Localization for Autonomous Driving

Brief tutorial to have fun with localization

Prepared by Oussama Code by Claude.ai

January 2025

I, Oussama EL HAMZAOUI confirm that the work presented in this report is my own (with the help of Claude.ai). Where information has been derived from other sources, I confirm that this has been indicated in the report.

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam et turpis gravida, lacinia ante sit amet, sollicitudin erat. Aliquam efficitur vehicula leo sed condimentum. Phasellus lobortis eros vitae rutrum egestas. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec at urna imperdiet, vulputate orci eu, sollicitudin leo. Donec nec dui sagittis, malesuada erat eget, vulputate tellus. Nam ullamcorper efficitur iaculis. Mauris eu vehicula nibh. In lectus turpis, tempor at felis a, egestas fermentum massa.

Acknowledgements

Interdum et malesuada fames ac ante ipsum primis in faucibus. Aliquam congue fermentum ante, semper porta nisl consectetur ut. Duis ornare sit amet dui ac faucibus. Phasellus ullamcorper leo vitae arcu ultricies cursus. Duis tristique lacus eget metus bibendum, at dapibus ante malesuada. In dictum nulla nec porta varius. Fusce et elit eget sapien fringilla maximus in sit amet dui.

Mauris eget blandit nisi, faucibus imperdiet odio. Suspendisse blandit dolor sed tellus venenatis, venenatis fringilla turpis pretium. Donec pharetra arcu vitae euismod tincidunt. Morbi ut turpis volutpat, ultrices felis non, finibus justo. Proin convallis accumsan sem ac vulputate. Sed rhoncus ipsum eu urna placerat, sed rhoncus erat facilisis. Praesent vitae vestibulum dui. Proin interdum tellus ac velit varius, sed finibus turpis placerat.

Contents

List of Figures

List of Tables

Abbreviations

ADAS	**A**dvanced **D**river **A**ssistance **S**ystems
CKF	**C**ubature **K**alman **F**ilter
EKF	**E**xtended **K**alman **F**ilter
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
HD	**H**igh **D**efinition (as in HD maps)
IMU	**I**nertial **M**easurement **U**nit
KF	**K**alman **F**ilter
LIDAR	**L**ight **D**etection and **R**anging
MCL	**M**onte **C**arlo **L**ocalization
PF	**P**article **F**ilter
RADAR	**R**adio **D**etection and **R**anging
RBPF	**R**ao-**B**lackwellized **P**article **F**ilter
SLAM	**S**imultaneous **L**ocalization and **M**apping
UKF	**U**nscented **K**alman **F**ilter
V2X	**V**ehicle-to-Everything Communication

Overview

Course chapters

Section to be deleted after completion of the couse

- ☒ Introduction to Localization
  - ☒ Problem statement and motivation
  - ☒ Types of localization problems
  - ☒ State estimation challenges
  - ☒ Sensor types and characteristics
  - ☒ Sources of uncertainty in robotics
- ☒ Probability Theory Foundations
  - ☒ Random variables and probability distributions
  - ☒ Bayes' theorem
  - [-] Conditional probability
  - [-] Markov assumption
  - ☐ Joint and marginal probabilities
  - ☐ Gaussian distributions
- ☐ Bayesian Filtering Framework
  - ☐ Recursive state estimation
  - ☐ Prediction step (motion model)
  - ☐ Update step (measurement model)
  - ☐ Chapman-Kolmogorov equation
  - ☐ Bayes filter algorithm
  - ☐ Linear vs nonlinear systems
- ☐ Kalman Filtering
  - ☐ Linear Kalman Filter
    - ☐ System model and assumptions
    - ☐ Prediction equations
    - ☐ Update equations
    - ☐ Uncertainty propagation
  - ☐ Extended Kalman Filter (EKF)
    - ☐ Linearization process
    - ☐ Jacobian matrices
    - ☐ Algorithm implementation
  - ☐ Unscented Kalman Filter (UKF)
    - ☐ Sigma points
    - ☐ Unscented transform
    - ☐ Algorithm implementation
- ☐ Particle Filtering
  - ☐ Monte Carlo methods

