity when used at short ranges, necessitating precise placement within the robot. Their fixed ranges and opening angles mean that careful selection is needed for optimal performance in various applications. Moreover, they may encounter issues with reflections under certain conditions.

An instance of such technology is the Kinect, an input device designed by Microsoft (originally by Primesense) for Xbox 360. This device functions as a three-dimensional scanner and is equipped with an infrared projector, an infrared camera, and an RGB camera.

Kinect	
<i>Range (m)</i>	0.7 up to 6
<i>Horizontal cone opening (°)</i>	57
<i>Vertical cone opening (°)</i>	43
<i>Infrared camera</i>	11-bit 640×480
<i>RGB camera</i>	30 Hz 8-bit 640×480

In this device, the distance from the camera Z_k is calculated as:

$$Z_k = \frac{Z_0}{1 + \frac{d}{fb} Z_0}$$

Time of flight camera Three-dimensional time-of-flight (TOF) cameras illuminate the scene using a modulated light source and capture the reflected light. The phase shift between illumination and reflection is then translated into distance information.

These sensors encounter challenges such as utilizing illumination from a solid-state laser or a near-infrared ($\sim 850\text{ nm}$) LED, where an imaging sensor converts captured light into electrical current. Additionally, distance information is encoded within the reflected component. Consequently, a high ambient component diminishes the signal-to-noise ratio (SNR).

2.1.4 Light detection and ranging (LIDAR)

Laser sensors offer superior accuracy with the following capabilities:

- Providing 180 ranges across a 180° field of view (expandable to 360°).
- Scanning 1 to 64 planes.
- Delivering scan rates of 10-75 scans per second.
- Achieving range resolutions of less than 1 cm .
- Offering a maximum range of up to 50-80 meters.
- Facing challenges only with mirrors, glass, and matte black surfaces.

2.1.5 Position sensor

Positioning outdoors can be determined using a Global Navigation Satellite System (GNSS), with multiple constellations available including GPS, GLONASS, Beidou, Galileo, and more.

The Global Positioning System (GPS) comprises 24 satellites circling the Earth twice daily. These satellites emit synchronized signals containing location and time data. Receivers compare the transmitted and received signal times to determine position. At least four satellite signals are needed for accurate positioning. The typical accuracy of GPS is approximately 2.5 m at a 2 Hz refresh rate, with the potential for even greater precision of around 20 cm with Differential GPS (DGPS).

There is also the RTKGPS that improves the time resolution with respect to the DGPS.

There are several limitations associated with GPS:

- It does not function indoors, underwater, or in urban canyons.

- Line of sight reception is required for optimal performance.
- GPS signals are susceptible to multiple paths and reflections, which can affect accuracy.

2.2 Inertial sensor

The inertial sensor can be divided into the following categories:

- *Gyroscopes*: measure angular velocities.
- *Accelerometers*: gauge linear accelerations with reference to the gravitational vector.
- *Magnetometers/compasses*: determine orientation based on the earth's magnetic field vector.

An Inertial Measurement Unit (IMU) integrates gyroscopes, accelerometers, and magnetometers to offer a complete six degrees of freedom pose estimate. However, integrating inertial measurements, such as for position computation, accumulates errors and drifts notably over time, particularly when using inexpensive MEMS (Micro-Electro-Mechanical Systems) technology.

2.2.1 Tactile sensor

Tactile sensors serve manipulation purposes and fall into two main categories:

- *Binary*: utilize switches placed on the fingers of a manipulator. Can be arranged in arrays (bumpers) on the external side to detect and avoid obstacles.
- *Analogical* (real valued): consist of soft devices producing a signal proportional to the local force. Utilize mechanisms like a spring coupled with a shaft or soft conductive material that changes resistance with compression. Capable of measuring movements tangential to the sensor surface.

2.2.2 Proximity sensor

Proximity sensors detect the presence of objects within a defined distance range, employed for grasping items and navigating around obstacles. Various technologies are utilized for this purpose:

- *Ultrasonic*: low cost.
- *Inductive*: detects ferromagnetic materials within a millimeters distance.
- *Hall effect*: detects ferromagnetic materials, small, robust, and inexpensive.
- *Capacitive*: detects any object, binary output, high accuracy when calibrated for a specific object.
- *Optical*: utilizes infrared light, offering binary or real-valued output.

2.3 Actuators

Effectors are responsible for altering the state of the environment, with actuators facilitating the actions of effectors. In robotics, we employ various types of actuators:

- *Electric motors*: these devices convert electrical energy into mechanical energy by leveraging the principles of electromagnetism. They produce rotational motion through the interaction between magnetic fields and electric currents.
- *Hydraulics*: this technology utilizes fluids to transmit force, employing the principles of fluid mechanics to generate, control, and transfer power via pressurized liquids.
- *Pneumatics*: a branch of engineering that employs compressed air or gas to transmit and regulate power, akin to hydraulics but using air or gas instead of liquids.
- *Photo-reactive materials*: these substances undergo a chemical change upon exposure to light.
- *Chemically reactive materials*: substances in this category undergo chemical reactions with other materials or their surroundings.
- *Thermally reactive materials*: these substances undergo changes in properties or behavior when subjected to variations in temperature.
- *Piezoelectric materials*: materials that generate electric charges in response to mechanical stress or pressure, while also displaying mechanical deformation under an electric field.

Originally, early robots were equipped with hydraulic and pneumatic actuators. Hydraulic actuators were costly, heavy, and required significant maintenance, making them suitable mainly for larger robots. Pneumatic actuators found use in stop-to-stop applications like pick-and-place tasks due to their swift actuation.

In modern times, electrical motors have become the prevalent choice for actuators. Typically, each joint incorporates its dedicated motor along with a controller. High-speed motors are often paired with elastic gearing to moderate their speed. These motors necessitate internal sensors for precise control. Stepper motors, on the other hand, don't require internal sensors; however, in case of an error, their exact position becomes unknown.

2.3.1 Direct current motor

Direct Current (DC) motors transform electrical energy into mechanical energy. They are compact, cost-effective, reasonably efficient, and straightforward to operate.

Electric current flows through coils of wire arranged on a rotating shaft. These wire loops create a magnetic field that interacts with the magnetic fields of permanent magnets positioned nearby. The resulting interaction between these magnetic fields causes them to repel each other, resulting in the rotation of the armature.

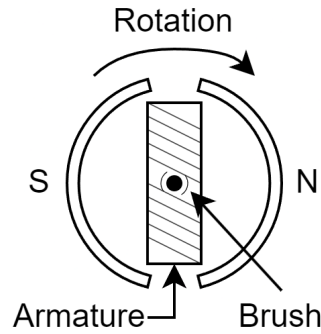


Figure 2.7: Brushed motor structure

Continuously adjusting the current causes the armature to keep rotating and generating motion. This current modification is facilitated by two connectors positioned at the center of the armature, known as brushes. It's worth noting that in lower-cost electrical motors, the external magnets remain stationary. However, these budget-friendly versions encounter several issues related to their brushes:

- Brushes gradually wear out over time.
- Brushes generate noise during operation.
- They impose a maximum speed limit.
- Cooling them proves to be challenging.
- They restrict the number of poles that can be utilized.

To circumvent this issue, one can opt for brushless motors, where external magnets are substituted with copper coils and a magnet is positioned at the center. This configuration yields a motor wherein brushes are replaced by electronics, permanent magnets reside on the rotor, and electromagnets are situated on the stator. While these motors offer superior performance, they also come at a higher cost compared to their brushed counterparts.

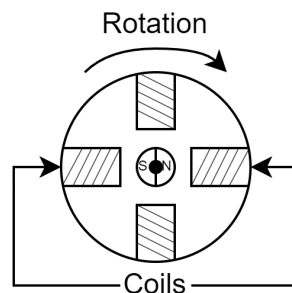


Figure 2.8: Brushless motor structure

2.3.2 Stepper motor

The stepper motor, a type of synchronous electric motor lacking brushes, transforms digital pulses into mechanical shaft rotations.

A stepper motor offers several advantages: it provides a direct correlation between input pulse and rotation angle, maintains full torque even at standstill when windings are energized,

enables precise positioning and repeatability, responds promptly to starting, stopping, and reversing commands, boasts high reliability due to the absence of contact brushes, facilitates open-loop control which simplifies and reduces costs, supports very low-speed synchronous rotation with directly coupled loads, and offers a wide range of rotational speeds. However, there are also disadvantages: it necessitates a specialized control circuit, consumes more current compared to DC motors, experiences a reduction in torque at higher speeds, risks resonances if not adequately managed, and finds it challenging to operate at extremely high speeds.

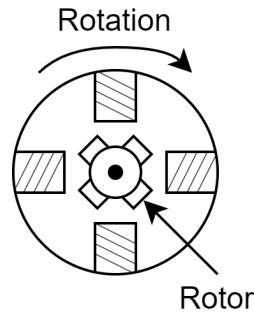


Figure 2.9: Stepper motor structure

The step angle, denoted by φ , can be determined using the following formula:

$$\varphi = \left(\frac{N_s - N_r}{N_s \cdot N_r} \right) \times 360^\circ$$

In this equation, N_s represents the number of teeth on the stator, and N_r represents the number of teeth on the rotor.

2.3.3 Servo motor

A servo is a type of specialized motor designed to precisely move its shaft to a specific position. These motors find common use in hobby radio control applications. They possess the capability to measure their own position and adjust for external loads in accordance with a control signal.

