Safe High Speed Navigation in Dynamic Environment using a Bayesian Framework

Gunjan Dabas (2021253), Soham Chitlangia (2021291)

Bayesian Machine Learning Final Project

Abstract

Imagine navigating a fast-paced, unpredictable environment where obstacles appear and move unexpectedly. Whether it is autonomous vehicles, drones or robots in a crowded space, the ability to avoid collisions and find the best path is essential but challenging. Traditional methods struggle in these dynamic settings, prompting the need for smarter and more adaptable solutions.Our work tackles this challenge by focusing on robust collision detection and adaptive path planning. Existing approaches like Receding Horizon Planners (RHP) or POMDPs rely on heavy computation and precise models, making them unreliable in scenarios with erratic obstacles. RHP's short-term focus also limits flexibility for long-term navigation. To overcome these issues, we integrate Rao Blackwellised Particle Filter SLAM for real-time mapping with A* for efficient, adaptable planning. Additionally, we address gaps in obstacle tracking. While most systems detect dynamic objects, they rarely predict behavior effectively. Using Bayesian methods with a Beta-Bernoulli prior, we achieve more accurate movement prediction, enhancing collision detection.Finally, we explore neural networks to further improve predictive accuracy. By combining computational efficiency and advanced techniques, our approach offers a promising solution to navigate complex, fast-changing environments.

Contents

1 Intoduction

Navigating through dynamic environments poses a significant challenge for autonomous systems such as drones, robots, and self-driving vehicles. These systems must efficiently traverse unpredictable spaces where obstacles may appear and move unexpectedly, all while ensuring high-speed, collision-free movement and optimal path planning. Traditional approaches, such as Receding Horizon Planners (RHP) or Partially Observable Markov Decision Processes (POMDPs), often require vehicles to slow down significantly to compute safe paths, limiting their effectiveness in scenarios demanding rapid motion.

This project addresses these limitations by proposing a robust navigation framework explicitly designed for high-speed motion. Unlike conventional methods, the proposed system ensures that the robot maintains its velocity while safely navigating dynamic environments. By integrating dynamic environment mapping, collision prediction, and adaptive path planning using a Bayesian framework, the system minimizes the need for deceleration, enabling swift and efficient movement even in complex settings.

The proposed approach progressively builds on three levels of sophistication:

1. **Collision detection** for static and dynamic obstacles using predictive probability models.
2. **Path planning** for optimal navigation using algorithms such as Dijkstra's and A*.
3. **Simple machine learning** techniques to enhance obstacle behavior prediction and collision avoidance.

By incorporating probabilistic reasoning and dynamic mapping, this framework offers a scalable and efficient solution for navigating uncertain environments without compromising on speed.

---

2 Problem Statement

Dynamic environments pose unique challenges for navigation systems. Obstacles in such environments are not only numerous but may also exhibit unpredictable movements. The problem is exacerbated when the system lacks complete knowledge of the environment. Specifically, the key challenges are:

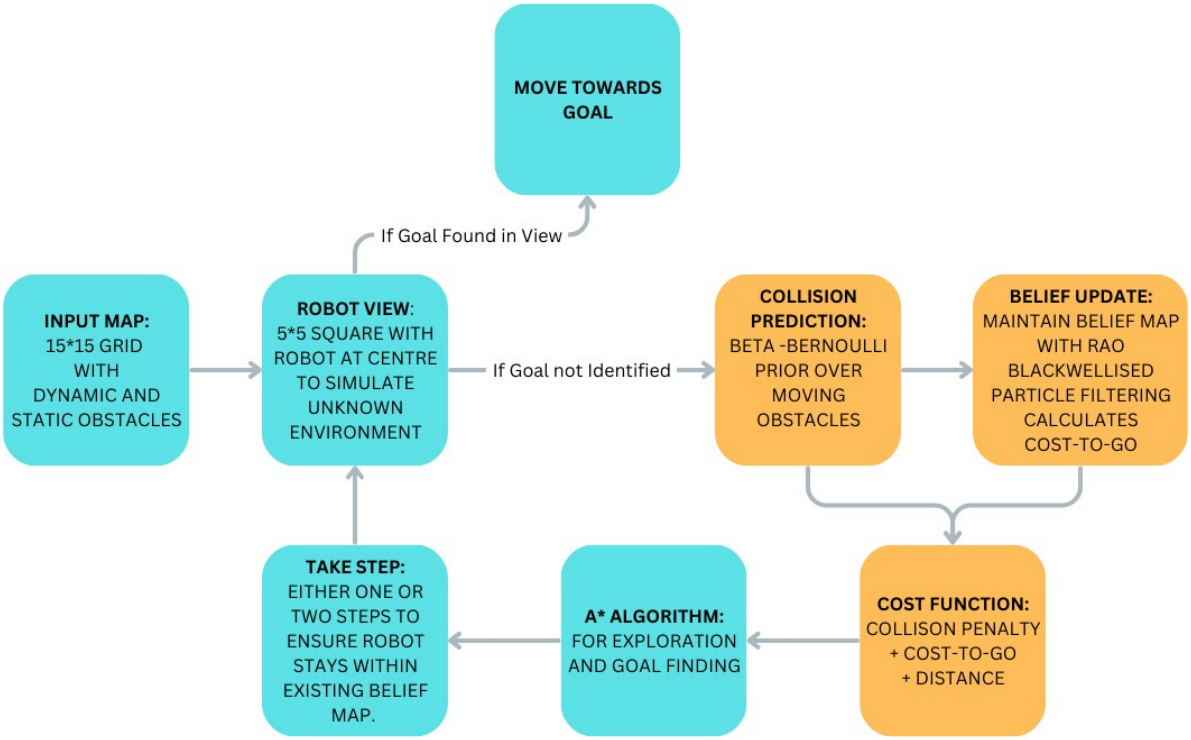
- **Collision Detection:** Identifying and predicting potential collisions with static and dynamic obstacles in real time.
- **Path Planning:** Determining the shortest and safest path to a goal while avoiding obstacles, even in partially observable or unknown environments.
- **Predictive Modeling:** Leveraging machine learning to anticipate obstacle behavior and refine navigation strategies.

This project addresses these challenges by developing a Bayesian navigation framework that integrates mapping, planning, and prediction, ensuring computational efficiency and adaptability.

---

3 Algorithm

Our main task is to reach the goal in an unknown environment with dynamic obstacles.We designed the following workflow and explain each of the yellow marked sections in detail below



3.1 Collision Detection

This algorithm computes a 5x5 collision probability matrix centered around a robot's current position, taking into account the environment's grid layout, the robot's belief map, and beta priors for obstacles. It begins by initializing a zero matrix for probabilities and defining the visible range as 2, which limits calculations to a 5x5 grid around the robot. For each cell within this range, the algorithm computes the global coordinates relative to the robot's position and checks if the cell exists in the belief map. If present, the algorithm retrieves Beta distribution parameters α, β from the priors and evaluates the cell's contents. For moving obstacles, the probability of collision is calculated using the Beta distribution formula $\frac{\alpha}{\alpha+\beta}$, and this probability is assigned to the corresponding cell. Additionally, a fraction of this probability is distributed to adjacent cells to model potential spread. Static obstacles are assigned a fixed probability of 1.0, reflecting certain collisions, while cells not present in the belief map are given a default probability of 0.2 to account for uncertainty. The algorithm outputs the updated matrix, providing a probabilistic representation of collision likelihoods for the robot's local area, which aids in navigation and decision-making in dynamic environments.