- ☐ Importance sampling
    - ☐ Particle representation
    - ☐ Sequential Importance Sampling (SIS)
    - ☐ Resampling techniques
    - ☐ Sample degeneracy and impoverishment
    - ☐ Adaptive particle filtering
- ☐ Advanced Topics
    - ☐ Multi-hypothesis tracking
    - ☐ SLAM basics
    - ☐ Sensor fusion techniques
    - ☐ Loop closure
    - ☐ Global vs local localization
- ☐ MATLAB Implementation: Kalman Filter
    - ☐ Linear KF implementation
        - ☐ State prediction
        - ☐ Measurement update
        - ☐ Covariance propagation
    - ☐ EKF implementation
        - ☐ System modeling
        - ☐ Jacobian computation
        - ☐ Filter implementation
    - ☐ Visualization and analysis
    - ☐ Performance evaluation
- ☐ MATLAB Implementation: Particle Filter
    - ☐ Basic PF framework
    - ☐ Particle initialization
    - ☐ Motion model implementation
    - ☐ Measurement model
    - ☐ Weight computation
    - ☐ Resampling implementation
    - ☐ Visualization tools
    - ☐ Performance metrics
- ☐ Practical Applications
    - ☐ Vehicle localization case studies
    - ☐ Robot navigation examples
    - ☐ Integration with mapping
    - ☐ Real-world challenges
    - ☐ Best practices and optimization
- ☐ Project Work
    - ☐ Implementation exercises
    - ☐ Real dataset analysis
    - ☐ Performance comparison of different filters
    - ☐ Parameter tuning
    - ☐ Documentation and presentation

Objectives

Chapter 1

Introduction to Localization

- ☒ Introduction to Localization
  - ☒ Problem statement and motivation
  - ☒ Types of localization problems
  - ☒ State estimation challenges
  - ☒ Sensor types and characteristics
  - ☒ Sources of uncertainty in robotics

1.1 Problem Statement and Motivation

In robotics and autonomous systems, localization addresses a fundamental question: "Where am I?" This seemingly simple question underlies many complex challenges in autonomous navigation. Imagine waking up in an unfamiliar room – you would use visual cues, memory, and perhaps a map to determine your location. Robots face a similar challenge, but must solve it using sensors and algorithms rather than human intuition.

Localization serves as the cornerstone of autonomous navigation. Without accurate knowledge of its position, a robot cannot effectively plan paths, avoid obstacles, or complete assigned tasks. This becomes particularly critical in applications like autonomous vehicles, where position errors of even a few centimeters can have serious consequences.

1.2 Types of Localization Problems

We can categorize localization problems based on their initial conditions and objectives:

Position Tracking represents the simplest case, where we know the initial position and need to maintain an accurate estimate as the robot moves. Think of using GPS in your car – you start from a known location and track your movement.

Global Localization presents a more challenging scenario where the initial position is unknown. The robot must determine its position from scratch using available sensor information and a map. This is analogous to opening a ride-sharing app in an unfamiliar city and waiting for it to locate you.

The Kidnapped Robot Problem is the most challenging variant, where a well-localized robot is suddenly transported to an unknown location. While this may seem artificial, it tests a system's ability to recover from catastrophic failures or sensor malfunctions.

1.3 Sensor Types and Characteristics

Localization systems typically rely on multiple sensor types, each with distinct advantages and limitations:

- Proprioceptive Sensors measure internal state changes, such as wheel encoders that track rotation or inertial measurement units (IMUs) that detect acceleration and angular velocity. While these sensors provide high-frequency updates, they suffer from cumulative errors through a process called dead reckoning.
- Exteroceptive Sensors observe the external environment. These include:
  - LIDAR (Light Detection and Ranging) which creates detailed 3D scans of surroundings
  - Cameras that provide rich visual information but require sophisticated processing
  - RADAR which offers reliable distance measurements even in adverse weather
  - GNSS (Global Navigation Satellite System) which provides absolute position but may suffer from urban canyon effects and multipath errors

1.4 Sources of Uncertainty

Understanding uncertainty is crucial for robust localization. Several factors contribute to localization uncertainty:

Motion Uncertainty arises from imperfect robot control and environmental interactions. When a robot moves, wheel slippage, uneven terrain, and mechanical play all introduce errors between commanded and actual motion.

Measurement Uncertainty stems from sensor limitations and noise. For example, LIDAR measurements might be affected by reflective surfaces, while camera images can be distorted by varying lighting conditions.

