Using Quadtrees for Realtime Pathfinding in Indoor Environments

Julian Hirt, Dominik Gauggel, Jens Hensler, Michael Blaich, and Oliver Bittel

University of Applied Sciences Konstanz, Germany
Laboratory for Mobile Robots
Brauneggerstr. 55 D-78462 Konstanz
{jhirt,gauggel,jhensler,mblaich,bittel}@htwg-konstanz.de
http://www.robotik.in.htwg-konstanz.de/

Abstract. During the last few years mobile robots got more and more important to solve different tasks in outdoor and indoor environments. To solve these tasks one very essential issue is to get from one point A to another point B as fast as possible. To find the least expensive route to the goal the pathfinding process needs a full state space information about the environment. With this information we can use optimally efficient algorithms like A* to find the route, but this might be very expensive on memory usage and time response. therefore we need to use other data structures to represent the whole information about the environment. This paper shows how the usage of quadtrees improves performance in terms of computation speed, memory requirements and path length.

1 Introduction

The Eurobot Challenge [1] is an international open mobile robotics contest which is taking place every year. The task which the amateur teams have to handle in 2010s challenge is to build a robot which collects virtual food on a 2m x 3m board. The robot has 90 seconds to collect as much food as possible while another robot does the same on the same board. All the robots have to do so completely autonomous. therefore we needed an algorithm which finds the best path on the board between two points.

Pathfinding is a fundamental problem for mobile robots. No mobile robot could work on almost any task without moving to another point in the environment. therefore different algorithms exists which find good solutions to get from one point A to another point B. This problem has been defined as the problem finding a path linking two vertices of a graph [2]. There are currently different algorithms to solve the pathfinding problem like the Wavefront or A* algorithms. One of the most common solutions is to implement the A* search algorithm [3]. This algorithm has been first described in 1968 and has been used in many different ways. A* is optimally efficient for a certain heuristic. Yet, in practice, pathfinding using A* might have huge problem with memory and time

requirements. This affects a mobile robot especially when routes have to be calculated very frequently.

The navigation process for mobile robots can be separated into two levels - global and local planning. The local planning is responsible for avoiding obstacles [4], reacting to sensory data and driving towards a subgoal. This paper focuses on combining the local and global navigation by using one map for both types.

The literature gives many different approaches to overcome the limitations of the A* search algorithm. Frequent advices are changing the heuristic, like Euclidean Distance or Manhattan Distance, caching parts of calculated routes or especially changing the search space representation. The most promising aspect is to change the underlying representation of the environment which is usually a grid [5]. The complexity of the pathfinding process expands with the size of the environment and their accuracy. For a higher accuracy more cells, which represents a certain area, are needed. By discretizing the world, the computational complexity of pathplanning can be controlled by adjusting the cell sizes. By enlarging the cells more space, which might not contain any obstacles or impassable regions, gets lost for the mobile robot. What we need is a data structure which represents the environment as accurate as possible with the least amount of nodes. So data structures with uniform cell sizes are not practical for this situation. Efficiency in map representation can be obtained by the use of quadtrees. In this paper we discuss our approach to solve the complexity of the pathfinding process using the A* search algorithm by using quadtrees to represent the environment of a mobile robot. We present our results of a path planner used for the Eurobot contest.

2 Map Representation

The discretization of a world offers the possibility to control the complexity of pathfinding algorithms as well as a flexible representation of regions which contains obstacles or just impassable regions. The simplest representation of a 2D world is a regular grid of squares. They support random-access lookup for any coordinates of a point in a map. To represent a world in a grid accurately the size of cells have to be as small as possible. On the one hand we need this accuracy to guarantee the quality of the solution path, but on the other hand the number of cells boost the complexity of memory and time requirements.

2.1 Regular Grid

Using a regular grid means to divide the entire map into uniform sized squares. Every square has four or eight neighbors, by adding the diagonal squares as well. This means if a region consists of a certain number of squares but shares the same information, it is still represented by the amount of squares. This represents the world inefficiently because it makes the search process very expensive. Moreover, a regular grid allows only eight angles which results in sudden direction changes.

