Chapter 6: Map building and path planning

Introduction

In many robotics applications and especially in the case of autonomous guided vehicles it is necessary for the robot to hold some sort of representation, or “map”, of its environment. The specific nature of this representation depends strongly on the kind of information one wishes to derive from this map. In this case, the map is used to retrieve navigational information for the robot. As a result, the used path planning techniques are closely related to the type of implemented map, which is why these two subjects are handled together in this chapter. The type of input – data, which is used for the map building process, will also have its influence on the type of algorithm used, as does the used control architecture. Considering all these different possibilities, it should not come as a surprise that there exists a vast multitude of paradigms for map building, yet all these can be categorized into two distinctive sets of theorems: the grid-based and the topological approach.

In theory

Grid or topological map?

Comparison

When using a grid, the environment is represented by evenly – spaced grids indicating, for example, the presence of an obstacle in the corresponding region of the environment. When using the topological approach, the environment is represented by graphs. Each node in such a graph corresponds to a distinct place or landmark. Arcs connect these nodes if there exists a direct path between them. Both methods have their advantages and disadvantages [14]; the most important ones are stated below:

Grid map:
+ These maps are very easy to build and to maintain
+ The relationship between map and environment is straightforward
+ Multiple viewpoints can easily be integrated by using a coordinate – transformation
+ Calculation of shortest path is fairly easy
− Very memory and space consuming since the complexity of the map does not depend on the complexity of the environment
Chapter 6: Map building and path planning

- As a result of the previous point the processing of these kind of maps can be very time-consuming
- The position of the robot itself must be known accurately
- Since a grid map provides a lower level of abstraction compared to topological maps, the returned data will require more pre-processing when used by symbolic problem solvers such as behaviour coordination algorithms.

Topological map:
+ Efficient planning, memory and space-saving, since the resolution depends on the complexity of the environment
+ Faster
+ The exact robot position isn’t that important
+ Very convenient interface towards symbolic problem solvers
- Quite difficult to construct and to maintain
- Requires recognition of landmarks and places: one must have the sensory equipment for doing this and even then, it is no simple task
- Paths calculated based on topological maps may not be optimal in terms of energy consumption or distance travelled

Grid maps

Grid maps are in most cases discrete, 2-dimensional occupancy grids, in which each cell has a value attached that marks the belief of finding an obstacle in the corresponding region of the environment. The cell values, or occupancy values, are determined based on consecutive sensor readings. The building process of grid maps can be divided into 4 distinct components:

1. Interpretation: The sensor readings must result in occupancy values for each cell. The interpretation process is facilitated by the use of the fuzzy logic based sensor-fusion component, which precedes the map-building procedure. Normally, one would consider using artificial neural networks for fusing the different raw sensor readings directly onto the map, but given the input parameters of the map-building procedure, this is no longer necessary. As not only mere position information regarding an obstacle or the target is provided, but also the standard deviation on this abstract measurement, this extra data must be taken into account usefully. This can be done by reconstructing the gauss-distribution on the map itself, so cells in the neighbourhood of the cell where the obstacle is reported will receive an occupancy value decreasing with their distance to this last cell. Of course, the dimensions of the obstacle-object on the map can be extended with a certain security distance taking into account that the robot is not really a point-object, or some suspicion regarding the sensor readings.
Chapter 6: Map building and path planning

2. *Integration*: Consecutive results of the interpretation process over time are integrated to come to more reliable maps, which can be performed by a whole number of techniques [16]. The integration of sensor measurements is also a process that the preceding data-fusion component has already taken care of.

3. *Position estimation*: The position of the robot on the map must generally be recalculated and re-estimated after every step. However, since the used map is relative to the robot, the robot position does not need to be referenced to the map after every move; it is the map that is re-referenced to the robot. This approach is used because the map will be initially empty, so the robot would not be able to reference its position to a certain landmark anyway. Changes in the map between movements will thus reflect newly found obstacles, improved position measurements of the target or an obstacle and dead reckoning errors.

4. *Path planning*: Based on a chosen criterion like minimal energy consumption or shortest path, the robot must find a path based on the map towards his goal. This last process is further handled in a following paragraph.

