WRITE SOMETHING CLEVER

951926 1001231 1024072 1028907

Abstract—**In this report, we present a system which performs the 1996 AAAI Mobile Robotics Competition task "Call a meeting"[1]. The robot was tasked with bringing a certain number of people to a meeting room previously determined as available. To complete the task, we use probabilistic road mapping, Monte Carlo localisation and a face detection module using the ROS framework, running on a Pioneer 2DX platform modified with a tripod-mounted Kinect. In the first section, we provide some background information about the areas of robotics that are relevant to the problem. We then provide a detailed description of our system and evaluate its performance. Finally, we discuss the experimental results and suggest areas for improvement.**

I. Background

Solutions to complex tasks often require the use of multiple techniques to solve different sub-problems, and this task is no exception. We were required to implement a localisation algorithm to determine the position of the robot, a probabilistic road map to allow paths to be planned through the space, a navigation algorithm for path planning, a method of exploring the space, and some way of detecting people. We then had to implement a system which would combine all of these separate modules into a single system that would complete the task. In this section we present some background information about techniques that are often used to solve these problems.

A. Localisation

The aim of localisation is to obtain an estimate of the position of a mobile robot using sensor data [2]. While it is possible to use odometry data to do so, this data is inherently noisy and prone to error. In particular, odometry errors can arise from changes in the surface being traversed and the robot's weight, among others. An important point to note is that localisation can be performed with a prepared map, or generating a map in real-time. The much more complex latter problem is called simultaneous localisation and mapping, which we will not discuss here. See [3] for an introduction to the SLAM problem.

1) Bayes Filter: Many advanced localisation techniques are based on the Bayes filter, which uses a belief distribution $bel(x_t)$ to represent the state x_t. The calculation of $bel(x_t)$ at time t is dependent on $bel(x_{t-1})$ at time $t-1$, the last action u_t, and the last measurement z_t. In the first step, called the prediction step, the prior belief $\overline{bel}(x_t)$ is calculated. This step merges two probability distributions; the prior belief over the previous state x_{t-1}, and the probability of transitioning from that state to the posterior state x_t given that the action u_t was taken. This step does not take into account any measurement taken in the posterior state, predicting based solely on the knowledge of the action taken. In the second step, the measurement update, the posterior belief is calculated by multiplying the prior belief with the probability of being in the posterior state given that the measurement z_t was observed. The result of this multiplication is generally not a probability, and therefore requires normalisation using some constant α. As there are usually multiple posterior states, x_t is usually a state vector rather than a single state, and so the two steps will be applied multiple times in order to update the belief for each state being considered. The filter is recursive, requiring some idea of the initial belief $bel(x_0)$ at time $t = 0$. The initial belief is either a distribution centred on x_0, in the case where the initial position is known, or a uniform distribution over the space otherwise. As the Bayes filter is not restricted to finite state spaces, it is not possible to implement it for anything other than very simple problems. There are two families of algorithms for localisation, known as *recursive state estimators*, with various properties that permit the use of the Bayes filter in more complex estimation tasks [4].

Algorithm 1 Bayes filter [4]

```
1: Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$)
2: for all $x_t$ do
3:     $\overline{bel}(x_t) = \int P(x_t \mid u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$
4:     $bel(x_t) = \alpha P(x_t \mid z_t) \overline{bel}(x_t)$
5: end for
6: return $bel(x_t)$
```

2) Gaussian Filters: The basic principle of the family of Gaussian filters is the use of multivariate normal distributions, which can be formulated from a mean μ and covariance Σ, to represent belief. As a result, the assumption that the system is a linear Gaussian system is made; the initial belief must be a Gaussian, and both the state transition function and measurement probability must be linear functions. Although Gaussian filters can be extended to non-linear systems, they perform best when the system meets the assumptions made. One of the main advantages of such filters is the computational complexity, which is polynomial with respect to the dimensionality of the state space. The main disadvantage is that Gaussians are unimodal and therefore cannot represent situations in which there are multiple hypotheses; a situation that is often encountered in robotics. Examples include the Kalman filter and the information filter, which are derived from two different ways of representing Gaussians. Both of these filters can be extended to non-linear systems by using a Taylor expansion to produce linear approximations of non-linear functions. Mixtures of Gaussians can also be used to extend the Kalman filter to encompass situations in which multiple hypotheses are required, but each extension increases the complexity of the algorithm. This extensibility is one of the reasons for the popularity of the Kalman filter in state estimation problems. With some extension, the information filter is particularly suited to multi-robot systems where information from multiple sources must be integrated, but issues with complexity have resulted in the Kalman filter becoming the more popular of the two for the majority of problems [4].

