EX6 Mobile Robotics

Abdulrahman Hameshli

December 2024

6.1 Simplified Global Localization in Grid World

Why the Sequence Works:

The proposed action sequence – **Right (R) until hit, Up (U) until hit, Left (L) until hit, Down (D) until hit** – systematically explores the boundaries of the grid. Here is why it works:

- **Right (R)** ensures the robot moves to the rightmost boundary of its row.
- **Up (U)** moves the robot to the topmost boundary of its column.
- **Left (L)** forces the robot to the leftmost boundary of its row.
- **Down (D)** moves the robot to the bottommost boundary of its column.

After executing these movements, the robot ends up at the **bottom-left corner** of the grid. Since the grid boundaries are deterministic and the robot always moves until hitting an obstacle, the sequence localizes the robot regardless of its starting position.

6.2 Markov Localization: Code Explanation

Overview of Markov Localization

Markov Localization is a probabilistic method used to localize a robot in an environment. The environment is represented as a grid map where each cell corresponds to a possible position of the robot. The method uses two models:

- **Transition Model:** Describes how the robot's position changes when it takes an action.
- **Observation Model:** Updates the belief about the robot's position based on sensor observations.

The belief about the robot's position is represented as a probability distribution over all cells in the grid.

—

Belief Prediction (Transition Model)

The transition model calculates the probability of moving to a specific state based on the robot's previous state and control input. The prediction step uses offsets u and v to account for all possible transitions into a given cell (i, j).

Key Variables:

- u and v: Offsets that represent the relative movement from a previous state $(\text{prev\_i}, \text{prev\_j})$ to the current state (i, j).
- p_{desired}: Probability of moving to the desired cell.
- p_{stay}: Probability of staying in the current cell.
- p_{diagonal}: Probability of moving diagonally.

Offsets and Directions: The values of u and v range from -1 to 1, creating a 3×3 neighborhood:

$$u, v \in \{-1, 0, 1\}.$$

The offsets correspond to the following movements:

u	v	Movement
-1	0	Move Up (North)
1	0	Move Down (South)
0	-1	Move Left (West)
0	1	Move Right (East)
-1	-1	Move Diagonally Up-Left
-1	1	Move Diagonally Up-Right
1	-1	Move Diagonally Down-Left
1	1	Move Diagonally Down-Right
0	0	Stay in Place

Belief Prediction Formula: For each cell (i, j), the belief prediction is computed as:

$$\text{beliefPrediction}(i, j) = \sum_{u=-1}^{1} \sum_{v=-1}^{1} P(i, j \mid \text{prev\_i}, \text{prev\_j}, \text{control}) \cdot \text{belief}(\text{prev\_i}, \text{prev\_j}),$$

where:

- $P(i, j \mid \text{prev\_i}, \text{prev\_j}, \text{control})$ is the transition probability.
- $\text{belief}(\text{prev\_i}, \text{prev\_j})$ is the belief at the previous state.

The transition probability depends on the control input:

$$P(i, j \mid \text{prev\_i}, \text{prev\_j}, \text{control}) = \begin{cases} p_{\text{desired}} & \text{if the move corresponds to the control input,} \\ p_{\text{stay}} & \text{if staying in place,} \\ p_{\text{diagonal}} & \text{if moving diagonally,} \\ 0 & \text{otherwise.} \end{cases}$$

—

Belief Update (Observation Model)

The observation model uses sensor data to refine the belief. The robot has bumper sensors in four directions: North, South, West, and East. The observations are represented as:

$$\text{observations} = [\text{North, South, West, East}].$$

Each observation can be either:

- 1: Sensor detects an obstacle.
- 0: Sensor does not detect an obstacle.

Observation Probability: The observation probability for a given state (i, j) is computed as:

$$P(\text{observations} \mid \text{state}) = \prod_{k \in \{\text{N, S, W, E}\}} P_k,$$

where:

$$P_k = \begin{cases} p_{\text{measure}} & \text{if the sensor detects an obstacle correctly,} \\ 1 - p_{\text{measure}} & \text{if the sensor misses an obstacle.} \end{cases}$$

Here, p_{measure} is the probability of a correct sensor measurement.