**Topological maps**

Topological maps are built on top of the grid maps. The idea is to partition the free-space of a grid map into a small number of regions, separated by critical lines. These critical lines correspond to narrow passages such as doorways. The partitioned map is then mapped into an isomorphic graph. Since topological maps are difficult to maintain, whereas the map will have to be changed continually in the used application; and since they require the recognition of certain landmarks, the implementation of topological maps is not a realistic option for this case, so they will not be further discussed.

**Path planning techniques**

**Classical methods**

To choose the path planning method used by the robot, first a study was made of some different possible approaches; these include vertex graph path planning, free space navigation, grid based navigation, distance transforms, stream field method and heuristic navigation. A brief introduction to these techniques that were not restrained can be found in appendix D.
Chapter 6: Map building and path planning

**Recursive algorithm**

The first approach tried out to deal with the path-planning problem was to implement a recursive algorithm calculating all the possible reachable points. The algorithm used to compute these steps as shown on figure 2 was therefore extended to calculate paths to a designated target. However, in view of the recursive nature of the algorithm, calculation for longer distances took excessive amounts of time. An adjustment was made to work with linear paths at greater distances, so the recursive calculation was only made when nearing the target. This resulted in paths as shown on the next figure:

![Figure 53: Path calculated with a recursive algorithm](image)

However, this approached lacked the flexibility to easily deal with more complex environments, so this track was abandoned for further implementation.

**Potential Field Navigation**

This is the main navigation technique that is used by the robot; this choice was made because the potential field method provides a quite natural and logical framework for addressing path-planning problems. Moreover, the potential field method is one of the very few methods able to provide the required map robustness, which is needed, as the robot will have to deal with highly incomplete environmental data due to the limited field of view of its sensors. Potential field navigation techniques make use of artificial forces: repulsive forces at impassable areas or obstacles keep the vehicle away; an attractive force at the goal point moves it towards the goal.
The potential field has to be tuned in a way such that the robot can never be dragged inside an obstacle and keeps a specified security distance, but moves towards the goal from all points in the environment. This is limited by the existence of local minima; there the resultant force on the robot disappears and a solution path cannot be found. Local minima exist for example on the opposite goal side of obstacles as shown on the following figure.

Figure 55: Local minimum on a map with one obstacle and one target
Chapter 6: Map building and path planning

The local minima problem has not been solved completely to this day, though considerable attention has been given to it. One way to solve the problem is to introduce a random movement of the vehicle hoping to escape a local minimum; another is to relocate the goal temporarily when stuck in a local minimum, or to mark areas that have been visited already as impassable. All these methods do not avoid the existence of local minima and are not always successful in their solutions. The used technique for the robot to avoid local minima is simple, yet very effective: the robot is considered an obstacle itself. By doing this, the robot creates a repulsive force away from the current position and will not get stuck in local minima. A convenient side effect of using this technique is that the progress the robot is making is made visible on the map. A drawback from using this method is that even more of the correlation between the generated map and the physical reality is lost.

Potential field navigation is very suitable for local navigation since the environment has only to be known in the vicinity of the vehicle and only a short piece of the path is calculated with each evaluation of the force fields. This implicates that it is not necessary to calculate on the global map with each step, provided the boundary conditions are known, or that they are not that important for the given problem. Unfortunately, this is not the case; the boundary conditions for local maps used by the robot change in an unpredictable manner, since new obstacles can arise at any time, so it will always be necessary to work with a global map.

Path generation can be implemented easily and effectively, since this calculation can take direct use of the cell values of the map. The basic idea used in implementing the potential field method is to find a harmonic function \[15\] [18]. A harmonic function \(\phi\) on a region is a function that satisfies Laplace's equation:

\[
\Delta \phi = \frac{\partial^2 \phi}{\partial x_1^2} + \frac{\partial^2 \phi}{\partial x_2^2} + \ldots = 0
\]

Having continuous second derivatives in the interior of the region.