Servo motors are typically constructed from direct current motors with additional components including gear reduction, a position sensor, and control electronics. The travel range of the shaft is usually limited to 180 degrees, which is adequate for the majority of applications.

Robot Odometry

3.1 Introduction

For autonomous robots and unmanned vehicles to execute their tasks effectively, they require: accurate self-location information, and detailed maps of the environment. However, these requirements aren't always feasible or dependable due to the following reasons:

- Global Navigation Satellite Systems (GNSS) may not always be reliable or available.
- Not all areas have been accurately mapped.
- Environmental conditions can change dynamically.
- Maps need regular updates to remain current and reliable.

The robot's position can be regarded as a random variable due to the uncertainty inherent in our estimation of its true position. The full SLAM (Simultaneous Localization and Mapping) problem involves determining the distribution of both the robot's poses and the positions of landmarks, considering the robot's actions and sensor measurements:

$$\Pr(\Gamma_{1:t}, l_1, \dots, l_N | Z_{1:t}, U_{1:t})$$

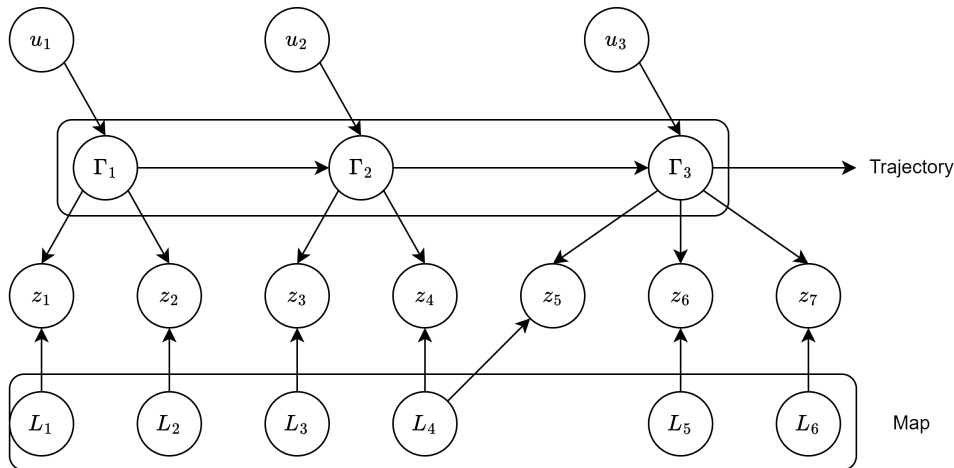


Figure 3.1: Simultaneous localization And Mapping

If a complete trajectory isn't necessary, a simplified version known as online SLAM can be used. This method provides the entire map and calculates the probability of only the most recent pose based on all measurements and actions:

$$\Pr(\Gamma_t, l_1, \dots, l_N | Z_{1:t}, U_{1:t}) = \int \int \int_1^{t-1} \Pr(\Gamma_{1:t}, l_1, \dots, l_N | Z_{1:t}, U_{1:t})$$

It's important to note that the term pose encompasses not only the position but also the orientation of the robot relative to the environment.

The motion model incorporates all actions u_1, \dots, u_N and their resulting poses $\Gamma_1, \dots, \Gamma_N$ describing how the robot's pose changes through the actuators.

On the other hand, the sensor model involves all poses $\Gamma_1, \dots, \Gamma_N$, all position probabilities z_1, \dots, z_N , and the map with landmarks L_1, \dots, L_N . It defines the probability distribution of a specific measurement given the robot's pose and the positions of the landmarks.

3.2 Direct kinematic

The robot kinematic is based on the motion model.

Definition (*Wheeled mobile robots*). A robot capable of locomotion on a surface solely through the actuation of wheel assemblies mounted on the robot and in contact with the surface. A wheel assembly is a device which provides or allows motion between its mount and surface on which it is intended to have a single point of rolling contact.

Various kinematic configurations are feasible:

- *Differential drive* (two wheels): basic design, prone to disturbances from uneven terrain, and lacks lateral translation capability.
- *Tracks*: ideal for outdoor surfaces, movement precision compromised, especially during rotations, intricate structure and behavior, and lateral translation not achievable.
- *Omnidirectional* (synchro drive): utilizes all three degrees of freedom, sophisticated design and functionality, and intricate structural composition.

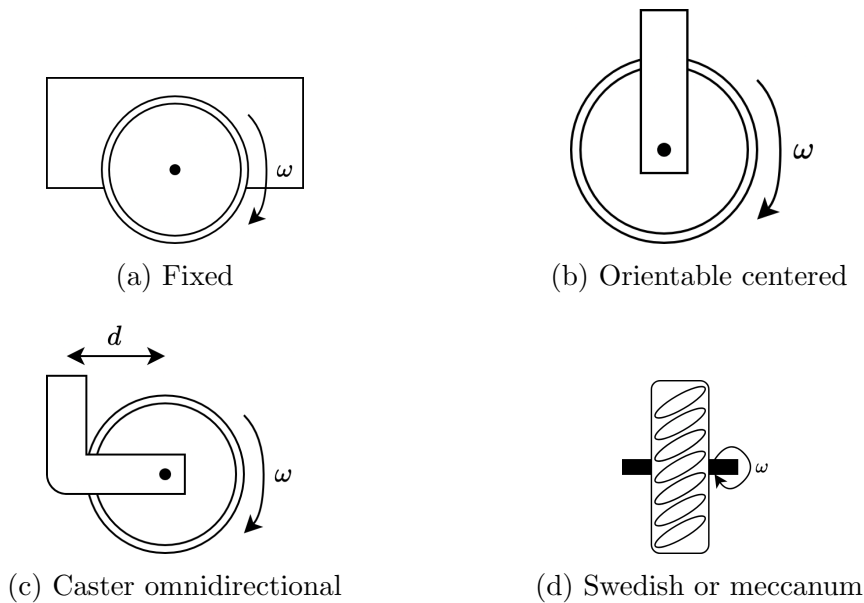


Figure 3.2: Wheel classification

Definition (*Locomotion*). Locomotion involves initiating movement in an autonomous robot:

Motion is achieved by applying forces to the vehicle.

Definition (*Dynamics*). Dynamics encompasses the analysis of motion through the modeling of forces, as well as the associated energies and velocities involved in these movements.

Definition (*Kinematics*). Kinematics is the examination of motion devoid of considerations regarding influencing forces.

It focuses on the geometric relationships dictating the system's behavior and the correlation between control parameters and the system's behavior in state space.

Definition (*Direct kinematics*). Direct kinematics involves determining the pose (x, y, θ) that a robot achieves given specific control parameters and a time of movement t .

Definition (*Inverse kinematics*). Inverse kinematics pertains to finding the control parameters necessary to reach a specified final pose (x, y, θ) within a given time t .

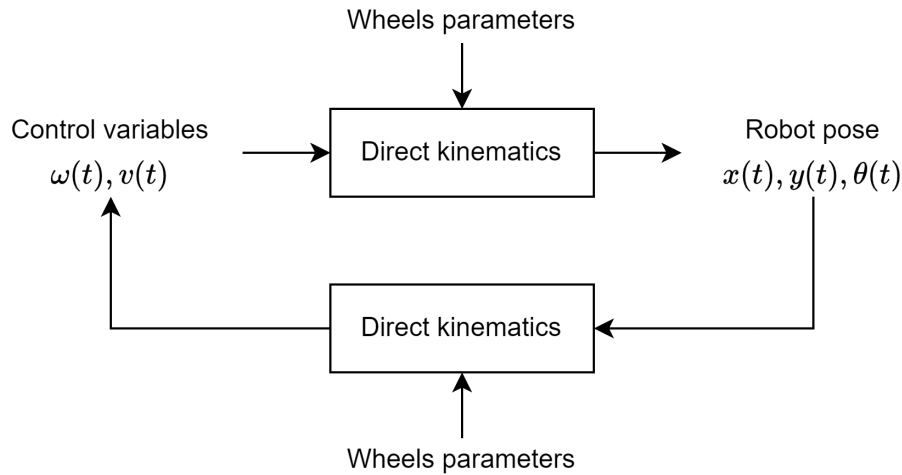


Figure 3.3: Direct and inverse kinematics

Wheels assumptions Several assumptions must be established concerning the wheels:

1. The robot consists solely of rigid components.
2. Each wheel possesses a single steering link.
3. Steering axes are perpendicular to the ground.
4. The wheel undergoes pure rolling about its axis (x -axis).
5. There is no translational movement of the wheel.

The critical parameters defining the wheels include their radius r , linear velocity v , and angular velocity ω .

For a robot to maneuver in the plane without slipping, the axes of the wheels must intersect at a specific point known as the Instantaneous Center of Curvature (ICC) or Instantaneous Center of Rotation (ICR). Failure of the wheel axes to intersect at a single point renders the robot immobile.

Cartesian representation The global reference system is external to the robot and serves as the frame of reference. The robot is characterized by its coordinates (x_b, y_b) relative to this reference frame. The angle θ represents the orientation of the robot's frontal face with respect to the x -axis.

Furthermore, a robot-centric reference frame can be established, centered within the robot itself.