It might be possible to straighten a path through a clear area but there is no guarantee that this smooth path is the optimal path. How a map can be presented in a regular grid is demonstrated in figure 1.

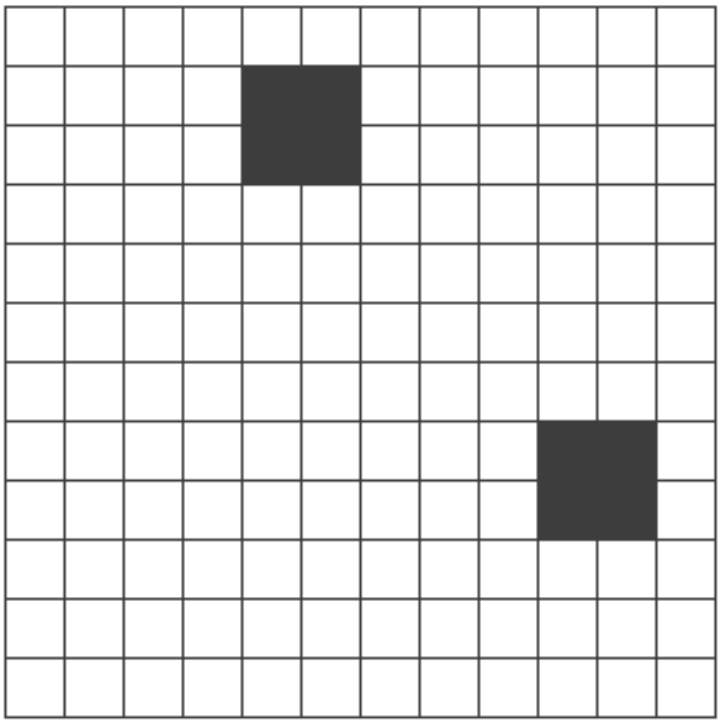

Fig. 1. A regular grid for a map which contains two obstacles. It consists of 144 uniform cells.

2.2 Quadtree

To reduce the amount of squares to represent a world, we use a quadtree, which is a different data structure to describe the map. Compared to a regular grid a quadtree has no uniform sized cells. It subdivides the whole map into four equally sized regions. If one of these new regions contains an obstacle it is subdivided into four other regions as well. A region is recursively subdivided until a region does not contain any obstacle or reaches a certain minimum size. Quadtrees allow to represent a large clear area with one single cell. Instead of the regular grid it describes a world efficiently but at the cost of quality. Paths generated by quadtrees are suboptimal because they are constrained to use the centers of the cells as shown in figure 2.

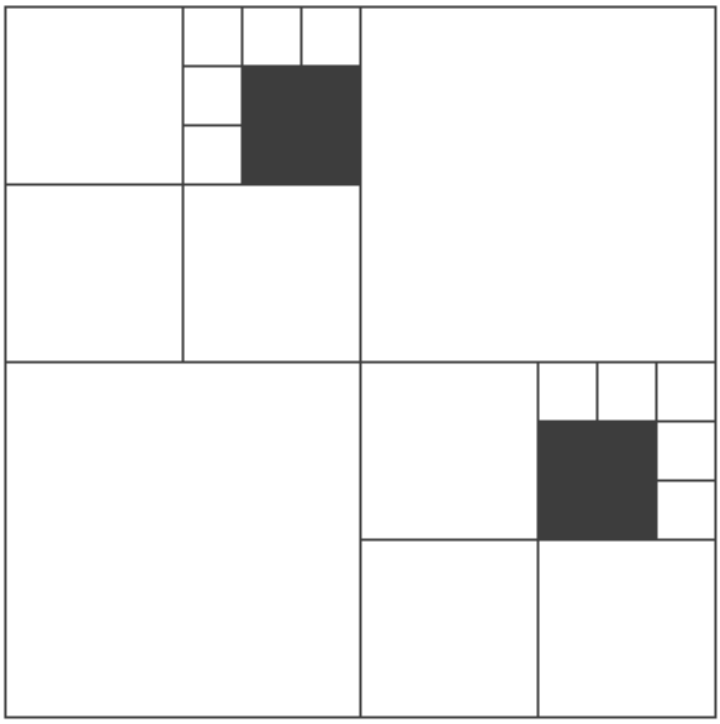

Fig. 2. A quadtree for a map which contains two obstacles. It consists of 26 cells.