Environmental Uncertainty relates to the dynamic nature of the real world. Moving objects, changing weather conditions, and modifications to the environment can all affect localization accuracy.

Model Uncertainty comes from our simplified representations of complex physical systems. Our mathematical models of robot motion and sensor behavior are approximations that introduce additional uncertainty.

1.5 The Role of Probability Theory

Given these uncertainties, deterministic approaches to localization often fail in real-world conditions. This necessitates a probabilistic framework that can: - Represent and propagate uncertainty through mathematical models - Fuse information from multiple, imperfect sensors - Handle conflicting measurements and outliers - Provide confidence estimates along with position estimates

This probabilistic approach leads us naturally to the Bayesian filtering framework, which we'll explore in subsequent chapters. The framework provides a mathematical foundation for combining prior knowledge, motion predictions, and sensor measurements to maintain an estimate of the robot's position over time.

Understanding these foundational concepts is crucial as we progress to more advanced topics in localization. The challenges and considerations introduced here will inform our discussion of specific algorithms and implementations throughout the course.

Would you like me to elaborate on any of these sections or move on to the probability theory foundations?

Chapter 2

Probability Theory Foundations

2.1 Random variables and probability distributions

2.1.1 Random Variables

A random variable is a mathematical way to describe outcomes of a random process. Think of it as a function that assigns a numerical value to each possible outcome of an experiment or observation.

Let's consider a practical example from robotics: imagine a robot's sensor measuring the distance to a wall. Even when the robot and wall are stationary, repeated measurements might give slightly different values due to sensor noise. Each measurement is a realization of a random variable that we could call "measured distance."

Random variables come in two main types:

Discrete random variables can only take specific, countable values. For instance, if we count the number of landmarks a robot sees in its field of view, this would be a discrete random variable - we can only see 0, 1, 2, or some whole number of landmarks.

Continuous random variables can take any value within a continuous range. Most sensor measurements in robotics are continuous random variables. Our distance sensor example could theoretically return any real number within its measurement range.

2.1.2 Probability Distributions

A probability distribution describes how likely each possible value of a random variable is to occur. It tells us the complete story of the random variable's behavior.

For discrete random variables, we use a Probability Mass Function (PMF). The PMF gives the probability of each possible value directly. For example, if we're counting landmarks:

- P(X = 0) = 0.1 (10% chance of seeing no landmarks)
- P(X = 1) = 0.3 (30% chance of seeing exactly one landmark)
- P(X = 2) = 0.4 (40% chance of seeing exactly two landmarks)

And so on…

For continuous random variables, we use a Probability Density Function (PDF). The PDF works differently because with continuous variables, the probability of getting any exact value is actually zero! Instead, the PDF gives us the relative likelihood of values occurring, and we integrate it over ranges to get probabilities.

The most important continuous probability distribution in robotics is the Gaussian (or Normal) distribution. It's defined by two parameters:

- (mu): the mean, representing the central value
- (sigma): the standard deviation, representing the spread

The Gaussian distribution appears naturally in many robotics scenarios because of the Central Limit Theorem. When many small random effects add up - like multiple sources of sensor noise - their combined effect tends to follow a Gaussian distribution.

In the context of localization, probability distributions help us represent:

1. The robot's belief about its position (often as a Gaussian in simple cases)
2. Uncertainty in sensor measurements
3. Noise in motion commands and their execution
4. The likelihood of different measurements given a particular position

Understanding these distributions is crucial because localization algorithms like Kalman filters and particle filters essentially manipulate these probability distributions to maintain and update the robot's position estimate over time.

2.2 Bayes theorem

Let me explain Bayes' theorem in a way that will make intuitive sense, starting with a simple example and then building up to its use in robotics.

Imagine you're a robot in a room, and you have a simple distance sensor. Sometimes your sensor shows a reading of 2 meters, but you're not sure if you're actually 2 meters from a wall or if your sensor is giving you a wrong reading.

To understand Bayes' theorem, let's break this situation down into pieces:

First, let's define what we know: - You might be 2 meters from a wall (we'll call this your "position") - Your sensor gives you a measurement of 2 meters (we'll call this your "measurement")

Now, what Bayes' theorem helps us figure out is: Given that your sensor reads 2 meters, what's the probability that you're actually 2 meters from the wall?