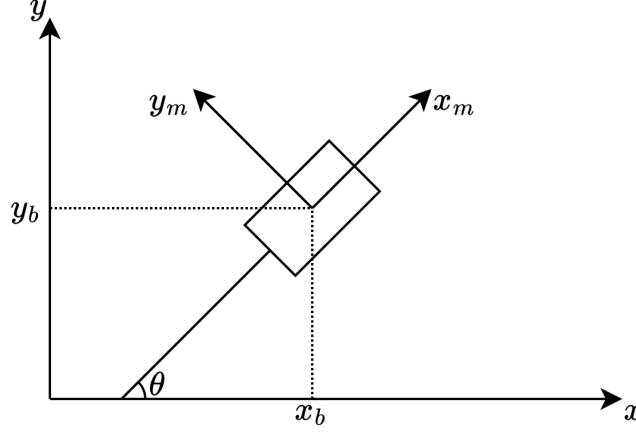


Figure 3.4: Robot reference Cartesian planes

The pose of the robot is determined by its position and orientation relative to the global reference system:

$$P(x_b, y_b, \theta) = (x, y, \theta)$$

3.3 Differential drive

The differential drive robot consists of two wheels positioned on the same axis, each with its own independent motor, along with a third passive supporting wheel.

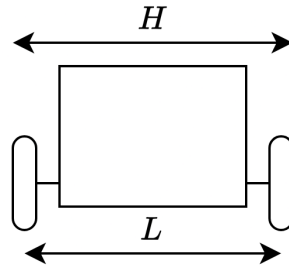


Figure 3.5: Differential drive robot

The baseline L represents the distance between the contact points of the wheels.

The variables under independent control are the velocities of the right wheel, v_R , and the left wheel, v_L . The robot's pose is represented in the base reference frame as $P(x, y, \theta)$.

Control inputs are the linear velocity of the robot, v , and its angular velocity, ω , which are linearly related to v_R and v_L . Both right and left wheels follow circular paths with an angular

velocity of ω and different curvature radii R :

$$\begin{cases} \omega \left(R + \frac{L}{2} \right) = v_R \\ \omega \left(R + \frac{L}{2} \right) = v_L \end{cases}$$

Given v_R and v_L , ω can be found by solving for R and equating:

$$\omega = \frac{v_R - v_L}{L}$$

Similarly, R can be found by solving for ω and equating:

$$R = \frac{L}{2} \frac{v_R + v_L}{v_R - v_L}$$

Note that rotation in place is achieved by setting $R = 0$ and $v_R = -v_L$, while linear movement is accomplished by setting $T = \infty$ and $v_R = v_L$.

The wheels move around an Instantaneous Center of Curvature (ICC) on a circumference with an instantaneous radius R and angular velocity ω :

$$\text{ICC} = \left\{ x + R \cos \left(\theta + \frac{\pi}{2} \right), y + R \sin \left(\theta + \frac{\pi}{2} \right) \right\} = \{ x - R \sin(\theta), y + R \cos(\theta) \}$$

The overall change in position over time is given by the equation:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega \cdot \delta t) & -\sin(\omega \cdot \delta t) & 0 \\ \sin(\omega \cdot \delta t) & \cos(\omega \cdot \delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - \text{ICC}_x \\ y - \text{ICC}_y \\ \theta \end{bmatrix} + \begin{bmatrix} \text{ICC}_x \\ \text{ICC}_y \\ \omega \cdot \delta t \end{bmatrix}$$

Finally, the control inputs are defined by the following formulas:

$$v = \frac{v_R + v_L}{2} \quad \omega = \frac{v_R - v_L}{L}$$

Where a positive angular velocity indicates the robot is turning left, otherwise it turns right. Additionally, when velocities are equal, the robot exhibits lower angular velocity if L is large, and higher otherwise.

3.3.1 Odometry

Given the known angular speed ω , the radius of the Instantaneous Center of Curvature (ICC) R , and the linear velocity v at each time instant, we can reconstruct the path and the final trajectory. This reconstruction of the path via integration is termed odometry.

The odometry of this robot involves computing the velocity in the base frame:

$$\begin{cases} v_x = v(t) \cos(\theta(t)) \\ v_y = v(t) \sin(\theta(t)) \end{cases}$$

Integrating the position in the base frame yields:

$$\begin{cases} x(t) = \int v(t) \cos(\theta(t)) \\ y(t) = \int v(t) \sin(\theta(t)) \\ \theta(t) = \int \omega(t) dt \end{cases}$$

Considering the velocities of both wheels v_R and v_L at a discrete time t' , we have:

$$\begin{cases} x(t) = \frac{1}{2} \int_0^t (v_R(t') + v_L(t')) \cos(\theta(t')) \\ y(t) = \frac{1}{2} \int_0^t (v_R(t') + v_L(t')) \sin(\theta(t')) \\ \theta(t) = \frac{1}{L} \int_0^t (v_R(t') - v_L(t')) dt' \end{cases}$$

Since computing these integrals is computationally intensive, we may resort to using approximations at discrete time instants.

Euler's integration When assuming constant linear velocity v_k and angular velocity ω_k within the time interval $[t_k, t_{k+1}]$, Euler integration can be employed to compute the robot odometry:

$$\begin{cases} x_{k+1} = x_k + v_k T_S \cos \theta_k \\ y_{k+1} = y_k + v_k T_S \sin \theta_k \\ \theta_{k+1} = \theta_k + \omega_k T_S \\ T_S = t_{k+1} - t_k \end{cases}$$

Here, the position $\{x_{k+1}, y_{k+1}\}$ is approximated, while the angle θ_{k+1} remains exact.

Runge-Kutta's integration Assuming constant linear velocity v_k and angular velocity ω_k within the time interval $[t_k, t_{k+1}]$, second-order Runge-Kutta integration can be used for computing robot odometry:

$$\begin{cases} x_{k+1} = x_k + v_k T_S \cos \left(\theta_k + \frac{\omega_k T_S}{2} \right) \\ y_{k+1} = y_k + v_k T_S \sin \left(\theta_k + \frac{\omega_k T_S}{2} \right) \\ \theta_{k+1} = \theta_k + \omega_k T_S \\ T_S = t_{k+1} - t_k \end{cases}$$

Although this method provides a better approximation, the orientation is not exact.

Exact integration When assuming constant linear velocity v_k and angular velocity ω_k within the time interval $[t_k, t_{k+1}]$, exact integration can be utilized to compute the robot odometry, resulting in:

$$\begin{cases} x_{k+1} = x_k + \frac{v_k}{\omega_k} (\sin \theta_{k+1} - \sin \theta_k) \\ y_{k+1} = y_k - \frac{v_k}{\omega_k} (\cos \theta_{k+1} - \cos \theta_k) \\ \theta_{k+1} = \theta_k + \omega_k T_S \\ T_S = t_{k+1} - t_k \end{cases}$$

In this case, all measurements are exact, but special attention must be given when the robot travels straight ($\omega \sim 0$). In such cases, Runge-Kutta integration should be used.

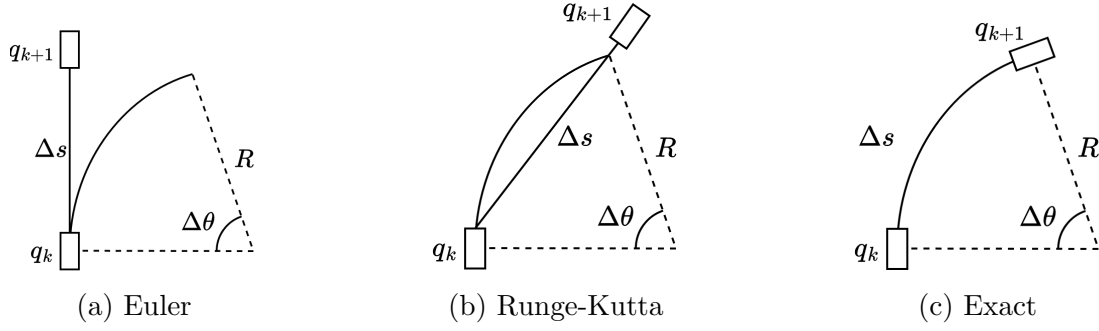


Figure 3.6: Integration techniques

These integration techniques serve different purposes depending on the system's frequency and the frequency at which parameters are checked. Euler and Runge-Kutta methods are suitable for high-frequency systems, while the exact approximation is necessary for low-frequency systems.

3.3.2 Sensors

Proprioceptive measurements are utilized to calculate linear velocity v_k and angular velocity ω_k :

$$\begin{cases} v_k T_S = \Delta s \\ \omega_k T_S = \Delta \theta \end{cases} \implies \frac{\Delta s}{\Delta \theta} = \frac{v_k}{\omega_k}$$

Here, Δs represents the distance traveled and $\Delta \theta$ denotes the change in orientation.

In a differential drive system, these quantities become:

$$\Delta s = \frac{r}{2} (\Delta \phi_R + \Delta \phi_L) \quad \Delta \theta = \frac{r}{L} (\Delta \phi_R - \Delta \phi_L)$$

Here, $\Delta \phi_R$ and $\Delta \phi_L$ correspond to the total rotations measured by wheel encoders.

It's important to note that this formula is applicable primarily for small-time frames, as larger ones introduce increased error due to various factors.

3.4 Synchronous drive

A synchronous drive robot is a sophisticated mechanical robot design incorporates three wheels for both propulsion and steering. It employs two motors: one for driving the wheels and another for steering them. All wheels are oriented in the same direction to ensure smooth movement. Additionally, the robot may include an extra actuator for precise control of angular velocity ω .

