Tactical Path Finding in Urban Environments

Arno Kamphuis, Michiel Rook, Mark H. Overmars

Department of Computer Science, Faculty of Science

Utrecht University, Padualaan 14, 3584 CH Utrecht, The Netherlands

Telephone: +31 30 2531454, Fax: +31 30 2513791

E-mail: arnok@cs.uu.nl, mlrook@cs.uu.nl, markov@cs.uu.nl

Abstract

In this paper we address the problem of finding paths for small groups of characters in an urban environment taking tactical information into account. This approach presents a way to plan the paths using an existing road map and extending this to a network of corridors. The road map is searched for a globally useful path, after which the corresponding corridors are traversed to find a tactically sound path for all characters in the group. Our approach is capable of utilizing nearly 100% of the available free space in the environment, including almost all tactically useful locations. The generated paths for the group of characters are realistic in the sense that they mimic paths that real humans would follow. After a pre-processing step, the approach is able to produce the paths in real-time.

Keywords

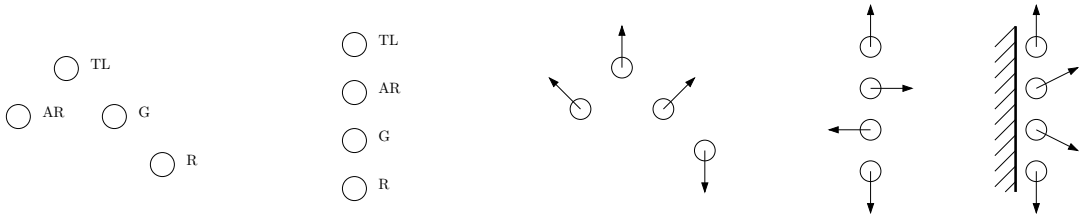
Games, Path Finding, Tactical Information, Formations

1. Introduction

Advanced games and simulation environments need to be able to plan/generate paths for a large number characters. In order to retain the suspension of disbelief [19] for users these paths should be believable or realistic. However, this is not an easy task, especially when tactical information is needed.

In this paper we focus on finding paths in tactical training simulations and games. In order to do this realistically we need to rely on military tactics, techniques and procedures. These tactics, techniques and procedures are documented in the so-called "doctrinal" literature. This literature covers four main areas:

- **Military doctrine** Fundamental principles by which the military forces, or elements thereof, guide their actions in support of objectives.
- **Tactics** The employment and arrangement of forces in battle.
- **Techniques** How to carry out tactics.
- **Procedures** How to perform tasks.



(a) The wedge and file formations of the fire teams. The abbreviations stand for: TL - Team Leader, AR - Automatic Rifleman, G - Grenadier and R - Rifleman.

(b) The fire sectors in different formation and environment situations.

Figure 1: The formations and fire sectors of a four-member fire team.

This literature can be found the training manuals of many military organization of many states. Many of the manuals used in the modern United States Army are freely available in print as well as on line [4].

Armed forces are probably the most structured organizations in the world. Such organizations have strong hierarchies, from top to bottom. Almost at the bottom are the fire teams. Fire teams are the smallest elements of some modern armed forces. A fire team allows speed, flexibility and control at the small-unit level. It consists of a team leader, an automatic rifleman, a grenadier and a rifleman or assistant automatic rifleman. Fire teams work in two basic formations: the wedge formation and the file formation (see Figure 1(a)).

The modern world is one of urbanization, i.e. the growth of population in urban areas, and the movement of people from rural to urban areas. This means that, in harsh contrast to previous times, armed conflict, both in the real world and in the virtual (training/simulation) world, will be situated in urban environments. To train todays and future armed forces and to generate realistic computer games we need to look at how urban conflicts are fought. Already in the late seventies, field manuals were written to train and educate soldiers in the complexities and difficulties of modern urban combat [5]. The term Military Operations on Urbanized Terrain (MOUT) was, and unofficially still is, the term for these operations, although today they are officially called Urban Operations (UO) [6]. In order to not being exposed to all types of fire, a soldiers needs to avoid being detected by the enemy. Both concealment and cover are very important in this. Concealment is the prevention of observation by the enemy, while cover is the process of making sure that enemy fire can't hit the soldier. The first, concealment, is achieved by using light, noise and movement discipline, as well as camouflage (both natural and man-made). The second, cover, can be achieved by using natural features such as trees, rocks, holes and man-made objects such as trenches, walls, rubble, fences and specially prepared fighting positions. Since the soldiers move through an urban environments they are potentially vulnerable to fires from all sides, as well as from above. All avenues of approach should be covered, and every field of fire of the team (each soldier must maintain and observe his/her own fire sector) should be observed. Figure 1(b) shows example fire sectors in standard combat situations.