3.2 Path Planning

Algorithm 1 Calculate Collision Probabilities

Input: Robot position $(robot\_x, robot\_y)$, Grid size $size$, Walls $walls$, Belief Map $belief\_map$, Priors $beta\_priors(size, size, 2)$
Output: Collision probability matrix $probabilities$ of size 5×5
Initialize $probabilities$ as a 5×5 zero matrix
Set $visible\_range = 2$
for $dx \leftarrow -visible\_range$ **to** $visible\_range$ **do**
  for $dy \leftarrow -visible\_range$ **to** $visible\_range$ **do**
    $x \leftarrow robot\_x + dx$
    $y \leftarrow robot\_y + dy$
    if $(x, y) \in belief\_map$ **then**
      Retrieve $\alpha, \beta \leftarrow beta\_priors[(x, y)]$
      if $Grid[x, y] == moving\_obstacle$ **then**
        $\alpha \leftarrow successes + 1$,
        $base\_prob \leftarrow \frac{\alpha}{\alpha+\beta}$
        $probabilities[dx + visible\_range, dy + visible\_range] \leftarrow base\_prob$
        for $(ndx, ndy) \in \{(-1, 0), (1, 0), (0, -1), (0, 1)\}$ **do**
          $nx \leftarrow x + ndx$, $ny \leftarrow y + ndy$
          $prob\_x \leftarrow nx - robot\_x + visible\_range$
          $prob\_y \leftarrow ny - robot\_y + visible\_range$
          $probabilities[prob\_x, prob\_y]$ += $base\_prob \times 0.25$
        end for
      end if
      if $Grid[x, y] == static\_obstacle$ **then**
        $probabilities[x, y] \leftarrow 1.0$
      end if
    else
      $probabilities[dx + visible\_range, dy + visible\_range] \leftarrow 0.2$
    end if
  end for
end for
Return $probabilities$

Algorithm 2 Update Belief Map Using Rao-Blackwellized Particle Filter (RBPF)

Initialize an empty list *new_particles* and a zero matrix *belief_accumulator* of the same size as the belief map
for each particle $(particle, belief\_map, weight)$ in *particles* **do**
  Copy the particle's *belief_map* into *new_belief_map*
  Extract the particle's position $(robot\_x, robot\_y)$
  Define $visible\_range = 2$
  for each (dx, dy) in the 5×5 grid centered around $(robot\_x, robot\_y)$ **do**
    Compute global coordinates $(x, y) = (robot\_x + dx, robot\_y + dy)$
    if (x, y) is within bounds of the grid **then**
      if (x, y) is a wall **then**
        Set $new\_belief\_map[x, y] = 1$
      else if (x, y) is a moving obstacle **then**
        Set $new\_belief\_map[x, y] = 2$
      else
        Set $new\_belief\_map[x, y] = 0$
      end if
    end if
  end for
  Update *weight* using the difference between *new_belief_map* and the observation
  Generate possible moves for the particle's position
  Select *next_particle* randomly from possible moves
  Add $(next\_particle, new\_belief\_map, weight)$ to *new_particles*
  Accumulate *belief_accumulator* with $weight \times new\_belief\_map$
end for
Normalize particle weights to sum to 1
Compute updated *belief_map* by normalizing *belief_accumulator*
Resample particles based on their weights
Return updated *belief_map* and resampled *particles*

The belief map is updated using a Rao-Blackwellized Particle Filter (RBPF), which combines particle filtering for the robot's position with a grid-based representation for the environment's belief. Each particle represents a hypothesis about the robot's position and its corresponding belief map, along with an associated weight indicating the particle's importance. The update process begins by iterating through all particles and creating a new belief map for each based on the particle's position. For each cell within a defined visible range (a 5x5 grid centered on the particle's position), the algorithm checks if the cell corresponds to a wall, moving obstacle, or empty space and updates the belief map accordingly.

The particle's weight is then adjusted based on how well its belief map matches the current observation, measured using an exponential function that penalizes differences between the predicted and observed maps. Next, possible moves for the particle's position are generated, and one is randomly selected to simulate the robot's motion. The new particle, its updated belief map, and its recalculated weight are added to a new list. During this process, a belief accumulator matrix aggregates the weighted belief maps of all particles.