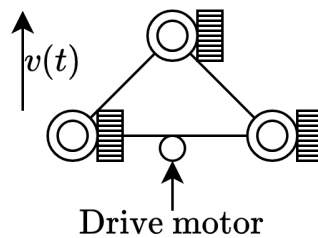


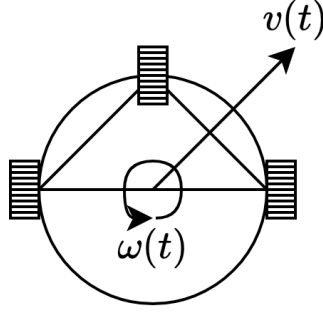
Figure 3.7: Synchronous drive robot

The robot's control variables consist of the linear velocity $v(t)$ and the angular velocity $\omega(t)$.

Its Instantaneous Center of Curvature remains at infinity, indicating non-holonomic behavior where the robot can only translate freely.

For the synchronous drive of the robot:

- Both linear velocity $v(t)$ and angular velocity $\omega(t)$ are directly controlled.
- Steering adjusts the direction of the Instantaneous Center of Curvature (ICC).



In this configuration, when $v(t) = 0$ and $\omega(t) = \omega$ at a specific time instant, the robot rotates in place. Conversely, when $v(t) = v$ and $\omega(t) = 0$ at a particular time instant, the robot moves linearly.

3.4.1 Odometry

To compute the velocity in the base frame, we use the following equations:

$$\begin{cases} v_x = v(t) \cos(\omega(t)) \\ v_y = v(t) \sin(\omega(t)) \end{cases}$$

Integrating the position in the base frame, we obtain the robot odometry:

$$\begin{cases} x(t) = \int_0^t \cos(\theta(t')) dt' \\ y(t) = \int_0^t \sin(\theta(t')) dt' \\ \theta(t) = \int_0^t \omega(t') dt' \end{cases}$$

3.5 Omnidirectional drive

The omnidirectional drive robot embodies a straightforward mechanical design featuring a minimum of three Swedish wheels, each powered by an independent motor. These wheels are strategically oriented in different directions, facilitating direct control over movements in x , y , and θ .

The robot's control parameters consist of the linear velocity $v(t)$ for each axis and the overall angular velocity $\omega(t)$.

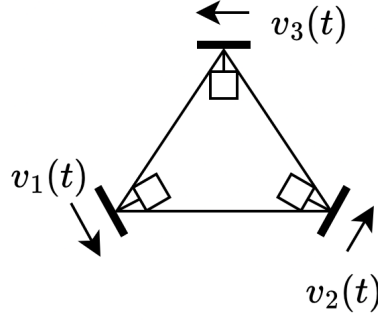


Figure 3.8: Omnidirectional drive robot

To manipulate the robot's position, the following matrix governs its motion:

$$\begin{bmatrix} v_x \\ v_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{\sqrt{3}}r & \frac{1}{\sqrt{3}}r \\ -\frac{2}{3}r & \frac{1}{3}r & \frac{1}{3}r \\ \frac{r}{3L} & \frac{r}{3L} & \frac{r}{3L} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

3.6 Tricycle drive

The Tricycle setup defines the kinematics of AGV as follows: it features a single actuated and steerable wheel along with two additional passive wheels. Independent control of θ is not possible unless $\alpha(t)$ can achieve up to 90 degrees. Additionally, the Instantaneous Center of Curvature (ICC) must align with the line passing through the fixed wheels.

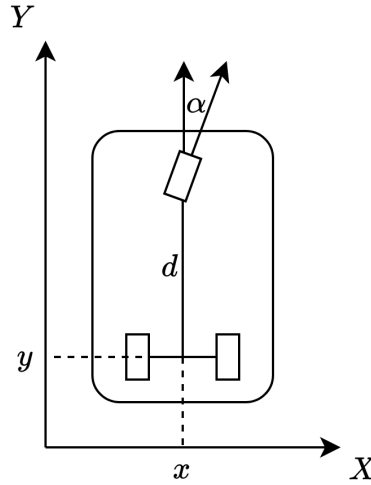


Figure 3.9: Tricycle drive robot

The robot's control parameters consist of the steering direction $\alpha(t)$ the angular velocity of the steering wheel $\omega_s(t)$.

In this configuration, when $\alpha(t) = 0$ and $\omega_s(t) = \omega$ at a specific time instant, the robot moves linearly. Conversely, when $\alpha(t) = 90^\circ$ and $\omega(t) = \omega$ at a particular time instant, the robot rotates in place.

3.6.1 Odometry

The direct kinematics can be derived as follows:

$$\begin{cases} r = \text{steering wheel radius} \\ V_S(t) = \omega_S(t)r \\ R(t) = d \tan\left(\frac{\pi}{2} - \alpha(t)\right) \\ \omega(t) = \frac{\omega_s(t)r}{\sqrt{d^2 + R(t)^2}} = \frac{V_S(t)}{d} \sin \alpha \end{cases}$$

In the base frame:

$$\begin{cases} \dot{x}(t) = V_S(t) \cos \alpha(t) + \cos \theta(t) = V(t) \cos \theta(t) \\ \dot{y}(t) = V_S(t) \sin \alpha(t) + \sin \theta(t) = V(t) \sin \theta(t) \\ \dot{\theta} = \frac{V_S(t)}{d} \sin \alpha(t) = \omega(t) \end{cases}$$

3.7 Ackerman steering drive

The most common form of kinematics worldwide involves vehicles equipped with four wheels in motion. These wheels are capable of turning but within certain limitations in their angles of rotation. Notably, this kinematic system does not incorporate in-place rotation.

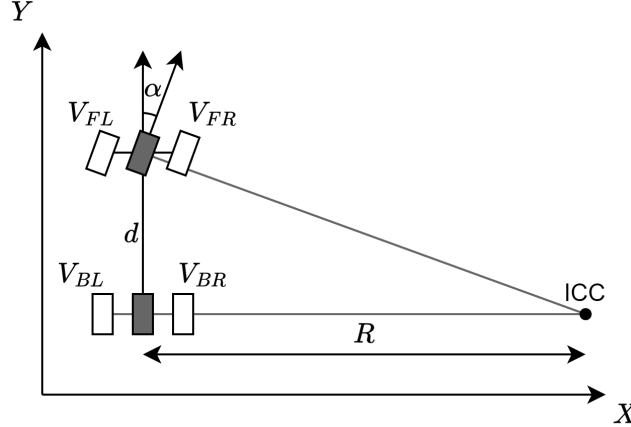


Figure 3.10: Ackerman steering drive robot

We simplify the model to resemble a bicycle:

$$\begin{cases} R = \frac{d}{\tan \alpha} \\ \frac{\omega d}{\sin \alpha} \end{cases}$$

This is with reference to the center of actual wheels:

$$\omega R = V \rightarrow \omega = V \frac{\tan \alpha}{d}$$

3.8 Skid steering drive

Vehicles equipped with tracks operate on a kinematic principle akin to the differential drive system. Each track's speed is individually controlled. The height of the track serves as the equivalent of wheel diameter. This configuration is commonly referred to as Skid Steering. However, it requires meticulous calibration and accurate modeling of slippage.

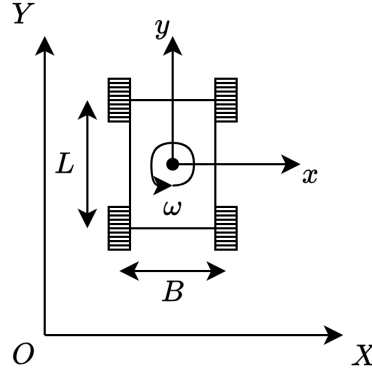


Figure 3.11: Skid steering drive robot

Let's make the following assumptions:

- The vehicle's mass is centralized.
- All wheels maintain contact with the ground.
- Wheels on the same side move at the same speed.

While in motion, we encounter multiple Instantaneous Centers of Rotation (ICRs), all sharing the same angular velocity ω_z :

$$\begin{cases} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = j_\omega \begin{bmatrix} \omega_l r \\ \omega_r r \end{bmatrix} \\ J_\omega = \frac{1}{y_l - y_r} \begin{bmatrix} -y_r & y_l \\ x_G & -x_G \\ -1 & - \end{bmatrix} \end{cases}$$

Assuming symmetry in the robot ($y_0 = y_l = -y_r$), we find:

$$\begin{cases} v_x = \frac{v_l + v_r}{2} \\ v_y = 0 \\ \omega_z = \frac{-v_l + v_r}{2y_0} \end{cases}$$

We can determine the instantaneous radius of curvature:

$$\begin{cases} R = \frac{v_G}{\omega_z} = \frac{v_l + v_r}{-v_l + v_r} y_0 \\ \lambda = \frac{v_l + v_r}{-v_l + v_r} \\ X = \frac{2y_0}{B} \end{cases}$$

3.9 Inverse kinematics

When faced with a desired position or velocity, several strategies can be employed to achieve it. While it's relatively straightforward to find some solution, determining the best solution can pose significant challenges. This best solution could be defined by various criteria such as:

- Shortest time to reach the goal.
- Most energy-efficient path or trajectory.
- Smoothest velocity profiles for comfortable operation.

Furthermore, if we encounter non-holonomic constraints and are limited to just two control variables, it becomes impossible to directly reach any of the three degrees of freedom final positions.