1.1. Previous and Related Work

Motion planning, or more precisely, path finding has received much attention in the scientific community[15, 9, 8, 7]. Especially in robotics, the main focus has been on generating feasible paths, preferably short, smooth and with a minimum of clearance [3]. However, this is

(partly) in conflict with our goal for tactical path finding. In tactical path finding, the character should be able to move close to obstacles and make detours to avoid dangerous areas.

Crowd simulations [10, 16] and flocking/steering techniques [13, 14] are related to path finding. Lately, both have received quite a lot of attention. Advances in crowd simulation make it more and more possible for use in path planning. However, in general, crowd simulations lack the ability to guarantee that a path exists, i.e. there is no definite control of the path the characters are going to follow. The same arguments holds for flocking and steering methods, although this problem is partly solved by combining them with global path planning methods [1].

Figure 2: An example of a fire team consisting of 4 soldiers from our experimentation software.

Tactical path planning has been a major problem in contemporary computer games. Several techniques have been proposed that use influence maps on grid or graph structures [2]. However, these approaches lack scalability in the sense that when the environment gets larger and larger, the amount of data that needs to be stored will increase very much. Also, care should be taken when dynamically updating tactical information, otherwise this can result in long update time, especially when high detail in the environment is required.

1.2. Our contribution

We present an approach for planning paths for small groups of four characters (one fire team) in an virtual environment. We present a method for generating a data structure to store tactical information, which can be efficiently updated once new information is gathered.

1.3. Outline of the Paper

This paper is organized as follows. First, Section 2. describes our approach in detail. Then, in Section 3. we discuss the implementation details, the conducted experiments and the final results. Finally, we give our conclusions and discuss the future work in Section 4. .

2. The Approach

The approach consists of several parts. Figure 3 depicts the different processes in the method. Overall, these processes are similar to the steps in a normal path finding method, with some adjustments. Several of the processes are preprocessing phases, while others are processes that are executed in real time during simulation or game play.

The preprocessing steps in our approach have the objective to construct a datastructure that reflects the free space and its connectivity. This datastructure also allows tactical information to be stored efficiently, allows this information to be updated in real-time. The preprocessing steps are the following:

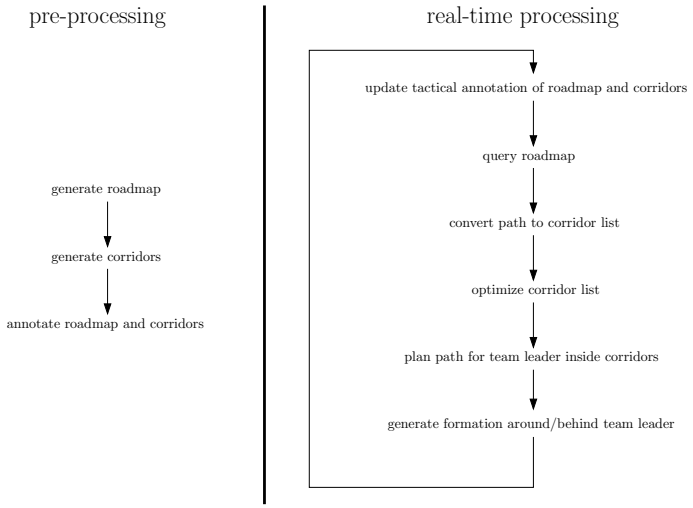


Figure 3: The processes of the approach. On the left the pre-processing steps, on the right the processes that are executed in real-time during game play or simulation.