It is possible to prove analytically that a harmonic function has neither a minimal point nor a maximal point in the interior of the region. Here the nature is explained intuitively. If a point is a minimal point, then a sum of the second derivatives has the positive sign. If a point is a maximal point, then a sum of the second derivatives has the negative sign. Therefore, if a sum of the second derivatives is zero, the point is neither a minimal point nor a maximal point. Consequently, there can be theoretically no minimal point in the interior of the region that satisfies Laplace's equation. In practice, adding arbitrary obstacles and targets to the potential field breaks the nature of the harmonic functions, so local minima do arise, but, as stated earlier, the technique of considering the robot as an obstacle deals with this problem adequately.
Chapter 6: Map building and path planning

**Behaviour Based Navigation**

Behaviour Based Navigation for mobile vehicles is based upon breaking up the navigation task into basic behaviours, e.g. "avoid obstacles", "follow wall", "stay on path", etc. The combination of these "micro-behaviours" results in the desired "macro-behaviour" of the vehicle. This approach results in solution paths that may be hard to foresee, but present an opportunity for reactive navigation independent from geometric path modelling models.

These behaviours are defined as "Motor-schemas". Motor-schemas origin in psychology and neurology and describe the interaction between perception and action of living beings. Motor-schemas are usually implemented using artificial potential fields. Each Motor-schema is implemented using a separate field or field property, e.g. an attractive force of the goal reflects “Move-towards-Goal”. Combining all Motor-schemas or behaviours is usually done by simply adding all forces at the vehicle’s position vectorially. One advantage of Motor-schemas is that they can be activated or de-activated separately as desired. For example, when travelling over a bridge a "stay-on-path" behaviour is essential, but it may be de-activated when travelling in a large area of free space in order to give the vehicle more flexibility in its path. Alternatively, the schemas might be given different priorities in case of opposing objectives.

The robot is actually given two such behaviours:

1. Head straight for the target when there are no obstacles in the way and the robot is aligned towards the target.
2. Search a path using the potential field method in other situations.

The followed behaviour is determined by the sensor fusion component. More specifically, as soon as the explicit accuracy bounds method determines that the ultrasonic sensors are measuring the target and not an obstacle, the second behaviour is chosen. As these sensors have a very limited opening angle, the robot must be more or less orientated towards the target in this case.

The reason for implementing this behaviour-based navigation is that it is not very intelligent to make time-consuming calculations to find a path towards the target if this target is straight ahead.

**In practice**

**Defining the map parameters**

**Received Input**

The map building procedure gets its input parameters from the fuzzy logic based data-fusion component, which fuses the sensor readings of the 4 abstract sensors. This fusion procedure delivers the following output-data as an input to the map building process:
Chapter 6: Map building and path planning

- Distance to the target
- Standard deviation on this distance
- Angle to the target
- Standard deviation on this angle

- Distance to an obstacle
- Standard deviation on this distance
- Angle to an obstacle
- Standard deviation on this angle

**Required Output**

The map building and path-planning module must return the best move the robot can do as a step towards reaching the goal. There is no need to perform a full path calculation every time, since it may be expected that the map will change as new information is gathered after the movement, so the rest of the path will become useless. This continuous alteration of the map is caused by the very limited field of view of the robot sensors, so the robot will have to manage with very incomplete maps and will have to recalculate a new path to the target after every move, since more information will be available every time.

It is also not necessary for the robot map to be a very exact representation of the physical reality, the goal is to do path planning, not to output detailed CAD-drawings of the explored environment.

**Grid parameters**

The number of cells used is an extremely important parameter in the map building process. For obtaining a decent resolution, the number of elements must be high enough. At a first instance, it seems logical to work with a map that has the same resolution as the sensors used. It is clear that a high resolution is always wanted, and consequently a lot of grid cells, but there’s an extremely restrictive factor to be taken into account here: time. When using a direct method (e.g. Gauss-method) for solving the system, time consumed for calculation is proportionate to $N^3$, $N$ being the number of grid cells. By using the Jacobi or Gauss-Seidel iterative techniques, one can reduce this to $N^2$, when using optimal relaxation to $\frac{N}{\sqrt{N}}$ and with multigrid even to $N$ [21].