3.9.1 Differential drive

To tackle the problem effectively, we can decompose it and focus on controlling only a few degrees of freedom at a time:

1. Begin by turning the robot so that the wheels align parallel to the line between its original and final positions:

$$-v_L(t) = v_R(t) = v_{max}$$

2. Proceed to drive straight until the robot's origin coincides with the destination:

$$v_L(t) = v_R(t) = v_{max}$$

3. Finally, rotate again to achieve the desired final orientation:

$$-v_L(t) = v_R(t) = v_{max}$$

3.9.2 Synchro drive

To address the challenge systematically, we can break it down and manage only a select few degrees of freedom at each stage:

1. Initiate a turn to align the wheels parallel to the line connecting the robot's original and final positions:

$$\omega(t) = \omega_{max}$$

2. Proceed to drive straight until the robot's origin reaches the destination:

$$v(t) = v_{max}$$

3. Rotate once more to achieve the desired final orientation:

$$\omega(t) = \omega_{max}$$

CHAPTER 4

Robot localization

4.1 Introduction

Given a map, our objective is to determine the precise location of a robot within that map. This problem can be addressed by leveraging a sensor model, which characterizes $\Pr(z|x)$, representing the probability of obtaining a measurement z when the robot is positioned at x . Specifically, a scan z comprises K measurements:

$$z = \{z_1, z_2, \dots, z_K\}$$

Each individual measurement is statistically independent, conditioned on the robot's position and the surrounding map:

$$\Pr(z|x, m) = \prod_{k=1}^K \Pr(z_k|x, m)$$

Measurements may originate from various sources:

- Beams reflected by obstacles.
- Beams reflected by individuals or influenced by crosstalk.
- Random measurements.
- Maximum range measurements.

4.2 Beam sensor model

If laser measurements are utilized to determine distances between the robot and obstacles, the beam sensor model can be applied. This model treats each beam independently and integrates various sources of error:

1. *Measurement Noise*: each measurement is subject to noise, which is represented using a Gaussian distribution:

$$\Pr_{hit}(z|x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2} \frac{(z - z_{exp})^2}{b}}$$

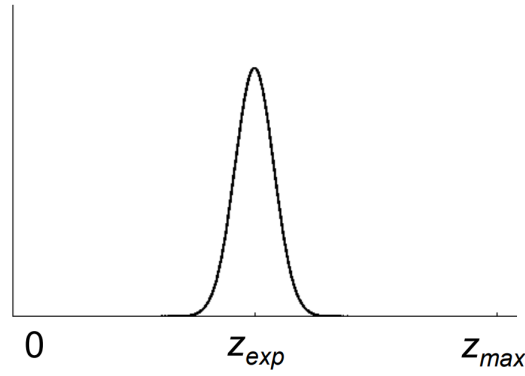


Figure 4.1: Measurement noise

2. *Unexpected obstacles*: measurements may deviate from the actual values due to temporary obstacles obstructing the beam's path. This probability is expressed as:

$$\Pr_{\text{unexp}}(z|x, m) = \begin{cases} \eta \lambda e^{-\lambda z} & z > z_{\text{exp}} \\ 0 & \text{otherwise} \end{cases}$$

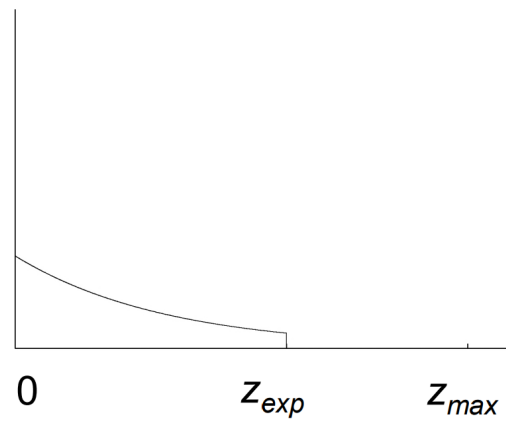


Figure 4.2: Unexpected obstacles

3. *Random measurement*: occasionally, a completely erroneous measurement may occur with a certain probability:

$$\Pr_{\text{rand}}(z|x, m) = \eta \frac{1}{z_{\text{max}}}$$

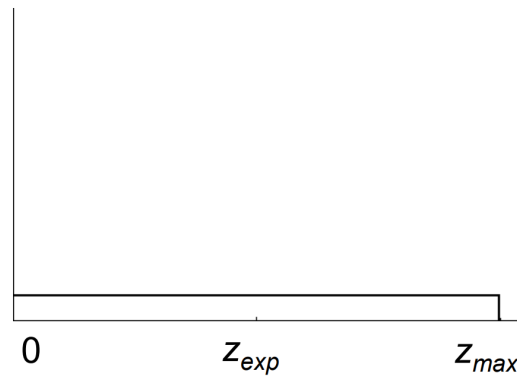


Figure 4.3: Random measurement

4. *Maximum range*: uncertainty arises from distances beyond the sensor's reach, which is modeled as:

$$\Pr_{\max}(z|x, m) = \eta \frac{1}{z_{\text{small}}}$$

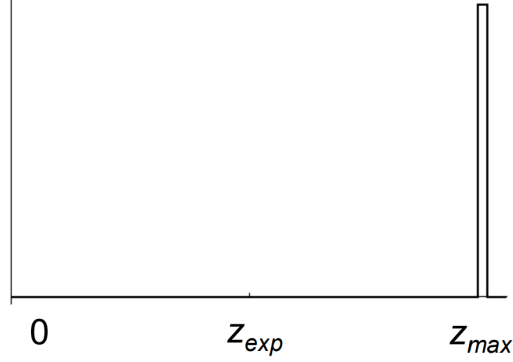


Figure 4.4: Random measurement

The total probability can be computed by combining these individual probabilities:

$$\Pr(z|x, m) = \begin{bmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{bmatrix}^T \cdot \begin{bmatrix} \Pr_{\text{hit}}(z|x, m) \\ \Pr_{\text{unexp}}(z|x, m) \\ \Pr_{\text{max}}(z|x, m) \\ \Pr_{\text{rand}}(z|x, m) \end{bmatrix}$$

The overall structure of this probability distribution is depicted as follows:

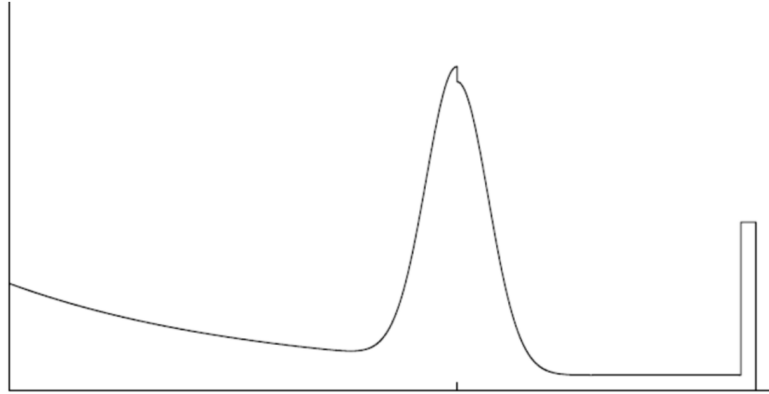


Figure 4.5: Probability distribution of beam sensor model

4.2.1 Calibration

To calibrate the sensor, we can collect data at specific distances, such as 300 cm and 400 cm, and then estimate the model parameters using maximum likelihood $\Pr(z|z_{\text{exp}})$. Since it's impractical to calibrate the sensor for every possible distance, we typically select a few representative distances and interpolate or use a mean for the missing data.

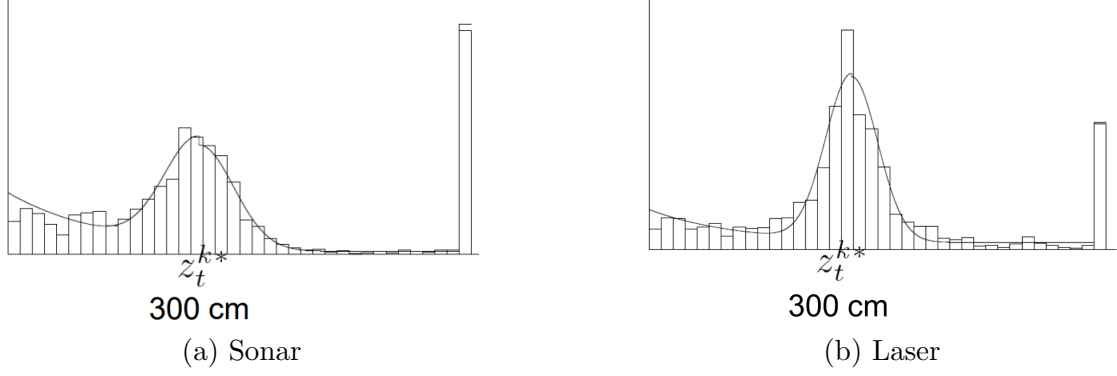


Figure 4.6: Calibration at three hundreds centimeters

By acquiring data at these distances, we can estimate the parameters of the sensor model to improve its accuracy across a range of distances.

4.2.2 Model likelihood

In this model, our objective is not to find the most likely position given the measurements, but rather to determine the position that maximizes the likelihood of the sensor readings. Therefore, in this context, we are searching for a position x such that the measurements from the sensors are maximized. This approach focuses on identifying the position that aligns best with the observed sensor data, rather than estimating the robot's actual position.

4.3 Scan sensor model

The beam sensor model, while assuming independence between beams and the physical causes of measurements, exhibits several issues:

- It tends to be overconfident due to its independence assumptions.
- Parameters need to be learned from data, adding complexity.
- A distinct model is required for different angles relative to obstacles, leading to increased complexity.
- It's inefficient as it relies on ray tracing for calculations.

To address these challenges, the scan sensor model simplifies the beam sensor model by:

- Utilizing a Gaussian distribution with the mean set at the distance to the closest obstacle.
- Employing a uniform distribution for random measurements.
- Incorporating a small uniform distribution for maximum range measurements.