1. First, a general roadmap should be constructed of the environments. This can be done in several ways, for example by hand or using a variant of the Probabilistic Roadmap Planner. Any roadmap construction scheme can be used. The final quality/success of the method is dependent on the quality of the roadmap in reflecting the topology of the free space. This step is a pre-processing step, and should only be executed once for a virtual environment.
2. Then, as a pre-processing step also, from this roadmap corridors are constructed. A corridor is a quadrilateral (see Section 2.1.) covering free space around a certain point on the roadmap. The corridors should be constructed such that the union of all corridors covers the free space as much as possible.
3. In this phase, the last pre-processing step, generates default tactical information values on the roadmap, and on the points in the corridors. The default values are updated in real-time during the use in a game or simulation, as soon as new tactical information is available (Section 2.2.).

During game play or simulation, the annotated roadmap and corridors are used in two ways. First, they are used to plan the path for the leader of the fire team. Second, they are used to calculate and propagate newly acquired tactical information. As depicted in Figure 3, the following steps can and should be executed during real-time use:

1. If new tactical information is received/uncovered this tactical information is annotated in the correct position on the roadmap and in the corresponding corridors. For this, some line of sight (LoS) calculations need to performed. Due to the availability of the corridors we are capable of performing these LoS calculations very efficiently (Section 2.1.).
2. Given a start position of the leader of the fire team, and a goal position, the roadmap is searched for a short and safe route.
3. Then, the path from the roadmap is extended to a list of corridors that need to be traversed. Some processing on this list of corridors might be required.

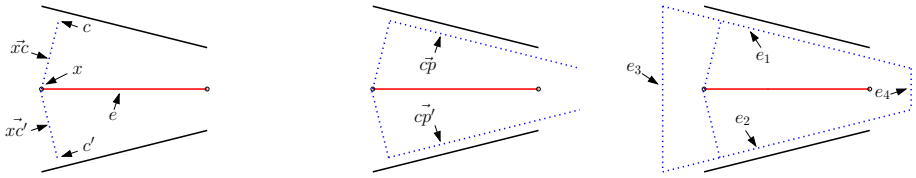


Figure 4: The step-by-step construction of corridors

4. Starting in the first corridor, the leader will search for a locally optimal point to move to. This is repeated, going from corridor to corridor, until the goal position is reached (Section 2.3.).
5. During the execution of the path of the leader, the rest of the fire teams assumes position behind the leader, in the best available formation, keeping the correct fire sector covered (Section 2.4.).

2.1. Corridors

As a pre-processing step we need to extend the roadmap to a set or collection of corridors. The shape of our corridors is an isosceles trapezoid [18]. The line through the middles of the two parallel sides is called the spine of the corridor. For every edge e in the roadmap we construct two corridors, one for every node connected to edge e. The construction of the corridor of node n connected to edge e is done in the following manner:

1. Align the spine of the corridor with edge e, and place the spine on n.
2. Find the closest point cp on the obstacles in the environment to node n.
3. Place one side of the trapezoid (a non-parallel one) perpendicular to the direction of n to cp, through cp.
4. Mirror this side in the spine of the trapezoid, thus forming the other side of the corridor.
5. Place the parallel sides of the trapezoid on n and move them away from this node until they hit/intersect an obstacle in the environment.

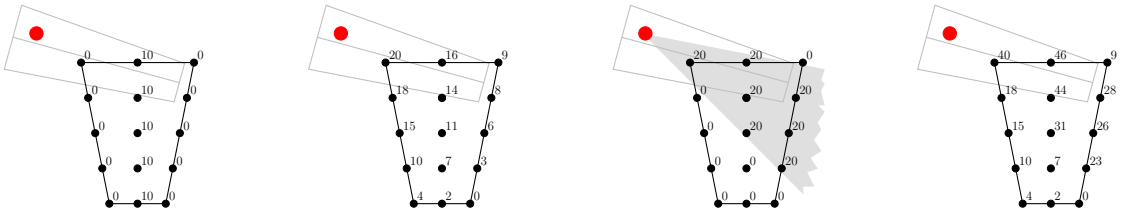
Figure 4 shows the construction of a corridor graphically. This process can fail, in the sense that the parallel edges can't move outward. This means that a corridor is created with zero area, which is an invalid corridor. These corridors are removed from the set of corridors.