After processing all particles, the weights are normalized so that their total equals one. The accumulated belief map is also normalized and clipped to ensure values fall within valid bounds, producing the updated belief map. To refine the particle set, resampling is performed, where particles are probabilistically selected based on their normalized weights, ensuring that more likely hypotheses are retained. This updated belief map provides a probabilistic representation of the environment, helping the robot understand the likelihood of obstacles or free space at each grid cell. The RBPF thus enables robust, probabilistic state estimation in dynamic environments.

3.3 A* Algorithm for Next Step

In this step we determine the robot's next step using a combination of belief map updates, path planning with A* search, and collision prediction. The choose_next_state method operates in three main phases:

Belief and Collision Updates:

The belief map, which represents the robot's understanding of the environment, is updated by refining the likelihood of obstacles using particles (update_particles) and resampling them to retain diversity (resample_particles). Collision probabilities are predicted using the updated belief map to estimate the risk of navigating through specific cells in the grid. Path Planning with A*:

The a_star_search method computes an optimal path from the robot's current position to a goal position using a heuristic-based approach. The algorithm begins by initializing an open set, where potential nodes are prioritized by cost. For each node, it evaluates its neighbors, considering movement costs, belief map costs, and penalties for potential collisions. The neighbor with the lowest total cost is added to the path. The cost calculation incorporates: Move Cost: A cost of 1 for adjacent moves and 1.414 for diagonal moves. Belief Cost: The probability of encountering obstacles as derived from the belief map. Collision Penalty: A penalty based on predicted collision probabilities, weighted heavily to avoid risky areas. The method tracks the best path to each node using the came_from dictionary and reconstructs the optimal path once the goal is reached. Path Execution:

After planning, the robot moves along the calculated path. If the path length permits, it moves

Algorithm 3 Decide Next Step for Robot Navigation

Step 1: Update Belief and Collision Models
Call `update_particles()` to refine the belief map
Call `resample_particles()` to maintain particle diversity
Predict collision probabilities using `predict_collision_probabilities()`
Step 2: Plan Path Using A* Search
Initialize *open_set* with the robot's position and $cost\_so\_far$ as zero
Maintain $came\_from$ to record the path and $cost\_so\_far$ to track costs
while *open_set* is not empty **do**
  Pop the node with the lowest cost from *open_set*
  if current position equals the goal **then**
    Break
  end if
  Generate *neighbors* of the current position within grid bounds
  for each *neighbor* in *neighbors* **do**
    if *neighbor* is a wall or has high obstacle probability **then**
      Continue
    end if
    Compute *move_cost* based on neighbor proximity (diagonal or adjacent)
    Compute $belief\_cost$ from the belief map
    Compute *collision_penalty* using predicted collision probabilities
    Compute *new_cost* as the sum of all costs
    if *neighbor* is not in $cost\_so\_far$ or *new_cost* is lower **then**
      Update $cost\_so\_far[neighbor]$ and $came\_from[neighbor]$
      Push *neighbor* into *open_set* with $priority = new\_cost$
    end if
  end for
end while
Reconstruct the path from $came\_from$
Return the planned path
Step 3: Execute Next Step
if path is non-empty **then**
  Move to the next position on the path (up to 2 steps)
  if robot collides with a wall or moving obstacle **then**
    Increment alpha in prior map and report the collision
  else
    Increment beta in prior map
  end if
end if

up to two steps. This allows the robot to adapt quickly to new observations and reduce unnecessary re-computations. If the robot collides with a wall or moving obstacle at its new position, it increments a collision counter and logs the event. This approach combines probabilistic modeling (via the belief map) and deterministic planning (via A* search), making it robust for dynamic and uncertain environments. The robot navigates efficiently while minimizing collision risks, continuously updating its understanding of the environment.

3.4 What's Bayesian

Beta-Bernoulli Prior over Collision Detection

In the context of collision detection and obstacle modeling, we use a Beta-Bernoulli prior to model the probability distribution of potential obstacles. This prior is based on a Beta distribution, which is particularly useful when the underlying problem involves binary events (e.g., the presence or absence of obstacles). The Beta distribution is parameterized by two hyperparameters, typically denoted as α, β which represent the number of collisions and non collisions, respectively, in prior observations of obstacle presence.