In this model, we calculate the likelihood of encountering an obstacle along the trajectory of the ray.

From the occupancy grid map, that is the real map with the real obstacles, we can compute the likelihood field.

The likelihood field enables matching of various scans (except for sonars). It operates highly efficiently, relying solely on 2D tables. It maintains smoothness concerning minor shifts in robot position, facilitating gradient descent pose optimization. However, it disregards the physical attributes of beams.

Summary In highly dynamic environments, opting for the beam sensor model is crucial as it accounts for temporary obstacles, unlike the scan model. Conversely, in static environments, this approach proves faster due to precomputed maps.

4.4 Landmark model

Landmark sensors provide information on distance, bearing, or both. These measurements can be obtained through active beacons such as radio or GPS, or passive methods like visual or retro-reflective techniques. The standard approach for utilizing this data is triangulation.

Explicitly incorporating uncertainty into sensing processes is crucial for ensuring robustness:

1. Establish a parametric model for noise-free measurements.
2. Analyze sources of noise, such as distance and angle.
3. Introduce appropriate noise to parameters, possibly incorporating densities for noise distribution.
4. Learn and verify parameters by fitting the model to empirical data.

The likelihood of a measurement is determined by probabilistically comparing actual measurements with expected ones.

4.4.1 Landmark detection

The measurement $z = (i, d, \alpha)$ for a robot positioned at (x, y, θ) , relative to landmark i on map m (denoted as m_i), is expressed as follows:

$$\hat{d} = \sqrt{(m_x(i) - x)^2 + (m_y(i) - y)^2}$$

$$\hat{\alpha} = \arctan [2(m_y(i) - y, m_x(i) - x) - \theta]$$

The detection probability of a particular landmark may rely on either the distance or the bearing:

$$\Pr_{\text{det}} = \Pr(\hat{d} - d, \varepsilon_d) \Pr(\hat{\alpha} - \alpha, \varepsilon_\alpha)$$

Additionally, consideration for false positives is necessary:

$$z_{\text{det}} \Pr_{\text{det}} + z_{\text{fp}} \Pr_{\text{uniform}}(z|x, m)$$

4.5 Bayesian filter

As the robot navigates from an initial position to a target destination along a specific path, it's essential to integrate both its motion and sensor models effectively. Bayesian filtering provides a method to achieve this integration seamlessly.

The motion model encompasses the robot's pose Γ and the actions it takes u . Meanwhile, the sensor model incorporates the robot's pose Γ , observations z , and the map of the environment L .

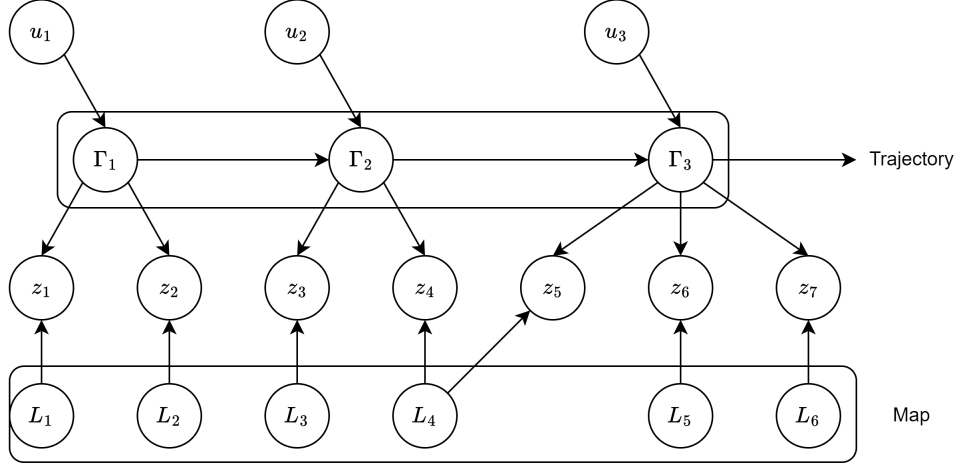


Figure 4.7: Localization framework

The filtering process aims to estimate the posterior probability distribution of the robot's state $x(t)$ given the stream of information about movement and sensors $d_t = \{u_1, z_1, \dots, u_t, z_t\}$ and the map of the environment m . We have the motion model which provides the probability distribution $\Pr(\Gamma_t | \Gamma_{t-1}, l_1, \dots, l_n)$ representing the robot's pose at time t given its past poses and the map. Our goal is to compute the belief $\text{Bel}(x_t) = P(x_t | u_1, z_1, \dots, u_t, z_t, m)$ representing the probability distribution of the robot's state x_t given all the available information. To compute $\text{Bel}(x_t)$, we typically use Bayes' rule, incorporating both the motion and sensor models. The motion model predicts the next state given the current state and action, while the sensor model updates this prediction based on the observations. This recursive process allows us to continuously refine our estimate of the robot's state as new information becomes available.

Assumptions We make the following assumptions, all of which are known:

- The prior probability of the system state $\Pr(x_0)$.
- The motion model $\Pr(x' | x, u)$, which describes the probability of transitioning from state x to state x' given action u .
- The sensor model $\Pr(z | x, m)$, which describes the probability of obtaining observation z given the system state x and the map m .

Markov assumption Additionally, we require the Markov assumption:

Property 4.5.1. A stochastic process possesses the Markov property if the conditional probability distribution of future states, conditioned on both past and present values, depends solely on the present state.

In other words, given the present state, the future states are independent of the past. Furthermore, we assume:

- Perfect model: the models accurately represent the system and environment.
- No approximation errors: the computations are exact.
- Static and stationary world: the environment does not change over time.
- Independent noise: the noise in sensor readings and system dynamics is independent and identically distributed.

4.5.1 Bayes filter

With the given assumptions, we can derive the belief on the robot's position using Bayes' theorem:

$$\begin{aligned}\text{Bel}(x_t) &= \Pr(x_t, z_1, \dots, u_t, z_t, m) \\ &= \eta \Pr(z_t | x_t, u_1, z_1, \dots, u_t, m) \Pr(x_t | u_1, z_1, \dots, u_t, m)\end{aligned}$$

Using the Markov property and Bayes' rule, we simplify further:

$$\text{Bel}(x_t) = \eta \Pr(z_t | x_t, m) \Pr(x_t | u_1, z_1, \dots, u_t, m)$$

Expanding the second term using the recursive Bayes filter equation:

$$\text{Bel}(x_t) = \eta \Pr(z_t | x_t, m) \int \Pr(x_t | u_1, z_1, \dots, u_t, x_{t-1}, m) \Pr(x_{t-1} | u_1, z_1, \dots, u_{t-1}, m) dx_{t-1}$$

Applying the Markov property again and simplifying:

$$\text{Bel}(x_t) = \eta \Pr(z_t | x_t, m) \int \Pr(x_t | u_t, x_{t-1}) \Pr(x_{t-1} | u_1, z_1, \dots, u_{t-1}, m) dx_{t-1}$$

Now, recognizing that the belief at x_{t-1} is the same as $\text{Bel}(x_{t-1})$, we can rewrite the integral:

$$\text{Bel}(x_t) = \eta \Pr(z_t | x_t, m) \int \Pr(x_t | u_t, x_{t-1}) \text{Bel}(x_{t-1}) dx_{t-1}$$

In this expression, z represents an observation, u represents an action, x represents a state, and m represents a map. This recursive formulation allows us to iteratively update our belief about the robot's position based on new sensor readings and actions.

4.5.2 Bayes filter algorithm

The Bayes filter algorithm computes the belief of a certain position x given perceptual data z or action data u . If the map is given, the belief at x_t is updated using the formula:

$$\text{Bel}(x_t | m) = \eta \Pr(z_t | x_t, m) \int \Pr(x_t | u_t, x_{t-1}, m) \text{Bel}(x_{t-1} | m) dx_{t-1}$$

Here's the Bayes filter algorithm:

Algorithm 1 Bayes filter algorithm

```

1: if  $d$  is a perceptual data item  $z$  then
2:   for all  $x$  do
3:      $\text{Bel}'(x) = \Pr(z|x)\text{Bel}(x)$ 
4:   end for
5:   normalize  $\text{Bel}'(x)$ 
6: else if  $d$  is an action data item  $u$  then
7:   for all  $x$  do
8:      $\text{Bel}'(x) = \int \Pr(x|u, x')\text{Bel}(x') dx'$ 
9:   end for
10: end if
11: return  $\text{Bel}'(x)$ 

```

This algorithm updates the belief state based on either a perceptual data item z or an action data item u . If d is a perceptual data item, it computes the product of the probability of observation given each state and the current belief state, then normalizes the result. If d is an action data item, it integrates over the motion model to update the belief state.

Based on this representation, various filtering algorithms can be implemented, including discrete filters, Kalman filters, sigma-point filters, and particle filters. These algorithms differ in how they represent and update the belief state and handle uncertainties in the system and sensor measurements.