Please note that every node in the graph can have at most the same number of corridors as it has edges connected to it. This results in at most $2|E|$ corridors, where E is the set of all edges in the roadmap.

2.2. Tactical Annotation

To be able to search the roadmap and generate paths inside the corridors both need to be annotated with tactical information. Let us first describe how the corridors are annotated, after which we will describe the roadmap annotation.

The annotation of the corridors is done by selecting the points on the sides and spine of the corridor. These points are chosen with a certain density. The higher the point density,



(a) The danger values based on the distance from the spine.

(b) The danger values based distance to a threat.

(c) The danger values based visibility/Line-of-Fire of the threat.

(d) The final danger values, which is the weighted sum of all values.

Figure 5: The danger values of a corridor. The point in the top-left corner (in a second corridor) is a threat.

the more accurate the information that can be stored. However, more points also require more resources, both in processing time and memory consumption. The value of each point is dependent on several criteria. The final value for a point on the corridor is the weighted sum of the danger levels determined by the following list of criteria (see Figure 5):

- If a point is closer to the spine, it will be further from any obstacle, meaning that the point is more exposed. More exposure means the danger level is higher.
- The distance from any known threat. The further a point is removed from a threat, the less danger there is on that point. Thus, this point will have a lower danger level.
- If a point is in the Line-of-Fire (or Line-of-Sight) from a known threat, the danger is is higher at that point, resulting in a higher danger level for that point.

Just as in general path planning technique that use roadmaps, the edges of the roadmap are assigned weights. Normally, these weights represent the traversal time of an edge. In our case, we also want to incorporate tactical information. Every edge corresponds to at most two corridors. We define the weight of the edge as the (weighted) sum of the edge length and the average danger value of the two corridors. The danger value of a corridor is the average value over all tactical points on the corridor.

Once a new threat is detected, or additional information is received, the danger values on the corridors are updated, followed by an update of the weights in the roadmap based on the new values in the corridors.

Please note that because we have the set of overlapping corridors we can speed up the Line-of-Sight/Line-of-Fire calculation compared to the normal techniques of doing line collisions. We can use the corridors, their layout and overlap in these calculations.

2.3. Path Planning and Corridor Traversal

When a path query is requested, the system searches for a path from the start to the goal (both given in the query) using the roadmap. The result from this is a list of nodes and edges that need to be traversed. This list is converted to a list or sequence of corridors (remember that every edge/node combination corresponds to a corridor). At the start, the leader of the fire team is situated in the first corridor of this list. The leader checks a number

of positions around its current position toward the intersections point of the current corridor and the next corridor in the list. For these candidate points a danger value is calculated interpolating the annotated values of the closest points from the character to the value points on the corridor. The candidate point with the lowest danger value is selected and the leader moves to that point. This process is repeated until the leader reaches the goal. Everytime the leader enters the next corridor in the list, this corridor is set as the current and the process continues.

2.4. Fire Teams Dynamics

The path planning and corridor traversal generates a sequence of position (a path) for the team leader or *point man* to follow. The other team members should try to stay in formation and follow the team leader.

To model the behavior of each individual team member we use an algorithm that, for each character, picks a number of candidate positions around that character and evaluate each of these positions according to a set of tactical criteria and selection rules [17]:

- Maintain place in formation (if available clearance allows it)
- Maintain separation between team members (if available clearance allows it)
- Use cover if available (hug the wall)
- Watch assigned fire sector (180/4 degrees when one side is blocked by wall or obstacles, 360/4 degrees in open space)
- Try to maintain Line-of-Sight (LoS) with at least one team member
- Keep out of Line-of-Fire (LoF) cone of team members.

For each character the best position is selected. This process is constantly repeated while the team leader moves along his/her path.