3) Nonparametric Filters: In contrast to Gaussian filters, nonparametric filters discretise the probability distribution and do not require assumptions of linearity and Gaussian belief distributions. Instead, the state is approximated by a finite number of values which are taken from the belief state at any given time. The number of these values can be varied, and nonparametric filters converge to the correct state as the number tends towards infinity. Because this family of filters does not impose restrictions on the posterior density, they are useful for problems such as global localisation, which require the state to be represented in a complex form. Global localisation is the problem of determining the position of the robot without knowing its initial position, resulting in global uncertainty and the need for a multimodal belief distribution. Although as a result nonparametric filters are more computationally expensive than Gaussian filters, it is possible to vary the number of values used to represent the belief to suit the problem using *adaptive* techniques. Examples of these types of filters are the histogram and particle filters. The histogram filter decomposes a continuous space into some finite number of regions, each of which is assigned a probability based on the belief. Extensions include using dynamic decomposition techniques to use more coarse representations in regions with lower probability, and the use of *selective updating*, which updates only the parts of the space which are deemed important. Particle filters approximate the belief by drawing a number of hypotheses called *particles* randomly from the belief distribution. The most important part of the particle filter is the resampling step, which selects particles from the initial set proportional to an importance factor. This has the effect of concentrating particles in areas of high likelihood, reducing computational power spent in areas which are not relevant. Improving the sampling method and adapting the number of particles based on time and uncertainty lead to more effective and error-resistant particle filters. A property which is particularly useful to us is that particle filters are very easy to implement [4].

Algorithm 2 Basic Monte Carlo Localisation [4]

```
1:  **Algorithm MCL**($\mathcal{X}_{t-1}, u_t, z_t, map$)
2:  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:  **for** $m = 1$ to $M$ **do**
4:      $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
5:      $w_t^{[m]} =$ **sensor_model**$(z_t, x_t^{[m]}, map)$
6:      $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:  **end for**
8:  **for** $m = 1$ to $M$ **do**
9:      draw $i$ with probability $\propto w_t^{[m]}$
10:     add $x_t^{[i]}$ to $\bar{\mathcal{X}}_t$
11: **end for**
12: **return** $\bar{\mathcal{X}}_t$
```

4) The Markov Assumption: One of the reasons that these filters are so efficient stems from the assumption that the Markov property holds. While this may not actually be the case, the Bayes filter that forms the base of these filters is robust to the violation of some of the requirements of the property. The Markov property states that the future states of the system do not depend on past states. In other words, considering all previous states of the system provides no additional information for predicting future states; all prediction can be done using only the current state. Previous states do not have to be included in calculations, and as a result do not need to be stored, which leads to faster computation and lower storage space requirements.

5) Sensor & Motion Models: To actually use any of these filters to solve a localisation problem, it is necessary to model the sensors and motion of the robot. These models include some parameters representing the uncertainty attached to performing a specific action or receiving a certain measurement from a sensor. The motion model is used in the prediction step to determine the state transition probability $P(x_t \mid x_{t-1}, u_t)$; the probability of going from state x_{t-1} to x_t given that the action u_t was performed. The sensor model is used to calculate $P(z_t \mid x_t, map)$, which represents the likelihood of the measurement z_t being received in the state x_t on a given map. The model incorporates knowledge about the noise parameters of the operating environment and the sensor being used in the form of probability densities. For example, a range sensor might include densities for correct measurements, measurements of unexpected obstacles, sensor failures and random measurements [4].