—

Normalization

After combining the belief prediction and the observation probabilities, the belief is normalized to ensure it sums to 1:

$$\text{map.belief}(i, j) = \frac{\text{updatedBelief}(i, j)}{\sum_{i,j} \text{updatedBelief}(i, j)}.$$

—

Full Process Summary

1. **User Input:** - The control input (`W`, `A`, `S`, `D`) sets the direction (`N`, `S`, `W`, `E`).
2. **Prediction Step:** - Use the transition model to spread the belief to neighboring cells.
3. **Update Step:** - Refine the belief based on the sensor observations using the observation model.
4. **Normalization:** - Normalize the belief so the sum of probabilities equals 1.

—

How the Control Input is Used

The control input determines the desired movement direction:

- `N` (North): Move up $\rightarrow u = -1, v = 0$.
- `S` (South): Move down $\rightarrow u = 1, v = 0$.
- `W` (West): Move left $\rightarrow u = 0, v = -1$.
- `E` (East): Move right $\rightarrow u = 0, v = 1$.

In the transition model, the control input determines which moves have the highest probability (p_{desired}), as shown in:

$$P(i, j \mid \text{prev\_i}, \text{prev\_j}, \text{control}).$$

06.4-06.5 EKF Implementation

The task involves implementing the observation prediction and data association step for an Extended Kalman Filter (EKF) Localization problem. The specific objectives are:

- Transform global beacon positions into the robot's frame using the robot's state.
- Compute the innovation, measurement Jacobian, and measurement uncertainty for each observed beacon.
- Perform data association between observed and predicted beacons using the Mahalanobis distance.
- Ensure statistical compatibility using a chi-square threshold ($\alpha = 0.95$).
- Stack results for EKF correction.

Solution Overview

1. Measurement Function

The measurement function transforms a global beacon position $\mathbf{p}_{\text{beacon}} = [x_b; y_b]$ into the robot's local frame using the robot's state $\mathbf{state} = [x_r; y_r; \theta_r]$:

$$\Delta = \mathbf{p}_{\text{beacon}} - \begin{bmatrix} x_r \\ y_r \end{bmatrix},$$

$$\mathbf{z}_{\text{pred}} = \begin{bmatrix} \cos(\theta_r) & \sin(\theta_r) \\ -\sin(\theta_r) & \cos(\theta_r) \end{bmatrix} \cdot \Delta.$$

The Jacobian of the measurement function is:

$$\mathbf{H} = \begin{bmatrix} -\cos(\theta_r) & -\sin(\theta_r) & -\sin(\theta_r)\Delta_x + \cos(\theta_r)\Delta_y \\ \sin(\theta_r) & -\cos(\theta_r) & -\cos(\theta_r)\Delta_x - \sin(\theta_r)\Delta_y \end{bmatrix}.$$

2. Data Association

Data association matches observed beacons to predicted beacons:

- The Mahalanobis distance is used:

  $$d^2 = \mathbf{y}^\top \mathbf{S}^{-1} \mathbf{y}, \quad \mathbf{S} = \mathbf{H}\mathbf{C}\mathbf{H}^\top + \mathbf{R},$$

  where $\mathbf{y} = \mathbf{z}_{\text{obs}} - \mathbf{z}_{\text{pred}}$ is the innovation.
- Beacons are matched if $d^2 \leq \chi^2_{0.95,2}$, ensuring statistical compatibility.

3. Implementation Steps

1. Extract beacon positions from the global map using column filtering.
2. For each observed beacon:
   - Predict its position in the robot's frame.
   - Compute the innovation and Mahalanobis distance.
   - Find the best match among global beacons.
   - Verify the match using a chi-square test.
3. Stack the innovation, Jacobians, and measurement uncertainty matrices for EKF correction.

Final Implementation

The key functions include:

- `measurementfunction`: Implements the transformation and Jacobian calculation.
- `predictandmatch`: Implements data association, innovation stacking, and compatibility checks.

Results and Conclusion

The implemented functions correctly predict beacon positions, compute innovations, and associate observations with map data. Statistical compatibility is ensured using the Mahalanobis distance and chi-square threshold. The final outputs include:

- Stacked innovation vector.
- Stacked measurement Jacobian.
- Stacked measurement uncertainty.

The solution integrates seamlessly into the EKF framework, enabling accurate state estimation for mobile robot localization.