The map used by the robot is a 250 x 250 map representing a 5m x 5m area, so the resolution is 2cm, which is about the sensor resolution level of the ultrasonic sensors. The most important guide for determining the grid distance was to make sure that two reachable points by the robot in the environment should be represented by two different cells on the map. Otherwise, problems could arise with the path planning procedure. The robot takes steps of 23cm (= 11.5cm on the map) and can turn over an angle of about 16°, so two reachable points are always more than 2cm apart.
A classic benchmark for finding a good map resolution is the so-called “doorway passing” problem. The resolution necessary to be able to pass through a door expressed as the minimum length reflected in the map - or the maximum side length of a cell in a grid-based map - is in general determined to:

\[ s_{\text{min}} = \frac{w - d_{\text{robot}} - d_{\text{security}}}{2} \]

Where \( w \) is the width of the smallest door that has to be passed, \( d_{\text{robot}} \) the diameter of the (assumed to be circular) vehicle and \( d_{\text{security}} \) the security distance the robot has to keep to each obstacle. Our robot is hardly circular, so this \( d_{\text{robot}} \) is arbitrarily chosen to be the ultimate distance from the front feet to the measurement reference point, which is at the base of the camera socle. When applied to the presented robot and its parameters, the above formula enables to calculate the width of the smallest door that can be passed:

\[
\begin{align*}
  w &= 2 \cdot \delta_{\text{min}} + d_{\text{robot}} + d_{\text{security}} \\
  &= 2 \cdot 2\text{cm} + 30\text{cm} + 15\text{cm} \\
  &\approx 50\text{cm}
\end{align*}
\]

However, the values of 30cm and 15cm used respectively for the robot diameter and the security distance are quite arbitrary, so no exaggerated value should be accorded to this calculation.
Chapter 6: Map building and path planning

Implementation techniques

The choice for the potential field method implicates that the Laplacian $\Delta \phi = 0$ of the presented field will have to be computed. These techniques were proposed for dealing with the problem:

Multigrid method

As stated above, the multigrid method is certainly the fastest of all the iterative methods. The basic idea is quite simple: the iteration is performed on multiple grids, each counting half as many points as the former, so the low frequency errors, which cause the other iterative methods to converge so slowly, are damped quickly on a coarser grid. The difficulty is to preserve the truncation error of the finest grid and to make sure that no new high-frequent errors are introduced. Since the programming of this method is not very simple and only existing source code for solving linear systems was available, this technique could not be used.

Analytical method

An interesting, yet quite unknown approach for solving the Laplacian is the analytical way [20]. The basic idea here is to write a general solution of the problem as a function of some unknown coefficients, which are calculated by performing a least-squares optimisation considering boundary conditions and singularities. This method would fit the map-building problem as posed for the robot nicely, since the used map consists typically of singularities as obstacles and target, and that is especially the kind of problem this method is designed for. As this technique was only learned about after the map building and path-planning module was actually written, it could not be introduced into the program structure due to a lack of time.

The Gauss-Seidel method

The iterative Gauss-Seidel method presents a technique to calculate a potential field in two-dimensional space. As this is the method actually used by the robot for its map calculations, this process is discussed more in detail below.  
To begin, the Laplace equation must be transformed into a discrete form. If the step sizes are all equal, this can simply be done by writing:

$$\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j} = 0$$

While

$$\phi_{i,j} = \frac{1}{4}(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1})$$
This equation illustrates that a potential on a mesh point is the mean of the values on the adjacent points. In order to satisfy the Laplacian over the whole region, the Gauss-Seidel iterative method is applied:

\[
\phi_{i,j}^{(n)} = \frac{1}{4} (\phi_{i+1,j}^{(n)} + \phi_{i-1,j}^{(n)} + \phi_{i,j+1}^{(n)} + \phi_{i,j-1}^{(n-1)})
\]

Where \(\phi\) is a numerical solution on the mesh point \((i, j)\) obtained from the \(n\)th iteration of the equation.

In order to speed up the calculations, the Successive Over-Relaxation (SOR) was used. The idea is to artificially increase the change between the new and the old cell value with a certain factor \(\omega\).

\[
\phi_{i,j}^{(n+1)} = \phi_{i,j}^{(n-1)} + \omega \left[ \frac{1}{4} (\phi_{i+1,j}^{(n)} + \phi_{i-1,j}^{(n)} + \phi_{i,j+1}^{(n)} + \phi_{i,j-1}^{(n-1)}) - \phi_{i,j}^{(n-1)} \right]
\]

An optimal value for \(\omega\) was found by trial and error; since this value is only 1.3, the performance gain is not that impressive, but it is noticeable.