4.5.3 Discrete Bayes filter algorithm

The discrete Bayes filter algorithm provides a method for updating the belief state when the map is discretized into regions. Here's the algorithm:

Algorithm 2 Discrete Bayes filter algorithm

```

1:  $h = 0$ 
2: if  $d$  is a perceptual data item  $z$  then
3:   for all  $x$  do
4:      $\text{Bel}'(x) = \Pr(z|x)\text{Bel}(x)$ 
5:      $\eta = \eta + \text{Bel}'(x)$ 
6:   end for
7:   for all  $x$  do
8:      $\text{Bel}'(x) = \eta^{-1}\text{Bel}'(x)$ 
9:   end for
10:  normalize  $\text{Bel}'(x)$ 
11: else if  $d$  is an action data item  $u$  then
12:   for all  $x$  do
13:      $\text{Bel}'(x) = \sum_{x'} \Pr(x|u, x')\text{Bel}(x')$ 
14:   end for
15: end if
16: return  $\text{Bel}'(x)$ 

```

This algorithm updates the belief state based on either a perceptual data item z or an action data item u :

- When the data item d is a perceptual input z , the algorithm computes the product of the probability of observation given each state and the current belief state. Then, it normalizes the resulting belief to ensure it sums to one.
- When d represents an action u , the algorithm updates the belief state by summing over all possible previous states x' weighted by the transition probabilities $\Pr(x|u, x')$.

The algorithm facilitates belief update upon sensory input and normalization, iterating over all cells. For efficiency, especially when the belief is peaked (e.g., during position tracking), it's advisable to avoid updating irrelevant parts. Instead, focus on updating relevant components based on the likelihood of observations given the active components.

For updating the belief upon robot motion, the algorithm assumes a bounded Gaussian model to reduce the computational complexity from $O(n^2)$ to $O(n)$. This involves shifting the data in the grid according to the measured motion and then convolving the grid using a Gaussian kernel.

4.6 Kalman filter

The belief can be computed as:

$$\text{Bel}(x_t|m) = \eta \Pr(z_t|x_t, m) \int \Pr(x_t|u_t, x_{t-1}, m) \text{Bel}(x_{t-1}|m) dx_{t-1}$$

It can be partially computed using the following two formulas:

$$\begin{cases} \bar{\text{Bel}}(x_t|m) = \int \Pr(x_t|u_t, x_{t-1}, m) \text{Bel}(x_{t-1}|m) dx_{t-1} \\ \text{Bel}(x_t|m) = \eta \Pr(z_t|x_t, m) \bar{\text{Bel}}(x_t|m) \end{cases}$$

Note that the variable η is also part of an integral. While it's impossible to compute these integrals in closed form for continuous distributions, it's feasible to do so at least for Gaussian distributions.

Univariate Gaussian distribution The univariate Gaussian distribution, denoted as $X \sim \mathcal{N}(\mu, \sigma^2)$, is described as follows:

$$\Pr(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

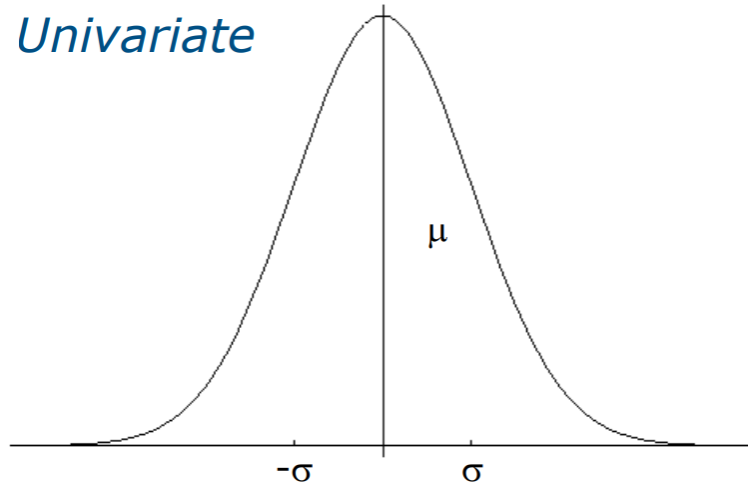
Univariate

Figure 4.8: Univariate Gaussian distribution

Given an univariate Gaussian distribution $X \sim \mathcal{N}(\mu, \sigma^2)$, the distribution $Y = aX + b$ is still Gaussian and is defined as:

$$Y \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$$

Given two univariate Gaussian distributions $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, the product of these distributions is:

$$\Pr(X_1) \cdot \Pr(X_2) \sim \mathcal{N}\left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\mu_2, \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}\right)$$

Multivariate Gaussian distribution The multivariate Gaussian distribution, denoted as $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, is described as follows:

$$\Pr(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})^T}$$

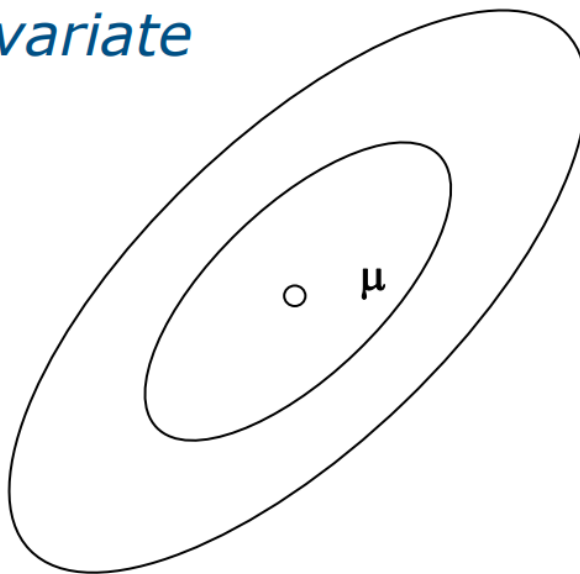
Multivariate

Figure 4.9: Multivariate Gaussian distribution

Given a multivariate Gaussian distribution $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the distribution $Y = \mathbf{A}X + \mathbf{B}$ is still Gaussian and is defined as:

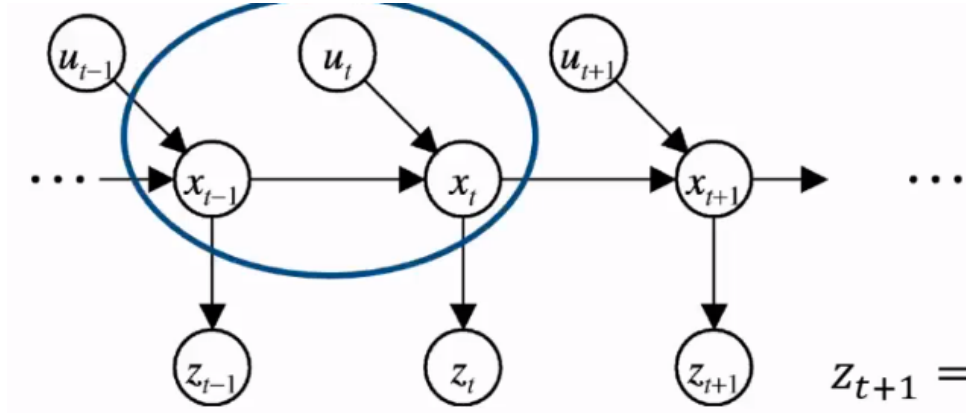
$$Y \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{B}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T)$$

Given two multivariate Gaussian distributions $X_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $X_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, the product between these distributions is:

$$\Pr(X_1) \cdot \Pr(X_2) \sim \mathcal{N}\left(\frac{\boldsymbol{\Sigma}_2}{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}\boldsymbol{\mu}_1 + \frac{\boldsymbol{\Sigma}_1}{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}\boldsymbol{\mu}_2, \frac{1}{\frac{1}{\boldsymbol{\Sigma}_1} + \frac{1}{\boldsymbol{\Sigma}_2}}\right)$$

4.6.1 Discrete time Kalman filter

Consider the scenario depicted below, where z represents observations, x denotes position, and u signifies the action performed:



Let's define the observation as a linear function of the position:

$$z_t = \mathbf{C}_t x_t + \delta_t$$

Here, C_t is a $k \times n$ matrix mapping the state x_t to observation z_t , and δ_t is a random variable representing independent and normally distributed process and measurement noise, with covariance Q_t .

Similarly, we can model the position of the robot (motion model) as:

$$x_t = \mathbf{A}_t x_{t-1} + \mathbf{B}_t u_t + \varepsilon_t$$

Here, A_t is an $n \times n$ matrix describing state evolution from time $t - 1$ to t without controls or noise, B_t is an $n \times l$ matrix illustrating how control u_t affects the state from time $t - 1$ to t , and ε_t is a random variable representing independent and normally distributed process and measurement noise, with covariance R_t .