3 Pathfinding in a Quadtree

In our first approach we implemented a regular A* search algorithm. With a resolution of 1cm per pixel we represent our board by a regular grid with 60000 cells. Using this regular grid with an A* algorithm takes 0.75 seconds to calculate a path from one corner to the opposite corner. During a match of 90 seconds we have to drive to around 30 goals. Hence, 0.75s * 30 takes too much time to just calculate the paths. Especially because the board is very dynamically. Obstacles on the board can move and particularly the other robot might be a problem. Our robot is driving with up to 1m/s. At this speed it is too dangerous to drive 0.75s while calculating a new path. therefore we implemented a quadtree in our second solution to solve this problem.

To use a quadtree in our pathfinding process, we have to generate a quadtree out of the regular grid every time we need to run the A* algorithm. It subdivides the world into subregions until a region does not contain any other obstacles. This reduced our amount of nodes enormously from 60000 down to 990. An example for a calculated path between two corners is shown in 3. therefore we tested the system as a Player [6] driver on our robot. Each of the green squares

represents one node in our quadtree. The violet lines show all squares which were visited by the A* search algorithm. The actual path which was found by the A* is the red line in figure 3. In order to use the quadtree for the local navigation as well, we need to update our grid every time the map changes. Thus we have to regenerate the quadtree to get a valid path for new goals.

3.1 Path Relaxation

As you can see the violet and red lines are from one center of a square to another center. This might give us a suboptimal path which we optimized by using the Split and Merge algorithm [7]. For the merge part of this algorithm we have to have a look at the centers of three following squares (s1,s2,s3) which are used in our path. If dist(s1,s3) is smaller than dist(s1/s2)+dist(s2/s3) we can merge them if there is no obstacle on the way from s1/s3. This smoothens our path and gives us a better path. This optimized path is shown in figure 3 as the cyan line.

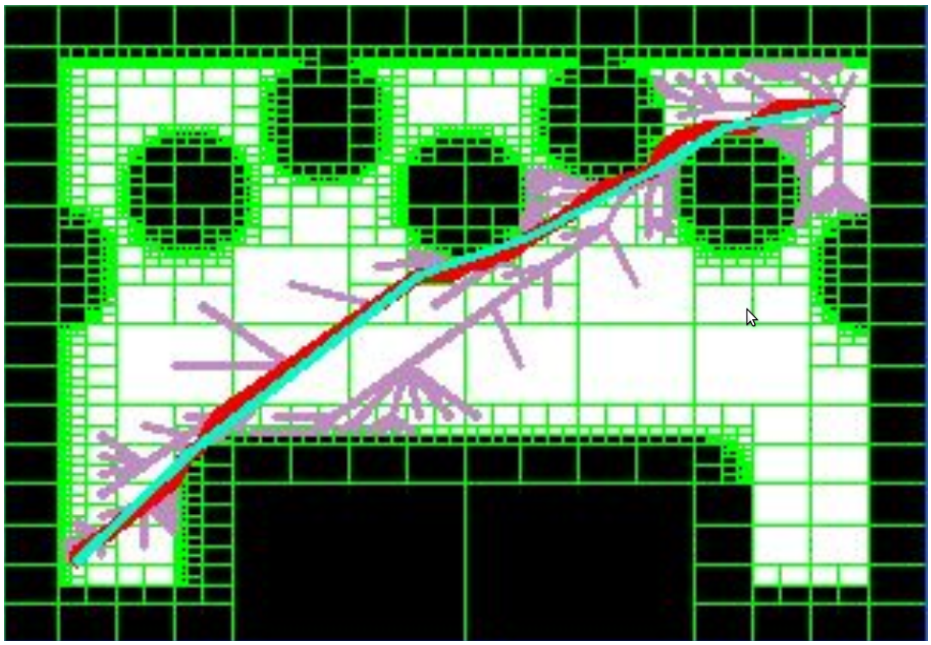

Fig. 3. Quadtree for calculated path between corners.