In order to avoid a creation of an unexpected minimal point due to a numerical calculation error, an initial value is set on each mesh point in the interior of the free space before the Gauss-Seidel iteration begins. This initial value for the free space is a very time-determining factor in the iteration process.

It’s clear that when this value is quite different from the solution, the iteration will take more steps. This is the reason why the implemented program uses a high initialisation, meaning the free-space cells are set to a value closer to the obstacle (=high) - level than to the target (=low) - level. Concrete, short integers are used for the cell values as experiments with chars showed a lack of resolution, so boundaries and obstacles are set to a value of +32767 and the target is given the value –32768. For a faster execution, the free-space cells were not initialised to 0 or 1, but to 25000. In consequence of the initial condition, a lower potential value propagates from goal point while the iteration progress. As a result of the propagation, the value of a point that is closer to goal point becomes lower. According to the boundary conditions, the potential field takes a high value at the surface of obstacles and takes a minimum value at the goal point.

A field of discussion is the exit condition for the iteration loop. Normally, one would stop the iteration as soon as the gradient of the potential field around a start point becomes large enough to determine the direction to a goal, so the time required for the calculation of the potential field would depend on the location of the starting point. But it is also important to make sure that a newly added obstacle can carry through its influence to the position of the robot, so a certain number of iterations is required in every case. Another point to keep in mind is that during the calculation of the very first step the map changes a great deal, since the process must start with the given initial conditions, so the gradient of the potential field around a starting point could also undergo some dramatic changes before converging to a stable value. This is the reason why initially the change was calculated between newly found maps and the previous version. When this change became low enough, the iteration could be halted. The number of iterations needed for the first and the later map calculations to come to stable maps were noted and brought into the program. Thus, the time consuming operation of recording the changes made to the map is no longer necessary in the final program.
Programming issues

The map building and path planning process is the time determining step in the program flow as it involves a long iterative calculation of the potential field. As a result, great interest has been given to improving the performance of the algorithm and to speed up the calculations, numerous techniques were used:

- Successive overrelaxation
- High initialisation
- Prior determination of the required number of iterations
- More efficient memory management
- Behaviour based navigation

Most of these techniques are already discussed and the results are clear as the time needed to calculate the next step was brought down from 8.5 minutes to 8.5 seconds. This time delay may still seem too much, but tests with the program on a more modern computer (AMD K7 750Mhz with 133MHz SDRAM) showed execution times within the time delay of two seconds, which is needed in every case for steering the pneumatic valves.

Experimental results

The actual working of the map building and path-planning module can be shown no better than by presenting the results of a real-world example. The environment set up for this experiment is sketched on the following figure:

Figure 57: Testing environment
The charts presented on this page show the potential field map at different stages along the way towards the target. These different stages are also marked on the above figure.

*Figure 58: Charts of the potential field as the robot is advancing in the environment*
On the final potential field graph shown on the next figure, one can clearly see the path followed by the robot. Note also the correspondence between the environment as shown in figure 55 and this potential field representation.

Figure 59: Potential field after completing a run

An interesting situation arises when the doorway passing is made smaller. Eventually the robot will decide to go the other way round as shown on the chart below.

Figure 60: Robot making a large detour before reaching its target
Chapter 6: Map building and path planning

This behaviour can be explained by following the robot logic systematically, which is done here for the key stages denoted on the figure above:

1. The robot starts here
2. The robot chooses to go left as the ball is located to the left of the central axis
3. The robot starts turning left as the detected obstacles on its right side cause a high potential there. It keeps turning left, eventually turning a 180°
4. The robot manoeuvres itself in between a passing on the right side without colliding with one of the obstacles
5. The robot reaches the target point

This conduct may seem erratic to the reader, but note that following the right passageway around the central obstacle was in fact the shortest path solution. However, the robot has no means of knowing this a priori due to the very limited field of view of its sensors. So it is normal for the robot to turn left first, to realise its mistake at a later stage as it has gathered more environmental information and to follow the shortest path eventually. In spite of all these justifications, it cannot be denied that using the potential field navigation technique together with the extremely limited field of view of the robot sensors, results in sometimes less logic behaviour. However, it must be noted that even in conditions where the robot intelligence fails in finding the actual shortest path, the robot still succeeds in reaching its target after a while without any collisions on the way.