Here's the magic formula (don't worry, we'll break it down):

```
P(position | measurement) = P(measurement | position) × P(position) / P(
    measurement)
```

Let's understand each piece:

1. `P(position | measurement)` is what we want to know: the probability of being at a position, given our sensor measurement. This is called the "posterior probability."

2. `P(measurement | position)` is how likely we are to get this measurement if we really are at that position. We know our sensor isn't perfect - maybe it's 90% accurate when we're actually at 2 meters. This is called the "likelihood."

3. `P(position)` is what we believed about our position before taking the measurement. Maybe based on our last estimate, we thought there was a 70% chance we were at 2 meters. This is called the "prior probability."

4. `P(measurement)` is how likely we are to get this measurement in general. Think of it as a normalizing factor that makes all our probabilities add up to 100%.

Let's put some numbers in: - If our sensor is 90% accurate: P(measurement | position) = 0.9 - If we thought we were probably at 2m: P(position) = 0.7 - Let's say P(measurement) = 0.8 (this is calculated considering all possibilities)

Then:

```
P(position | measurement) = 0.9 × 0.7 / 0.8 = 0.79
```

This tells us that after getting the measurement, we're 79% confident about our position - more confident than our prior belief of 70%!

The beautiful thing about Bayes' theorem is that it gives us a formal way to: 1. Start with what we believe (prior) 2. Consider new evidence (likelihood) 3. Update our belief (posterior)

In robotics, we use this process continuously. Every time we: - Move (this changes our prior belief) - Take a measurement (this gives us new evidence) - We use Bayes' theorem to update our belief about where we are

This is the foundation of probabilistic robotics and the basis for algorithms like Kalman filters and particle filters, which we'll explore later.

2.3 Conditional probability

Think of probability as measuring how likely something is to happen. Now, conditional probability takes this a step further by asking: "How likely is this event to happen, given that we already know something else has happened?"

The formal notation for conditional probability is $P(A|B)$, which reads as "the probability of A given B." Mathematically, it's expressed as:

$P(A|B) = P(A \cap B)/P(B)$

Let's break this down with a real-world example. Imagine we have a deck of 52 playing cards, and we want to know the probability of drawing a king, given that we've already drawn a red card.

To solve this:

1. First, we identify what we know: we've drawn a red card (this is our condition B)
2. We want to find the probability of having a king among these red cards (this is our event A)

3. $P(B)$ = probability of drawing a red card = $26/52 = 1/2$
4. $P(A \cap B)$ = probability of drawing a red king = $2/52 = 1/26$
5. Therefore, $P(A|B) = (2/52)/(26/52) = 2/26 = 1/13$

This shows us something interesting: while the probability of drawing a king from the full deck is 4/52 (about 0.077), the probability of drawing a king given that we know the card is red is 1/13 (about 0.077). In this case, knowing the card is red didn't change the probability of it being a king, because kings are evenly distributed between red and black cards.

This leads us to an important concept: independence. If knowing one event doesn't affect the probability of another event, we say these events are independent. In such cases, $P(A|B) = P(A)$. However, in many real-world scenarios, events are dependent, and conditional probability helps us account for this dependency.

Consider a medical example: the probability of having a certain disease might be 1% in the general population, but if we know a person has a specific symptom, the conditional probability of having the disease given this symptom might be much higher, say 30%. This is why doctors use symptoms to update their diagnostic probabilities.

2.4 Markov assumption

The Markov assumption, also known as the Markov property, is a fundamental concept in probability theory that helps us model complex sequences of events in a manageable way. Let me break this down step by step.

The core idea of the Markov assumption is that the future state of a system depends only on its present state, not on its past states. In probability terms, this means that if we want to predict what happens next, we only need to know what's happening right now, not the entire history of what happened before.

To understand this more concretely, imagine you're watching the weather. A pure Markov process would say that tomorrow's weather only depends on today's weather, not on what the weather was like last week or last month. While this might seem like an oversimplification (and in reality, weather patterns are more complex), this assumption often proves surprisingly useful in many real-world applications.

Let's express this mathematically. For a sequence of events $X_1, X_2, X_3, ..., X_n$ the Markov property states that:

$$P(X_{n+1}|X_n, X_{n-1}, ..., X_1) = P(X_{n+1}|X_n)$$

This equation tells us that the probability of the next state (X_{n+1}) given all previous states is equal to the probability of the next state given just the current state (X_n). This dramatically simplifies our calculations while still capturing many important patterns in real-world processes.