Integrating the sensor and motion models into the particle filter results in an algorithm known as Monte Carlo localisation, otherwise known as MCL, the most basic version of which is shown in Algorithm 2, in which $\mathcal{X}_{t-1}$ represents the particles from the previous time step, M the total number of particles, $x_t^{[m]}$ the state of the mth particle with the action u_t applied to it via the motion model, $w_t^{[m]}$ the importance weight of the mth particle, and $\bar{\mathcal{X}}_t$ the resampled set of particles. We use an adaptive implementation of MCL provided by the ROS system.

B. Route Planning

1) Probabilistic Roadmaps: A probabilistic roadmap (PRM) is a multi-query motion planning algorithm for robots in static workspaces[5] which solves the problem of determining a path between start and goal configurations while avoiding collisions with obstacles in the workspace. The roadmap algorithm consists of two phases; the learning phase and the query phase.

In the first step of the learning phase, the construction step, a graph G with vertices V and edges E is constructed. Free configurations are repeatedly sampled from the workspace and added to V. A free configuration is one where the robot does not intersect with walls or other obstacles in the workspace. Once a specified number of configurations have been sampled, the vertices in V are connected. Attempts to connect two vertices v and n are only made if they are within a neighbourhood specified by a maximum distance D, and a connection is only made if the path between the two nodes is also valid, i.e. there is no configuration on the path where the robot intersects with walls or obstacles. If a connection is made, the edge from v to n is added to E. The second step of the learning phase is the expansion step, which is intended to improve the connectivity of the graph. This step can be skipped if a sampling algorithm which takes into account difficult (narrow) areas of the workspace is used in the construction step. Once the graph has been constructed, the query phase is entered. In this phase, the roadmap is used to find paths

Algorithm 3 Probabilistic Road Map Generation

```
1: **Algorithm generate_PRM**($map$)
2: $V$ = **sample_vertices**($map$)
3: **for** $v \in V$ **do**
4:     **while** $a < A \wedge$ **nconns**($v$) $< C$ **do**
5:         $n$ = **get_closest**($V, v$)
6:         **if dist**($v, n$) $< D$ **then**
7:             **if** $\neg$**neighbour**($v, n$) $\wedge$ **vpath**($map, v, n$) **then**
8:                 **connect**($v, n$)
9:             **end if**
10:        **else**
11:        **end if**
12:    **end while**
13: **end for**
```

Algorithm 4 Path Smoothing

```
1: **Algorithm smooth_path**($P, I, map$)
2: **for** $i := 0$ **to** $I$ **do**
3:     **for** $j := 0$ **to** $|P| - 2$ **do**
4:         $a \leftarrow$newpath($i$)
5:         $b \leftarrow$newpath($i + 1$)
6:         $c \leftarrow$newpath($i + 2$)
7:         **if freepath**($map, a, c$) **then**
8:             **remove**($P, b$)
9:         **end if**
10:    **end for**
11: **end for**
12: **return** newpath
```

between start and goal configurations s and g. In order to do this, s and g must first be connected to the graph using the same connection strategy used in the construction step. A graph search algorithm such as A* can then be used to find the shortest path between the two newly added vertices.

There are a number of ways in which the basic PRM algorithm can be improved. Most of these relate to the construction of the graph, as the graph is the part of the algorithm which most influences subsequent performance—if vertex placement is bad, then even if the graph search algorithm is good, it is possible that a path will not be found. The sampling method is particularly important, and many different techniques have been proposed to improve the coverage of the PRM. Roughly speaking, the techniques can be separated into uniform and advanced techniques. Uniform techniques include random sampling, sampling points on a grid, and splitting the workspace into cells and sampling some number of points from each of those subspaces. Uniform methods provide a simple but effective way of sampling, but may experience a drop in performance for more difficult workspaces, for example those with narrow corridors. In these cases, using advanced techniques such as medial axis sampling, which generates points at the midpoints between obstacles, can be effective [6]. The strategy used to connect vertices can also affect the structure of the graph, which in turn affects the query phase. Improvements to the query phase can be made by using different graph search algorithms, but unless the graph is extremely dense the query time is negligible. Path smoothing can also be used on the path found by the graph search algorithm to improve it further; our version is found in Algorithm 4. In some cases, a path may not be found with the graph being used, and although expensive, re-generating the graph can sometimes remedy this problem.