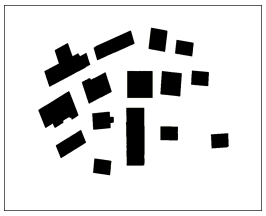
3. Experiments and Results

3.1. Implementation

We implemented the approach using Microsoft Visual Studio C++ running under Windows XP Pro (SP2) on a Pentium IV (2.8 Ghz) with 1 Gb of RAM. For the visualization and collision detection we used the Callisto library [11]. We also used the CGAL library[12] for the coverage calculations.

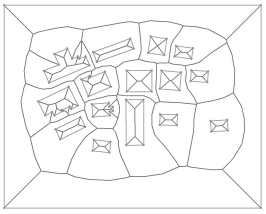
3.2. Test Environment

In all experiments we use a model of the McKenna MOUT Training site at Fort Benning, Georgia, USA. This site was constructed 20 years ago and consists of a number of wooden buildings (including a church). It was build to resemble a typical European village or city. This training site is still in heavy use by the United States Armed Forces. Figures 6(a) and 6(b) shows the layout of the city, and a photograph taken of the real site.

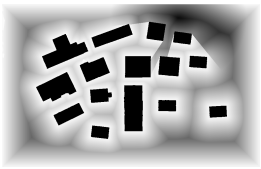


(a) The layout of the McKenna MOUT training site at Fort Benning, Georgia, USA.

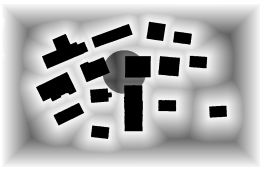
(b) A photograph of the McKenna MOUT training site at Fort Benning, Georgia, USA.



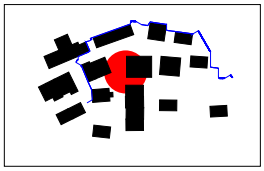
(c) The roadmap used as input for our approach.



(d) A typical dangermap.



(e) The resulting dangermap calculated using our approach.



(f) The resulting path calculated using our approach.

Figure 6: The scene, the roadmap and a resulting dangermap of the environment

area	**coverage percentage full roadmap**	**coverage percentage roadmap w/o tentacles**
full free space	99.9%	99.5%
tactically-handy free space area (non-convex parts)	92.0%	56.7%

Table 1: The coverage percentages for the full free space and the tactically-handy areas of free space.

In all experiments we used the roadmap that was generated using the voronoi diagram or medial axes of the environment (see Figure 6(c)). In one case, the first experiment, we used this roadmap and removed all the tentacles from it, resulting in only the skeleton of the medial axes. The roadmap generated from the medial axes is in general a high-quality roadmap, it is both smooth and has a large clearance. It also captures the connectivity of free space very well.

3.3. Experiments

To show the basic effectiveness and give a proof of concept of our approach, we have performed a number of preliminary experiments.

The first experiment was to see how well our approach covers the free space. From the roadmap (both the full and the reduced roadmap) we generated the corridors and determined the amount of coverage of free space generated by the corridors. Table 1 shows the results. It is clear that our method covers large amounts of free space, even with the reduced roadmap. However, when looking at very useful areas in the environment, the non-convex parts of the free space (e.g. corners between buildings), we see that the coverage is good when using the full roadmap, but drops drastically with the reduced roadmap. We can conclude that in order to get good coverage, the methods needs a roadmap with paths leading into the corners in the environment.

The second experiment was designed to show how tactical information is calculated and propagated over the corridors, and how these are used for the final path of the leader. For this we placed a threat in the top-right corner of the environment between the buildings. On every location on in the environment we determined the threat-level using our approach. The resulting dangermap is depicted in Figure 6(d), where the light-color indicates low danger levels and dark indicate high danger levels. Clearly visible are the lines-of-sight/lines-of-fire of the threat in the center. The dangermap also shows that open areas are more dangerous than area close to buildings.

Figures 6(f) and 6(f) show a danger map and the resulting path. The path clearly avoids highly dangerous areas. Movieclips of this and other paths can be found at `http://www.cs.uu.nl/people/arnok/tactical`.