Think of it like playing a game of chess. While each position arose from a long sequence of moves, a player really only needs to look at the current board position to decide their next move. The specific sequence of moves that led to this position, while interesting historically, isn't directly relevant to choosing the next best move.

The Markov assumption is particularly powerful because it allows us to build practical models of complex systems. It's used in:

1. Natural Language Processing - In simple language models, the probability of the next word might depend only on the current word (or last few words), not the entire sentence history.
2. Financial Markets - Some basic models assume that tomorrow's stock price depends only on today's price, not the entire price history.
3. Biology - Gene sequences can be modeled using Markov chains, where each base pair depends only on the previous few pairs.
4. Machine Learning - Hidden Markov Models use this property to model sequential data in a computationally efficient way.

It's important to note that the Markov assumption comes in different "orders." What I've described is a first-order Markov process, where we only look at the immediate previous state. In a second-order Markov process, we look at the last two states, and so on. Higher-order Markov processes can capture more complex dependencies but require more computational resources.

This assumption, while powerful, isn't always perfectly accurate in real-world situations. Many processes have longer-term dependencies that a simple Markov model might miss. However, the simplification it provides often outweighs these limitations, making it an invaluable tool in probability theory and its applications.

Chapter 3

Bayesian Filtering Framework

- ☐ Bayesian Filtering Framework
  - ☐ Recursive state estimation
  - ☐ Prediction step (motion model)
  - ☐ Update step (measurement model)
  - ☐ Chapman-Kolmogorov equation
  - ☐ Bayes filter algorithm
  - ☐ Linear vs nonlinear systems

Chapter 4

Kalman Filtering

- ☐ Kalman Filtering
  - ☐ Linear Kalman Filter
    - ☐ System model and assumptions
    - ☐ Prediction equations
    - ☐ Update equations
    - ☐ Uncertainty propagation
  - ☐ Extended Kalman Filter (EKF)
    - ☐ Linearization process
    - ☐ Jacobian matrices
    - ☐ Algorithm implementation
  - ☐ Unscented Kalman Filter (UKF)
    - ☐ Sigma points
    - ☐ Unscented transform
    - ☐ Algorithm implementation

Chapter 5

Particle Filtering

- ☐ Particle Filtering
    - ☐ Monte Carlo methods
    - ☐ Importance sampling
    - ☐ Particle representation
    - ☐ Sequential Importance Sampling (SIS)
    - ☐ Resampling techniques
    - ☐ Sample degeneracy and impoverishment
    - ☐ Adaptive particle filtering

Chapter 6

Advanced Topics

- ☐ Advanced Topics
  - ☐ Multi-hypothesis tracking
  - ☐ SLAM basics
  - ☐ Sensor fusion techniques
  - ☐ Loop closure
  - ☐ Global vs local localization

Chapter 7

MATLAB Implementation: Kalman Filter

- ☐ MATLAB Implementation: Kalman Filter
  - ☐ Linear KF implementation
    - ☐ State prediction
    - ☐ Measurement update
    - ☐ Covariance propagation
  - ☐ EKF implementation
    - ☐ System modeling
    - ☐ Jacobian computation
    - ☐ Filter implementation
  - ☐ Visualization and analysis
  - ☐ Performance evaluation

Chapter 8

MATLAB Implementation: Particle Filter

- ☐ MATLAB Implementation: Particle Filter
  - ☐ Basic PF framework
  - ☐ Particle initialization
  - ☐ Motion model implementation
  - ☐ Measurement model
  - ☐ Weight computation
  - ☐ Resampling implementation
  - ☐ Visualization tools
  - ☐ Performance metrics

Chapter 9

Practical Applications

- ☐ Practical Applications
    - ☐ Vehicle localization case studies
    - ☐ Robot navigation examples
    - ☐ Integration with mapping
    - ☐ Real-world challenges
    - ☐ Best practices and optimization

Chapter 10

Project Work

- ☐ Project Work
  - ☐ Implementation exercises
  - ☐ Real dataset analysis
  - ☐ Performance comparison of different filters
  - ☐ Parameter tuning
  - ☐ Documentation and presentation