The Beta distribution is a continuous probability distribution that is often used to model proportions or probabilities. In our case, it models the probability that a given cell in the belief map is occupied by an obstacle.The Bernoulli distribution is a discrete distribution that models binary events (like whether a cell is free or blocked). When combined with the Beta distribution as a prior, it forms a Beta-Bernoulli model, which is used to update beliefs about each grid cell's occupancy.

In the robot's grid, each grid cell might be either free or blocked by an obstacle. The Beta-Bernoulli model updates the probability of a cell being an obstacle (colliding with the robot) based on observed data over time. For each cell, the model incorporates the success/failure of previous observations (whether an obstacle was detected or not) to refine the estimate of its collision probability. This approach allows for a probabilistic treatment of collision detection, where the belief about whether a cell is an obstacle is updated as new evidence is obtained. The Beta-Bernoulli prior helps to continuously refine this probability, making the system adaptive to dynamic environments.

Belief map as prior for the next step

A belief map is a grid representation of the environment where each cell's value reflects the robot's belief about the presence of obstacles or free space. This belief is updated over time using sensor data and predictive models.

The belief map serves as a prior in decision-making. In the context of path planning and robot navigation, the belief map represents the robot's current understanding of its surroundings, including the locations of obstacles and free spaces. This prior information is essential for making informed decisions about where the robot should move next.

When deciding the next step, the robot uses the belief map to estimate which areas are safe or risky to navigate. If a region is likely to contain an obstacle (as inferred from the belief map), the robot avoids it. Conversely, the robot prioritizes areas with low obstacle probability. The belief map helps the robot avoid collisions by incorporating information from its prior experiences and sensor observations.

As the robot moves and senses new information, it updates its belief map, refining its understanding of the environment and improving the accuracy of its next step decisions. This dynamic update process is a key feature of probabilistic robotics, where decisions are made based on the robot's evolving belief about the world.

Bayesian Inference with Rao Blackwellised Particle Filtering

Bayesian Inference is a statistical method used to update the probability estimate for a hypothesis based on new evidence or data. In robotics, this is used to update the robot's belief about its position, obstacles, and other environmental factors based on sensor measurements. Bayesian inference helps to merge prior knowledge (e.g., belief map, past particle locations) with new sensory data to make decisions that reflect both uncertainty and new observations.

Particle filters, also known as Monte Carlo Localization (MCL), are commonly used in robotics for state estimation in environments with uncertainty. A particle filter approximates the posterior distribution of a robot's state by maintaining a set of particles (samples) representing possible states of the robot. Each particle has an associated weight that reflects how well it matches the observed data. The filter updates the particles over time, resampling them as needed to reflect the latest belief about the robot's position and environment.

The Rao-Blackwellisation technique improves particle filtering by combining particle filtering with an analytical solution for some parts of the system. In RBPF, certain aspects of the state space are marginalized analytically using a closed-form solution (e.g., a Kalman filter), while the rest of the state is estimated using particles. This combination allows for more efficient sampling and improved accuracy. In robotics, RBPF is especially useful when certain parts of the robot's state (such as position) can be modeled with an exact solution, while others (like sensor readings or environmental models) require approximation via particles.

In the context of robot navigation, RBPF is used to estimate both the robot's position and the belief about obstacles in the environment. As the robot moves, the belief map is updated by combining the particle filter's estimate of the robot's position with the collision probability model. The use of Bayesian inference ensures that the robot's belief about its environment is constantly refined based on new data from its sensors, leading to more informed decision-making. By using RBPF, the robot can more accurately estimate its position and the environment. The Rao-Blackwellisation process helps reduce the number of particles needed and increases the efficiency of the filter, leading to faster and more reliable updates, especially in large or complex environments.