4. Conclusions and Future Work

Tactical path finding depends highly on knowing the free space of the world, and being to evaluate the danger on every position in the free space. The approach presented in this paper succeeds in doing this. By efficiently covering the free space using simple geometric structure, i.e. nearly 100% coverage, and by annotating these structures with tactical information our method is able to find paths for the leader that are tactically sound. Using the same datastructure, the method is able to update tactical information rapidly and accurately. This makes the method applicable for real time games and simulations.

We are currently looking into extending and/or adapting the technique in order to cope with larger groups. This includes adapting the geometric information to suit the larger groups, as well as performing artificial intelligence goal-oriented planning for (sub)tasks in the group. Besides this, there are some open problems where we are currently working on. One open problem is that, in some cases, the fire teams should split up and perform a so-called *bounding overwatch* procedure, where two members of the team cross an open space, while the two remaining members provide cover. Another open issue is that the list of corridors needs to be optimized correctly. In some cases, corridors are redundant and should be removed from the list. However, this should be done with great care to prevent strange erroneous paths for the leader.

Acknowledgement

The authors would like to thank Dennis Nieuwenhuisen for the development of the collision detection and visualization library Callisto.

The research was partly supported by the Dutch Organization for Scientific Research (N.W.O.). The research was also supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2001-39250 (MOVIE - Motion Planning in Virtual Environments). Also, part of this research has been funded by the Dutch BSIK/BRICKS project.

References

[1] O.B. Bayazit, J.-M. Lien, and N.M. Amato. Better flocking behaviors using rule-based roadmaps. In *Algorithmic Foundations of Robotics V, Springer Tracts in Advanced Robotics 7*, pages 95–111. Springer-Verlag Berlin Heidelberg, 2004.

[2] Arjen Beij and William van der Sterren. Killzones ai : Dynamic procedural combat tactics. In *Proceedings of the Game Developers Conference*. CMPEvents, 2005.

[3] Roland Geraerts and Mark H. Overmars. Clearance based path optimization for motion planning. In *International Conference on Robotics and Automation (ICRA)*. IEEE Press, San Diego, CA, 2004.

[4] GlobalSecurity.org. Army field manuals. `http://www.globalsecurity.org/military/library/policy/army/fm/`, 2005.

[5] Washington DC. Headquarters, Department of the Army. Fm 90-10, military operations on urbanized terrain (mout), 1979.

[6] Washington DC. Headquarters, Department of the Army. Fm 3-06, urban operations, 2003.

[7] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *International Conference on Robotics and Automation (ICRA)*, pages 2138–2139. IEEE Press, San Diego, CA, 1994.

[8] L. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:556–580, 1996.

[9] Tsai-Yen Li and Hsu-Chi Chou. Motion planning for a crowd of robots. In *International Conference on Robotics and Automation (ICRA)*. IEEE Press, San Diego, CA, 2003.

[10] S. Raupp Musse and D.Thalmann. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):152–164, 2001.

[11] Dennis Nieuwenhuisen. Callisto library for visualization and collision detection. `http://www.cs.uu.nl/people/dennis/callisto/callisto.html`, 2005.

[12] The CGAL Project. Computational geometry algorithms library. `http://www.cgal.org`, 2005.

[13] C.W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[14] C.W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, 1999.

[15] G. Sanchez and J.C. Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2112–2119, 2002.

[16] B. Ulicny and D. Thalmann. Crowd simulation for interactive virtual environments and vrtraining systems. In *Eurographics Workshop on Animation and Simulation*, pages 163–170. Springer-Verlag, 2001.

[17] William van der Sterren. *Squad tactics: Team AI and emergent maneuvers*, pages 233–246. AI Game Programming Wisdom. Charles River Media, Inc., Hingham, Massachusetts, USA, 2002.

[18] Eric W. Weisstein. Isosceles trapezoid. mathworld – a wolfram web resource. `http://mathworld.wolfram.com/IsoscelesTrapezoid.html`.

[19] Wikipedia. Suspension of disbelief. `http://en.wikipedia.org/wiki/Suspension_of_disbelief`.