The initial belief is normally distributed:

$$\text{Bel}(x_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$$

We assume that observations are linear functions of the state, with additive noise. Thus, the probability of having a certain observation becomes:

$$\Pr(z_t | x_t) = \mathcal{N}(z_t; \mathbf{C}_t x_t, \mathbf{Q}_t)$$

Similarly, we assume that states are linear functions of the previous state and control, with additive noise. Consequently, the probability of being in a certain state becomes:

$$\Pr(x_t | u_t, x_{t-1}) = \mathcal{N}(x_t; \mathbf{A}_t x_{t-1} + \mathbf{B}_t u_t, \mathbf{R}_t)$$

Closed form prediction With these assumptions, we can predict the position as follows:

$$\begin{aligned}\bar{\text{Bel}}(x_t) &= \int \Pr(x_t|u_t, x_{t-1}) \cdot \text{Bel}(x_{t-1}) dx_{t-1} \\ &= \int \mathcal{N}(x_t; \mathbf{A}_t x_{t-1} + \mathbf{B}_t u_t, \mathbf{R}_t) \cdot \mathcal{N}(z_t; \mathbf{C}_t x_t, \mathbf{Q}_t) dx_{t-1} \\ &= \eta \int e^{-\frac{1}{2}(x_t - \mathbf{A}_t x_{t-1} - \mathbf{B}_t u_t) \mathbf{R}_t^{-1} (x_t - \mathbf{A}_t x_{t-1} - \mathbf{B}_t u_t)^T} \cdot e^{-\frac{1}{2}(x_{t-1} - \bar{\boldsymbol{\mu}}_{t-1}) \bar{\boldsymbol{\Sigma}}_{t-1}^{-1} (x_{t-1} - \bar{\boldsymbol{\mu}}_{t-1})^T} dx_{t-1}\end{aligned}$$

In the end, the closed-form prediction step is equal to:

$$\bar{\text{Bel}}(x_t) = \begin{cases} \bar{\boldsymbol{\mu}}_t = \mathbf{A}_t \bar{\boldsymbol{\mu}}_{t-1} + \mathbf{B}_t u_t \\ \bar{\boldsymbol{\Sigma}}_t = \mathbf{A}_t \bar{\boldsymbol{\Sigma}}_{t-1} \mathbf{A}_t^T + \mathbf{R}_t \end{cases}$$

Closed form correction With these assumptions, we can correct the position as follows:

$$\begin{aligned}\text{Bel}(x_t) &= \eta \Pr(z_t|x_t) \cdot \bar{\text{Bel}}(x_t) \\ &= \eta \mathcal{N}(z_t; \mathbf{C}_t x_t, \mathbf{Q}_t) \mathcal{N}(x_t; \bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t) \\ &= \eta \int e^{-\frac{1}{2}(z_t - \mathbf{C}_t x_t) \mathbf{Q}_t^{-1} (z_t - \mathbf{C}_t x_t)^T} \cdot e^{-\frac{1}{2}(x_t - \bar{\boldsymbol{\mu}}_t) \bar{\boldsymbol{\Sigma}}_t^{-1} (x_t - \bar{\boldsymbol{\mu}}_t)^T} dx_{t-1}\end{aligned}$$

In the end, the closed-form update step is equal to:

$$\text{Bel}(x_t) = \begin{cases} \mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^T (\mathbf{C}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^T + \mathbf{Q}_t)^{-1} \\ \boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t (z_t - \mathbf{C}_t \bar{\boldsymbol{\mu}}_t) \\ \boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t) \bar{\boldsymbol{\Sigma}}_t \end{cases}$$

Algorithm With these steps, we can finally construct the Kalman algorithm:

Algorithm 3 Kalman filter algorithm

- | | |
|--|-------------------|
| 1: $\bar{\boldsymbol{\mu}}_t = \mathbf{A}_t \bar{\boldsymbol{\mu}}_{t-1} + \mathbf{B}_t u_t$ | ▷ Prediction step |
| 2: $\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}_t \bar{\boldsymbol{\Sigma}}_{t-1} \mathbf{A}_t^T + \mathbf{R}_t$ | |
| 3: $\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^T (\mathbf{C}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^T + \mathbf{Q}_t)^{-1}$ | ▷ Correction step |
| 4: $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t (z_t - \mathbf{C}_t \bar{\boldsymbol{\mu}}_t)$ | |
| 5: $\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t) \bar{\boldsymbol{\Sigma}}_t$ | |
| 6: return $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$ | |
-

The complexity is polynomial in measurement dimensionality k and state dimensionality n : $O(k^{2.376} + n^2)$. It is optimal for linear Gaussian systems.

However, most robotics systems are nonlinear, and the Kalman filter represents unimodal distributions, limiting its applicability in such cases.

4.6.2 Extended Kalman filter

In linear systems, Gaussian noise is assumed to follow:

$$\begin{cases} x_t = \mathbf{A}_t x_{t-1} + \mathbf{B}_t u(t) + \varepsilon_t \\ z_t = \mathbf{C}_t x_t + \delta_t \end{cases}$$

However, in non-linear systems, Gaussian noise is distributed as:

$$\begin{cases} x_t = g(u_t, x_{t-1}) \\ z_t = h(x_t) \end{cases}$$

Here, $g(\cdot)$ and $h(\cdot)$ are nonlinear functions.

Prediction step In the case of nonlinear functions, the prediction is computed as:

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \boldsymbol{\mu}_{t-1}) + \frac{\partial g(u_t, \boldsymbol{\mu}_{t-1})}{\partial x_{t-1}}(x_{t-1} - \boldsymbol{\mu}_{t-1}) \\ &\approx g(u_t, \boldsymbol{\mu}_{t-1}) + \mathbf{G}_t(x_{t-1} - \boldsymbol{\mu}_{t-1}) \end{aligned}$$

Here, \mathbf{G}_t represents the Jacobian matrix of $g(\cdot)$ evaluated at $(\boldsymbol{\mu}_{t-1}, u(t))$.

Correction step In case of nonlinear functions we have that the correction is computed as:

$$\begin{aligned} h(x_t) &\approx h(\bar{\boldsymbol{\mu}}_t) + \frac{\partial h(\bar{\boldsymbol{\mu}}_t)}{\partial x_t}(x_t - \bar{\boldsymbol{\mu}}_t) \\ &\approx h(\bar{\boldsymbol{\mu}}_t) + \mathbf{H}_t(x_t - \bar{\boldsymbol{\mu}}_t) \end{aligned}$$

Here, \mathbf{H}_t represents the Jacobian matrix of $h(\cdot)$ evaluated at $\bar{\boldsymbol{\mu}}_t$.

Algorithm With these steps, we can finally construct the Extended Kalman Filter algorithm:

Algorithm 4 Extended Kalman filter algorithm

- | | |
|--|-------------------|
| 1: $\bar{\boldsymbol{\mu}}_t = g(u_t, \boldsymbol{\mu}_{t-1})$ | ▷ Prediction step |
| 2: $\bar{\boldsymbol{\Sigma}}_t = \mathbf{G}_t \boldsymbol{\Sigma}_{t-1} \mathbf{G}_t^T + \mathbf{R}_t$ | |
| 3: $\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^T + \mathbf{Q}_t)^{-1}$ | ▷ Correction step |
| 4: $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(z_t - h(\bar{\boldsymbol{\mu}}_t))$ | |
| 5: $\boldsymbol{\Sigma}_t = (I - \mathbf{K}_t \mathbf{H}_t) \bar{\boldsymbol{\Sigma}}_t$ | |
| 6: return $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$ | |
-

Here, $\mathbf{G}_t = \frac{\partial g(u_t, \boldsymbol{\mu}_{t-1})}{\partial x_{t-1}}$, and $\mathbf{H}_t = \frac{\partial h(\bar{\boldsymbol{\mu}}_t)}{\partial x_t}$. Extended Kalman Filter indeed exhibits polynomial complexity in measurement dimension k and state dimension n , typically $O(k^{2.376} + n^2)$. However, it's not optimal and can diverge if nonlinearities are significant. Despite its shortcomings, the Extended Kalman Filter often performs well even when some assumptions are violated.

As alternatives, methods like the Unscented Kalman Transform are employed, offering more robustness in handling nonlinearity and uncertainties.

4.7 Particle filter

Particle Filters are a methodology utilized for estimating non-Gaussian and nonlinear processes by representing belief through random samples. This technique, also known as Monte Carlo filter, embodies the concept of survival of the fittest, where the most informative particles prevail. Through iterations, the process of condensation refines the particle distribution, enhancing accuracy. Another term often used interchangeably is the Bootstrap filter, highlighting the resampling technique at its core.

The computation of belief can be expressed as:

$$\text{Bel}(x_t) = \eta \Pr(z_t|x_t) \int \Pr(x_t|x_{t-1}, u_{t-1}) \cdot \text{Bel}(x_{t-1}) dx_{t-1}$$

For the particle filter we select:

1. Select a sample x_{t-1}^i from $\text{Bel}(x_{t-1})$.
2. Choose a sample x_t^i from $\Pr(x_t|x_{t-1}^i, u_{t-1})$.
3. Compute the importance factor w_t^i for the sample x_t^i as:

$$w_t^i = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{\eta \Pr(z_t|x_t) \Pr(x_t|x_{t-1}^i, u_{t-1}) \text{Bel}(x_{t-1}^i)}{\Pr(x_t|x_{t-1}^i, u_{t-1}) \text{Bel}(x_{t-1}^i)} \propto \Pr(z_t|x_t)$$

Algorithm The particle filter algorithm can be executed in the following manner:

Algorithm 5 Particle filter algorithm

```

1:  $S_t = \emptyset$ 
2:  $\eta = 0$ 
3: for  $i = 1$  to  $n$  do                                     ▷ Generate new samples
4:   Sample index  $j(i)$  from the discrete distribution given by  $w_{t-1}$ 
5:   Sample  $x_t^i$  from  $\Pr(x_t|x_{t-1}, u_{t-1})$  using  $x_{t-1}^{j(i)}$  and  $u_{t-1}$ 
6:    $w_t^i = \Pr(z_t|x_t^i)$                                      ▷ Compute importance weight
7:    $\eta = \eta + w_t^i$                                          ▷ Update normalization factor
8:    $S_t = S_t \cup \{\langle x_t^i, w_t^i \rangle\}$                        ▷ Insert
9: end for
10: for  $i = 1$  to  $n$  do
11:    $w_t^i = \frac{w_t^i}{\eta}$                                      ▷ Normalize weights
12: end for

```

Monte Carlo localization Monte Carlo localization, also known as particle filter localization, is an algorithm utilized by robots to determine their position and orientation within an environment. This method leverages a particle filter to estimate the robot's pose as it moves and gathers sensory information within a mapped environment.