2) Rapidly Exploring Random Trees: An RRT is a randomised data structure and algorithm that is also used to solve path planning problems. RRTs are designed to handle nonholonomic constraints and high degrees of freedom [7]. Instead of sampling points and then attempting to connect them as in PRMs, RRTs are biased towards moving into unexplored areas of the state space by sampling points and being "pulled" towards them [8]. There are several planners for RRTs, including a bidirectional planner where the tree grows from the start and goal vertices and uses an aggressive heuristic to connect the trees. In contrast to the multi-query PRM, an RRT is a single query algorithm; it is rerun every time a query is made. The properties of RRTs make them particularly suited to problems involving robot arms and robots with nonholonomic kinematics. As our platform can be considered holonomic, we mention RRTs for completeness.

C. *Exploration*

frontier based techniques, exploration for AUVs

D. *Robot Vision*

II. Design

A. *System Structure*

MENTION ALGORITHM COMPLEXITY! brief ROS description, callback based system, finite state automaton

B. *Platform*

Stuff about the pioneer—available sensors, some data about its size, specifications, our additions to it. Kinect specs. Include a picture of the robot with the kinect on it.

III. Experimentation

A. *PRM*

inflated map - show inflated map superimposed onto the original map Redo experiment for sampling methods. short, medium, long path length. Display image of map with one of the routes displayed and show the difference between the sampling methods. Find the optimum route by sampling a massive number of vertices on to the space and then finding a route using that—the flattened path is then the most optimal route, and we compare the other routes to this route for each experiment.

Redid sampling experiments - now have comparison for neighbourhood, threshold and nearestN strategies on 4 different paths, including trying to get into the corridor. 5,10,20,30 maxconn for each strategy, inflation radius of 5 so that you can just about get into the corridor. random vertices:25,50,100,200,400,800, grid step 0.5,1,2,4, cell size 1,2,4,6, target per cell 2,5

REDO THIS ONCE THE ABOVE DATA IS COMPILED Best sampling: (approximate) optimum path found for each path by using grid sampling with a step of 0.2m and

50 max connections and neighbourhood connection strategy. Screenshots available in screenshots dir. Repeated experiments five times for cell and random, once only for grid. 5,10,20,30 maxconnections. Used results from previous experiments on the sampling strategy, but compared the best parameters for each. Also checked whether the corridor which causes issues was accessible when using each strategy as a measure of its ability to populate tight spaces. Not necessarily a good evaluation, since grid may be good on this map by accident, but terrible on others. new_prmlogs contains data.

B. Vision

C. Exploration

Coverage experiments info: cell and grid done for cell size and grid spacing of 10,8,6,4,2,1, with cell repeated three times for each. The fov max distance was 3.5, angle 57, minimum distance 0.3. Started at 2.80,18.97,40 in stage. Max move speed 0.7m/s, max rotation speed 0.4 rad/sec. Ran until the end of the exploration path was reached.

IV. Discussion

A. Performance

B. Potential Improvements

the path generated to do exploration could be improved by using heuristic techniques

C. Conclusions

References

[1] D. Kortenkamp, I. Nourbakhsh, and D. Hinkle, “The 1996 aaai mobile robot competition and exhibition,” in *AI Magazine*, vol. 18, 1997.

[2] W. Burgard, D. Fox, and S. Thrun, “Active mobile robot localization by entropy minimization,” in *Advanced Mobile Robots, 1997. Proceedings., Second EUROMICRO workshop on*, pp. 155–162, IEEE, 1997.

[3] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents Series, MIT Press, 2005.

[5] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[6] R. Geraerts and M. Overmars, “Sampling and node adding in probabilistic roadmap planners,” *Robotics and Autonomous Systems*, vol. 54, no. 2, pp. 165–173, 2006.

[7] S. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” tech. rep., Iowa State University, 1998.

[8] S. LaValle and J. Kuffner Jr, “Rapidly-exploring random trees: Progress and prospects,” 2000.