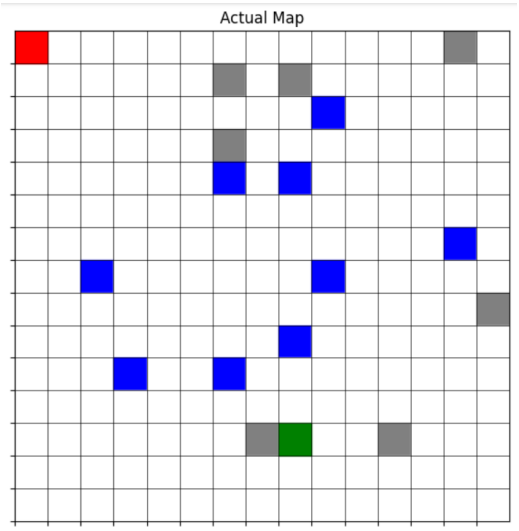
This combination of Bayesian inference and RBPF allows the robot to handle the uncertainty in both its location and the environment, making it robust to errors in sensor readings, movement, and the dynamic nature of the world around it. It is especially valuable in environments where both robot localization and obstacle detection need to be performed simultaneously and efficiently.

4 Results

4.1 Test Environment Description

We simulates a 15*15 grid with static obstacles and dynamic obtsacles. The Robot always starts at the upper leftmost corner and the goal is randomly set such that a path to it always

exists. An example is given here; the red denotes the robot, blue denotes moving obstacles, gray denotes static obstacles or walls, and green denotes the goal state.



4.2 Comparison with Baseline

We implemented a simple algorithm that performs exploration and reaches the goal with a very conservative approach, i.e. if it views an obstacle, it does not move two steps forward, but rather only one, for comparison purposes. Our Model Outperforms this model in terms of speed significantly. Although there is a slightly higher risk of collision in our model, since it is speeding in an unknown environment, we have added a time penalty of 3 iterations to it, to simulate reconfiguration in real-life scenario.

Base Line Model Average No. of Iterations to reach Goal: 51.689

Current Model Average No. of iterations to reach Goal: 16.178

Average No. of Collisions in Current Model : 0.016

This is averaged over 500 simulations and 10 dynamic obstacles.

When we increase the dynamic obstacle to 20:

Base Line Model Average No. of Iterations to reach Goal: 65.689

Current Model Average No. of iterations to reach Goal: 16.822

Average No. of Collisions in Current Model : 0.017

Thus while the baseline model takes a larger No. ofIterations, The Current Model's no. of iterations does not change drastically and neither does it's collisons

5 Conclusions

The project presents a Bayesian framework for safe high-speed navigation in dynamic environments, effectively addressing the limitations of traditional methods. By integrating collision detection, adaptive path planning using A*, and predictive modeling through Bayesian inference, the proposed solution demonstrates robust performance.

- **Efficiency:** The approach significantly reduces the average iterations required to reach the goal (16.18 compared to 51.69 for the baseline model), showcasing superior computational efficiency.
- **Adaptability:** The system adapts to dynamic and uncertain environments using a belief map and Bayesian updates, ensuring reliable decision-making.
- **Scalability:** The combination of Rao-Blackwellized Particle Filtering and Beta-Bernoulli models provides a scalable solution for real-time obstacle detection and prediction.

While the model involves a slightly higher risk of collision due to its high-speed focus, the inclusion of a reconfiguration penalty ensures practicality for real-life applications. Overall, the project successfully demonstrates a balance between safety and performance, marking a step forward in autonomous navigation systems.

References

1. Richter, C., Vega-Brown, W., Roy, N. (2018). Bayesian learning for safe high-speed navigation in unknown environments. Robotics Research: Volume 2, 325-341.
2. Lidoris, G., Wollherr, D., Buss, M. (2008). Bayesian Framework for State Estimation and Robot Behaviour Selection in Dynamic Environments. In Greedy Algorithms. IntechOpen.
3. Grisetti, G.; Stachniss, C.; Burgard, W. (2005). Improving Grid-based SLAM with Rao Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling International Conference of Robotics and Automation (ICRA), Barcelona, Spain.
4. Miller, I.; Campbell, M. (2007). Rao-Blackwellized Particle Filtering for Mapping Dynamic Environments. IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy.