SES/RAS 598: Space Robotics and AI

Lecture 2: State Estimation Techniques

Dr. Jnaneshwar Das

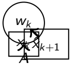
Arizona State University
School of Earth and Space Exploration

Spring 2025

Lecture Outline

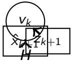
Kalman Filter Overview

Kalman Filter Steps

Prediction Step 

$\hat{x}_{k+1} = Ax_k + Bu_k$

$P_{k+1} = AP_kA^T + Q$

Update Step 

$K_{k+1} = P_{k+1}H^T(HP_{k+1}H^T + R)^{-1}$

$=$ (I - K$_{k+1}H)P_{k+1}$

Implementation Example: Kalman Filter

```python
import numpy as np

class KalmanFilter:
    def __init__(self, A, B, C, Q, R):
        self.A = A  # State transition matrix
        self.B = B  # Input matrix
        self.C = C  # Measurement matrix
        self.Q = Q  # Process noise covariance
        self.R = R  # Measurement noise covariance

    def predict(self, x, P, u=None):
        """Predict next state and covariance."""
        if u is not None:
            x_pred = self.A @ x + self.B @ u
        else:
            x_pred = self.A @ x
        P_pred = self.A @ P @ self.A.T + self.Q
        return x_pred, P_pred

    def update(self, x_pred, P_pred, y):
        """Update state estimate using measurement."""
        K = P_pred @ self.C.T @ np.linalg.inv(
            self.C @ P_pred @ self.C.T + self.R)
        x = x_pred + K @ (y - self.C @ x_pred)
        P = (np.eye(len(x)) - K @ self.C) @ P_pred
        return x, P
```

Introduction to Particle Filters

- **Key Features:**
  - Non-parametric estimation
  - Handles non-linear systems
  - Represents arbitrary distributions

Introduction to Particle Filters

- **Key Features:**
  - Non-parametric estimation
  - Handles non-linear systems
  - Represents arbitrary distributions
- **Working Principle:**
  - Particle representation
  - Sequential importance sampling
  - Resampling step

Introduction to Particle Filters

- **Key Features:**
  - Non-parametric estimation
  - Handles non-linear systems
  - Represents arbitrary distributions
- **Working Principle:**
  - Particle representation
  - Sequential importance sampling
  - Resampling step
- **Advantages:**
  - No linearity assumption
  - Handles multi-modal distributions
  - Robust to outliers

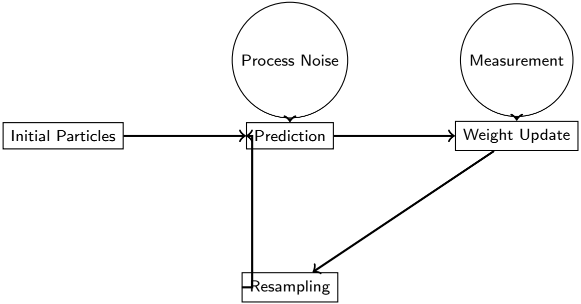
Particle Filter Algorithm

```
for each time step k do
    for each particle i do
        Sample new state: x_k^i ~ p(x_k | x_{k-1}^i, u_k)
        Update weight: w_k^i = w_{k-1}^i p(z_k | x_k^i)
    end for
    Normalize weights: w_k^i = w_k^i / Σ_j w_k^j
    if N_eff < N_threshold then
        Resample particles
    end if
end for
```

Implementation Example: Particle Filter

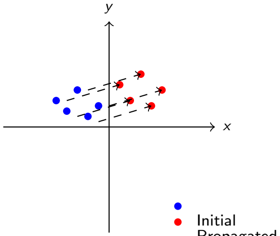
```python
import numpy as np
from scipy.stats import multivariate_normal

class ParticleFilter:
    def __init__(self, n_particles, motion_model, measurement_model):
        self.n_particles = n_particles
        self.motion_model = motion_model
        self.measurement_model = measurement_model
        self.particles = None
        self.weights = None

    def initialize(self, initial_state, initial_cov):
        """Initialize particles from Gaussian distribution."""
        self.particles = multivariate_normal.rvs(
            mean=initial_state,
            cov=initial_cov,
            size=self.n_particles
        )
        self.weights = np.ones(self.n_particles) / self.n_particles

    def predict(self, u=None):
        """Propagate particles through motion model."""
        for i in range(self.n_particles):
            self.particles[i] = self.motion_model(
                self.particles[i], u)

    def update(self, measurement):
        """Update particle weights using measurement."""
        for i in range(self.n_particles):
            likelihood = self.measurement_model(
```

Particle Filter Overview

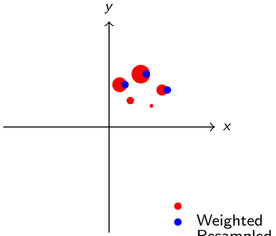


Particle Filter Steps

Particle Propagation:



Resampling:



Assignment Overview

- **Objectives:**
  - Implement 2D state estimation using ROS2
  - Compare Kalman and particle filter performance
  - Visualize results using RViz

Assignment Overview

- **Objectives:**
  - Implement 2D state estimation using ROS2
  - Compare Kalman and particle filter performance
  - Visualize results using RViz
- **Key Components:**
  - ROS2 node implementation
  - Filter implementation
  - Visualization and analysis

Assignment Overview

- **Objectives:**
  - Implement 2D state estimation using ROS2
  - Compare Kalman and particle filter performance
  - Visualize results using RViz
- **Key Components:**
  - ROS2 node implementation
  - Filter implementation
  - Visualization and analysis
- **Evaluation:**
  - Code quality and documentation
  - Filter performance metrics
  - Analysis and discussion

ROS2 Environment Setup

```bash
# Create ROS2 workspace
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src

# Create package
ros2 pkg create --build-type ament_python \
    state_estimation_assignment

# Build workspace
cd ~/ros2_ws
colcon build

# Source workspace
source install/setup.bash
```

Basic ROS2 Node Structure

```python
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import PoseStamped
from nav_msgs.msg import Odometry

class StateEstimator(Node):
    def __init__(self):
        super().__init__('state_estimator')
        self.subscription = self.create_subscription(
            Odometry,
            'odom',
            self.odom_callback,
            10)
        self.publisher = self.create_publisher(
            PoseStamped,
            'estimated_pose',
            10)

    def odom_callback(self, msg):
        # Implement state estimation here
        pass
```

Preparation for Next Week

- **Assignment 1:**
  - Review ROS2 basics
  - Study filter implementations
  - Start coding basic structure

Preparation for Next Week

- **Assignment 1:**
  - Review ROS2 basics
  - Study filter implementations
  - Start coding basic structure
- **Reading:**
  - Extended Kalman Filter
  - Unscented Kalman Filter
  - Advanced particle filter topics

Preparation for Next Week

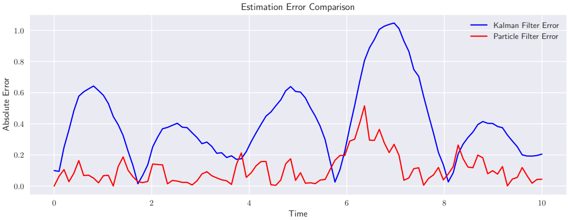
- **Assignment 1:**
  - Review ROS2 basics
  - Study filter implementations
  - Start coding basic structure
- **Reading:**
  - Extended Kalman Filter
  - Unscented Kalman Filter
  - Advanced particle filter topics
- **Tools:**
  - Test ROS2 installation
  - Practice with RViz
  - Explore sensor fusion tutorial

Questions?

Thank you!

Contact: jdas5@asu.edu

Filter Performance Comparison



- Kalman Filter: Optimal for linear systems with Gaussian noise
- Particle Filter: Better handles non-linear dynamics and non-Gaussian noise
- Trade-off between computational cost and estimation accuracy