4 Experimental results

For the following results we used an Intel mainboard with an Atom 330 processor (2x1.6GHz) and 2Gb memory. Instead of just running the A* search algorithm

on a regular grid, we need three different steps using a quadtree. The breakup of the computation times is shown in table 1. The first step is to generate the actual quadtree, which takes 97.5% of the process time. To find a path with an A* algorithm needs only 2% which is about 0.0004s. This shows us the very enormous improvement compared to a regular grid which needed 0.75s. To simplify the resulting path takes another 0.0001s which is only 0.5% of the whole process time. Compared to the regular grid the quadtree needs only 2.73% of the time what we need to calculate the same path on a regular grid although we have to do two more steps.

Part	Time used	%
Generate Quadtree	0.02s	97.5 %
A*	0.0004s	2.0 %
Split And Merge	0.0001s	0.5 %

Table 1. Breakdown of computation times using a quadtree.

Type of Representation	Time used	%
Regular Grid	0.75s	100 %
Quadtree	0.0205s	2.73 %

Table 2. Comparison of the computation speed between A* on a regular grid and a quadtree.

Type of Representation	Amount of nodes	%
Regular Grid	60000	100 %
Quadtree	990	1.65 %

Table 3. Comparison of the memory requirements between A* on a regular grid and a quadtree.

5 Conclusion and future work

We have implemented a method for a pathfinding process on a 2m x 3m board for a mobile robot. Our solutions combines the optimally efficient A* algorithm with the quadtree. This gives us the possibility to find an almost optimal path

between two points in much shorter time than an A* in combination with a regular grid. The quadtree data structure optimizes the representation of our map by partitioning the map into non uniform sized cells. Although this might not result in an optimal path, the savings in time and memory requirements are way more significant. Even the time to generate a new quadtree does not affect that.
Actually, we might be able to improve the efficiency of our algorithm in our future work. It is not necessary to generate the whole new quadtree after the map changed. It is possible to generate just the regions which were changed. This might give us another huge improvement in the computation speed.
Another point would be to have a look at the framed quadtree [8]. We might be able to replace our Split and Merge algorithm by implementing a framed quadtree. With this technique we might be able to smoothen our resulting path even more.

References

1. Sciences, E.P.: 02/27/2010. http://www.eurobot.org (2010)
2. Niewiadomski, R., Amaral, J., Holte, R.: A performance study of data layout techniques for improving data locality in refinement-based pathfinding. The ACM Journal of Experimental Algorithmics (2004) 1–28
3. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics (1968) 100–107
4. Kunchev, V., Jain, L., Ivancevic, V., Finn, A.: Path planning and obstacle avoidance for autonomous mobile robots: A review. Lecture Notes in Computer Science, Springer Verlag (2006)
5. Yap, P.: Grid-based path-finding. Lecture Notes in Computer Science, Springer Verlag (2002)
6. Collett, T.H.J., MacDonald, B.A., Gerkey, B.: Player 2.0: Toward a practical robot programming framework. In: Australasian Conference on Robotics and Automation, Sydney (2005)
7. Thorpe, C.E.: Path relaxation: Path planning for a mobile robot. AAAI-84 Proceedings. (1984)
8. Yahja, A., Stentz, A., Singh, S., Brumitt, B.L.: Framed-quadtree path planning for mobile robots operating in sparse environments. In: IEEE Conference on Robotics and Automation (ICRA), Leuven, Belgium. (1998)