Roadmap-Based Level Clearing of Buildings

Samuel Rodriguez and Nancy M. Amato*

Parasol Lab, Dept. Computer Science and Engineering, Texas A&M University
sor8786@neo.tamu.edu,amato@tamu.edu

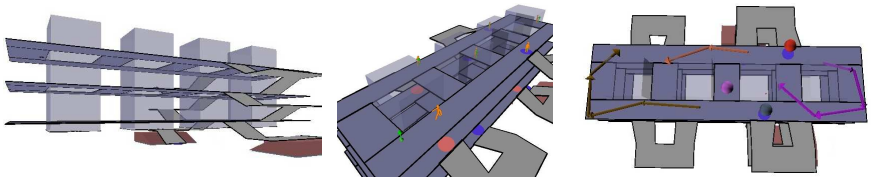
Abstract. In this paper we describe a roadmap-based approach for a multi-agent search strategy to clear a building or multi-story environment. This approach utilizes an encoding of the environment in the form of a graph (roadmap) that is used to encode feasible paths through the environment. The roadmap is partitioned into regions, e.g., one per level, and we design region-based search strategies to cover and clear the environment. We can provide certain guarantees within this roadmap-based framework on coverage and the number of agents needed. Our approach can handle complex and realistic environments where many approaches are restricted to simple 2D environments.

1 Introduction

Searching an environment is a problem that has been extensively studied and has application in games, virtual reality, computational geometry, and robotics. In this problem, the searching agents have requirements of either clearing an environment or searching as much or as quickly as possible. This can be a highly cooperative scenario which may involve many agents working together to achieve their goal. Target scenarios include searching for adversarial agents, guarding an environment and search and rescue to find people in distress or discover dangerous areas. The target environments of interest often include simple one level environments, but in games and virtual reality, environments often include terrains and multi-level structures. Our approach is not only applicable to 2D planar environments but can be used for multiple level environments.

The encoding of this problem has been done in a number of ways. Many approaches simplify the environmental representation to be able to provide more complete solutions. For example, graph-based approaches to the pursuit-evasion game often limit the agents to moving on the nodes and edges [19]. Geometric approaches limit the types of environments that can be handled as they are often 2D, polygonal environments [9]. Visibility restrictions on agents are one of the main limiting factors [10, 8]. While providing a means for developing a more complete solution, these assumptions limit the scope of the problem and consequently the applicability of the techniques to more complex scenarios. Extending to scenarios where 3D visibility is considered is essential for handling more complex problems.

* This research supported in part by NSF awards CRI-0551685, CCF-0833199, CCF-0830753, IIS-096053, IIS-0917266 by THECB NHARP award 000512-0097-2009, by Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).



(a) Building Example (b) Coverage Locations (c) Clearing Paths

Fig. 1. (a) A building example with four levels and multiple passages between each level. (b) Agents clearing a level by using coverage locations. (c) Clearing paths generated for top-level given agent visibility restrictions.

In our previous work [21, 20], we proposed our overall approach to pursuit-evasion in complex 3D environments which included basic heuristic search, pursuit and evasion behaviors. In [20], we also presented a tracking algorithm which was applicable to 3D structured environments that would allow agents to track other agents that have left an agent's view.

In this paper, we develop specialized search strategies for building environments with multiple levels. Within our search strategies we partition the environment into regions (levels) and apply ideas which have been applied in simple 2D environments to each region. In this way we can handle complex 3D environments and offer some guarantees given the underlying roadmap. This approach is also tunable so that the searching agents can function with varying levels of intelligence. As will be shown in the results, the quality of the underlying roadmap and agent capabilities can affect how well the agents clear/cover each level. Also, depending on the difficulty a game designer might desire a scenario to be, our approach can give insight into the coverage that can be attained given the number of agents available.

2 Related Work

In this section, we describe work that is relevant to the problem that we consider. This problem is similar to the art gallery problem and the pursuit evasion problem. Our extension to 3D building-like structures differentiates our approach from others that have been proposed.

An overview of the art gallery problem is presented in [18] which includes solutions, bounds and variants. In the most basic form of this problem, the goal is to cover a simple polygon with as few guards as possible. The polygon is considered covered if every point in the polygon lies within the visibility range of one of the guards. A discussion of the 3D version of the same problem (guarding a polyhedron) is included. An experimental approach to finding guard locations is presented in [2] where the goal is to heuristically find guard locations. A candidate set of potential guard locations are generated in a number of ways. The next step is the selection of guard locations from the set of all guard locations. In our approach, we use a similar greedy strategy for selecting guard locations from the selections of nodes at the current level being searched. Our approach also has to account for pathways between levels to guarantee clearing.

A great deal of work has been done on the pursuit-evasion problem. A classical, purely graph-based approach [19] considers the pursuit-evasion problem

on a graph in which agents can only move on edges between nodes and where the pursuer can capture the evader by occupying the same graph node. Another classic approach for polygonal environments is described in [9]. There has also been work in finding exact bounds on the number of pursuers needed in 2D environments [17]. A frontier approach is described in [23] based on exploring undiscovered portions of the environment.

Roadmap-based pursuit-evasion is described in [11] where the pursuer and evader both share a roadmap, navigate solely on the graph, and play different versions of the pursuit-evasion game. The versions of these games include determining if the pursuer can eventually collide with the evader and if collision can occur within some predefined amount of time.

In [3, 4], the benefits of integrating roadmap-based path planning techniques with flocking techniques were explored. A variety of group behaviors, including exploring and covering, were simulated utilizing a roadmap. This work was done in strictly 2D environments.

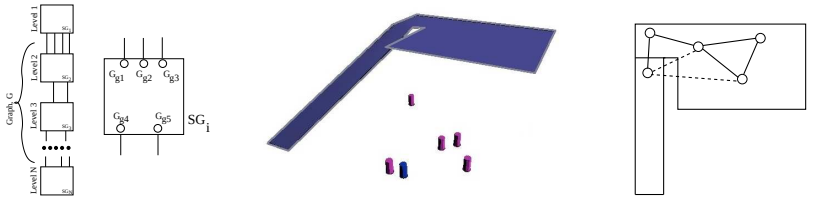
Our roadmap-based approach to 3D pursuit-evasion problems was first proposed in [21]. The work focused on different types of 3D problems that could be studied using a roadmap-based framework which included terrains, multi-level environments, in crowds of agents and for actual robots. The work did not include in depth analysis on guaranteed searching behaviors or requirements necessary to find a target in an environment.

Many forms of the pursuit-evasion problem have been considered. A graph-based approach to the pursuit-evasion problem is given in [16] where agents use either blocking or sweeping actions by teams of robots to detect all intruders in unknown 2D environments. One form of the problem [14] involves a team of agents with the goal of maximizing the average number of targets that are observed by at least one of the team members. Large teams of robots have been considered to detect intruders with communication allowed between agents within some range and with no map of the environment [15]. Other forms of collaboration between searching agents have been considered where agents utilize frontier information [6] with communication to switch between roles [7].

An agent's visibility plays an important role in what the agents detect in the environment and what kinds of environments can be handled. The problem of restricting or limiting the amount of visibility is considered in [10] with the field of view being variable in [8]. In [5] a team of agents with limited sensory abilities attempt to capture a single evading agent using sweep formations. The problem of pursuit-evasion on height maps, which restrict agent visibility is described in [13].

3 Problem Definition

The search strategy presented here is for clearing a building-like environment which consists of multiple levels with stairwells between levels. The goal is to develop search strategies that can guarantee a clearing of the environment. A geometric covering of the environment would ensure that when agents are stationed at guard locations, the environment would be completely covered (i.e. each point defined by the surface at that level would lie within an agent's view radius). A clearing of the environment is similar to coverage, however, agents that are clearing only have to guarantee that portions of the environment that have been cleared do not become recontaminated.



(a) Abstract Problem (b) Simple 3D Surface (c) Projected Surface

Fig. 2. (a) An abstraction of the problem is given showing a complete graph, G, with the subgraphs labeled $SG_{1..N}$. Each level in our building construction can be decomposed to subgraphs, with exits defining connections between other levels. (b) An example 3D surface with a long ramp leading up to the main level. (c) The surface projected to 2D. Also shown are sample configurations with valid and invalid connections.

While we do not guarantee that the geometric model of the environment is covered, we provide guarantees within our roadmap-based framework. We can guarantee that, given a set of agents, a certain percentage of the roadmap (which represents the environment) is covered at each level. We can also provide the maximum number of agents needed to cover or clear each level of the environment. The overall behavior is a clearing of the environment from one end of the environment to the other (bottom-up or top-down).

3.1 Environment Encoding

The building-like environments studied here are composed of levels, stairwells and exits. Each level represents a floor which the agents should try to cover. Adjacent levels are connected by stairwells. Input to the problem includes connections between levels and stairwells. Exits for a level are where stairwells connect to the level. We use the exits to define areas of the graph that must be guarded in order to guarantee a clearing. Graph algorithms could be used to find these portions of the graph automatically. A map of this type of environment has a structure like the graph shown in Figure 2(a). The subgraphs are shown, labeled SG_i, which represent valid movements at level i.

4 Overview of System

Our roadmap-based approach allows us to handle many classical problems along with many that have not been considered in the literature. This includes agents moving in 3D environments such as on a terrain, in multi-level environments or in areas with crowds. Here we describe our overall system and approach to this problem.

4.1 Approach

Overall, our approach uses a real-time simulation which consists of the movement of agents (utilizing the roadmap), complex pursuit/evasion/search strategies, and interesting agent interactions with both the environment and other agents.

We have designed and implemented a simulation infrastructure for storing and manipulating agents, behaviors, the environment, groupings, and relationships between groups. The focus of this paper is on a search strategies within our framework.

Agents have the ability to have different parameters. This affects their knowledge of the environment, behavior, movement through the environment, and interactions. In the search strategy we consider here, the main parameter affecting the result is the agent's view radius. This parameter determines the distance at which points in the environments are visible.

The behaviors of agents determine how they react throughout the simulation. In our framework, agents have either individual or group-based behaviors. While an agent performing an individual behavior, operates mainly independently, a group behavior often deals with more complex coordination. The level clearing behavior that is described in this paper is a group-based behavior requiring coordination where a leader agent computes guard and coverage locations and other agents in the same search group use those goals when finding paths.

Visibility is mainly affected by two factors. The first is obstacles and surfaces which can block visibility in the environment. The second parameter, mentioned earlier, is an agent's view radius. For simplicity, we assume an agent has a full circular view angle although in our simulation we have the ability to restrict this angle. We also have the ability in our simulation to have visibility be further restricted by considering that agents may block one another's view in the environment.

4.2 Roadmaps

The roadmap is an encoding of the environment that gives agents insight into potential paths through the environment. The complex environment is reduced to this abstract representation through a two stage process. The first stage is a series of sampling valid configurations in the environment which is followed by a connection phase where samples are connected and valid edges are kept if they do not collide with the environment. Standard sampling techniques [12, 1, 22] can be used in the portion of the environment dictated by the plane and obstacles in the plane.

When agents also operate on surfaces, this sampling process consists of randomly selecting configurations that are on the surface, projected to the 2D plane, and then obtaining the height component by finding where the projected point belongs on the surface. On a surface, valid connections are made between samples on the same surface which have a straight line connecting them which is completely inside the projected surface. This allows agents to navigate on the surface. An example surface and its projection to a 2D surface is shown in Figure 2(b,c).

5 Level Clear Algorithms

Our level clearing algorithm is an intuitive approach to handle complex three-dimensional environments within our roadmap-based approach. It can be described as a process where agents iteratively attempt to clear adjacent levels of the environment until reaching the last level. This clearing process is done

by guarding the stairwells between levels, which prevents recontamination from portions of the environment that have not been cleared. The first step is to generate a roadmap based on the environment. The roadmap is then partitioned into sub-graphs, as in Figure 2(a), based on the input configuration of the environment and exits and stairwells that exist. The partitioning is something that could be automated with more advanced graph-based techniques; we leave this for future work. The agents then search the environment from one level to the next until each level has been cleared. We describe two strategies for clearing of each level. The first strategy involves selecting coverage locations from the roadmap with the goal of covering the entire roadmap at a given level. The second strategy involves building clearing paths between portions of the roadmap that are cleared to uncleared areas until the roadmap (at that level) is fully cleared. The agents plan their paths to goal locations dictated by each strategy and traverse their path reaching the final goal. Once all agents are at the goal locations, the environment at that level is considered covered (to some extent) and the process can continue at the next level.

5.1 Strategy 1: Guard and Coverage Locations

The process of generating coverage locations for a given level, i, is shown in Algorithm 1. In this process, $nodes_i$ in the input roadmap at level i are first selected. Each exit at level i is considered a guard location G_{g_i}. These exits are located near stairwells and would prevent a target agent from entering or leaving the current level without being detected. These locations are shown in Figure 2(a) and correspond to input and output of the underlying subgraph at level i, SG_i. In our current problem formulation, these exit locations are input configurations.

The remaining problem is to cover level i as much as possible with the given agents. We achieve this using a greedy algorithm based on the roadmap nodes that were generated at level i. The nodes, $nodes_i$, which are covered by guard locations G_{g_i} are marked as cleared. Coverage locations, G_{c_i}, for level i are generated in a greedy manner such that the next node added to G_{c_i} is the node that will maximize coverage among all the remaining uncleared nodes in $nodes_i$. This is a process similar to what was presented in [2]. The number of agents needed to guarantee a covering of level i is $G_{g_i} + G_{c_i}$.

Overall, the maximum number of agents needed to guarantee a clearing of an environment is given by: $\max_{i \in env} \left\{ G_{g_i} + G_{c_i} \right\}$ where i represents each level to clear and G_{g_i} are guard locations at each level and G_{c_i} are the coverage locations selected.

An interesting aspect to this solution is that the ordering of the returned coverage locations affects the amount of coverage. Often, the total number of searching agents needed may be much more than the number of agents available. In these cases, we can report the portion of each environment that would be covered based on the coverage value returned by the number of agents available. These agents would also attempt to maximize coverage by going to coverage locations in the order that they were selected based on the greedy algorithm presented in Algorithm 1.

Algorithm 1 Generate Coverage Locations

Input: environment env, roadmap $rdmp$, current level searching i

$nodes_i \leftarrow rdmp$.nodesAtLevel(i)
$G_{g_i} \leftarrow env$.ExitsAtLevel(i)
$markNodesCleared(nodes_i, G_{g_i})$
$G_{c_i} \leftarrow \emptyset$
while existNodesUncleared($nodes_i$) **do**
  $N_j \leftarrow$ uncleared node $\in nodes_i$ with max coverage
  $G_{c_i} += N_j$
  markNodesCleared($nodes_i, N_j$)
end while
return { $G_{g_i} + G_{c_i}$ }

5.2 Strategy 2: Guard Locations and Clearing Paths

The basic idea of this strategy is similiar to Strategy 1. However, when using clearing paths, agents can clear entire areas using the underlying encoding of the roadmap and cooperation between other agents. As in Algorithm 1, guard locations are generated initially and the nodes covered are marked as clear. The main difference is that instead of selecting the node that maximizes the coverage, the clearing path selected is one that maximizes the path traveled through the level as a way of maximizing the area covered by the next clearing path added. Due to space limitations, we only give an overview of how clearing paths are generated in our framework.

Clearing paths are started on the frontier of the area that is already covered (i.e. from nodes that are not already cleared). A clearing path, CP, keeps track of the nodes that are initially covered, N_{init}, and nodes that are currently covered, N_{cur}. As a clearing path is built up we consider the set of potential moves to be the nodes that are outgoing from N_{cur} which are not in N_{cur} or N_{init}. A move is considered valid if moving to that node leaves the nodes leading to the previous outgoing nodes covered. In this way we can ensure clearing paths are built up which prevent their N_{cur} from becoming contaminated. We build the paths up from all frontier nodes and continue until no more valid moves exist. We use a greedy strategy to select the path that maximizes the area covered. We continue to build clearing paths until all nodes on a level are cleared.

5.3 Pathway Assignments

The process of pathway assignment is critical when attempting to guarantee that contamination of a covered region does not occur. As a simple example, a guard agent at a guard location on level i whose group of agents is progressing to level $i+1$ might naturally be the guard of the exit associated with the stairwell connecting level i and $i+1$. This may not happen if an agent is present whose pathway is shorter to the guard location at level $i+1$. In the assignment process, all agents have pathways generated from their current location to one of the goal locations generated for the next level. The goal locations are then assigned by the leader agent based on the path distance. This is an iterative process where the shortest path is selected from all the pathways. Once an agent and goal location pair has been selected, its associated pathways are removed from the

full pathway list as are the pathways that are associated with the goal location assigned. In this way, we guaranteed that an agent guarding a stairwell at level i leading to another level $i+1$ is selected to guard the exit at level $i+1$.

5.4 Dependence on Mapping

The main aspects of this approach, generating coverage locations, generating clearing paths, and pathway assignments, have an inherent dependence on the roadmap. When attempting a full coverage, we assume that the roadmap gives a good coverage of the environment. One way we attempt to ensure this is by preventing samples or configurations from being generated too close to one another. In this way, the nodes representing the free space in the environment are spread out. The assignment of pathways also assumes that valid pathways exist between guard locations, through stairwells, between levels. We ensure that the roadmap is generated in such a way that these pathways do exist although it is a known limitation to our approach. Due to the inherent dependence on the roadmap, our searching agents can be made more or less intelligent based on the quality of the mapping. This will be shown in the results. Agents that have a better mapping will have a better chance of fully clearing an environment whereas agents with a simpler mapping may miss portions of the environment. This may in fact be useful in games when trying to have varying capabilities of adversarial agents.

6 Experimental Results

We present results in two simulated building-like environments, shown in Figures 1(a) and 3. Each of these environments consists of multiple levels with stairwells connecting adjacent levels. In these problems, stairwells do not exist between non-adjacent levels. We present results in the form of the average number of agents that would be needed to cover each level of the environment given the visibility constraints by varying the view radius (VR) distances. The number of agents needed to clear the environment would be the maximum needed over each of the individual floors (which is based on the roadmaps generated). Results are generated using roadmaps that adequately cover the environment, which includes valid pathways between open areas, hallways and exits. The effect of agent view radius restrictions is not taken into account during the roadmap construction phase. Results presented are averaged over ten trial runs, unless specified otherwise. Animations and additional results can be found on our webpage: http://parasol.tamu.edu/groups/amatogroup/research/flock/.

6.1 4 Floor, 5 Hallways

This environment, shown in Figure 1(a), consists of four levels. The first level consists of an open area with stairwells leading to the second level. The second level and each subsequent level is divided by four obstacles, creating five hallways off the main two hallways. The upper levels are connected by two stairways between the levels. The environment dimensions are 230 units $\times$140 units. We tested our search algorithm with a number of agent view radius capabilities (10, 30, 60, 120 and 240 units); results are shown in Table 1. Results are omitted

Table 1. Search results showing number of agents needed in 4 Floor, 5 Hallway environment.

	Strategy 1 Coverage Locations				Strategy 2 Clearing Paths			
Level	1	2	3	4	1	2	3	4
VR: 10 - Avg. number of searchers needed to clear	41	50	49	50	23	30	31	27
VR: 30 - Avg. number of searchers needed to clear	16	11	12	12	12	7	7	6
VR: 60 - Avg. number of searchers needed to clear	8	8	8	6	5	8	7	5
VR: 120 - Avg. number of searchers needed to clear	4	8	8	6	4	7	7	4

for view radius capabilities of 240 units since they match those for agents with a view radius of 120 units. The maximum number of searchers needed to clear each level is shown.

As the view radius increases, the number of agents needed to cover each level decreases. In fact, once the view radius gets large enough, the average number of agents necessary to cover an environment levels off. In this environment, this happened when the view radius was at 120 and 240 units, requiring the same number of agents at each level. For a view radius of 60 units and higher, the upper levels all require the same number of searchers. This corresponds to guard agents blocking off the exits and hallways and agents required for guarding the hallways.

The coverage of the environment can be analyzed as coverage locations are added. The coverage gradually increases until complete coverage is achieved. Given N agents available to cover and clear an environment, the first N coverage locations would be used as goals to maximize coverage. If coverage were only required to a certain level, our approach could give insight into the number of agents necessary to achieve this coverage. For example, if only 98 percent coverage were required then only 6 agents would be needed as goal locations 7 and 8 would be unnecessary for the first three levels.

The number of agents needed when generating clearing paths in this environment is much fewer than are needed when trying to cover each level with agents. This is because the agents can build off the areas that are previous covered to expand the area they clear while preventing contamination of that region. In fact, when agents have smaller view radius values using the clearing paths strategy requires far fewer agents to clear the levels.

6.2 3 Floor Office Building

The second environment we performed experiments in is shown in Figure 3. This environment is much more complex than the previous one. It consists of three levels, with many rooms on each level with walls obstructing the view between rooms and hallways. There are three stairwells between the first and second floor and two between the second and third floor. This environment has dimensions of 355 units $\times$ 275 units. We tested our search algorithms with a number of agent view radius capabilities (30, 60, and 300 units); results are shown in Table 2. The first floor has 900 samples to cover the inside and outer portions of the building while floor 2 and 3 have 250 samples to cover. This minimal sampling is good

enough to have expected pathways through the environment. As before, for each view radius tested, the average number of searchers needed per level is shown.

When agents are equipped with a view radius of 300 units they have the ability to see across almost the entire environment. This results in much fewer searchers needed to guarantee a clearing of the roadmap. The first level generally requires more searchers because the open space around the building is also considered area to cover in the problem. When agents have a view radius of only 30 units, the number of searchers needed to cover the roadmap is nearly twice the number needed when the view radius is 300 units.

The number of agents needed to cover the roadmap with a view radius of 60 units is similar to when they have a view radius of 300. Although we do not have the ability to compute the optimal geometric coverage locations, we look at the number of locations needed to guarantee coverage on the third floor of this environment. By simple inspection, the number of coverage locations needed is shown in Figure 3(a) when an infinite view radius is used. These locations correspond to two searchers for the main four hallways (Locations 1 and 2), one for each main room (Locations 3–23) and two to guard the exits (Locations 24 and 25). These locations would guarantee a geometric coverage of the third level (with guarding of the exits). With a view radius of 60 units, an example of locations that are generated is shown in Figure 3(b). For the most part, the locations shown correspond to those in Figure 3(a) although one room (corresponding to Location 19) is not guaranteed to be covered. The underlying roadmap was covered though.

Our approach results in fewer agents needed when the view radius is set to 300 units. This is because while the roadmap may be covered, the geometric representation of the environment is not guaranteed to be covered. An example of both good and missed coverage locations is shown in Figure 4. In this example, we generated 100,000 random samples on the third level of the environment. Samples that were not covered by the coverage locations are drawn in red. These locations are generally in the corner of rooms, however cases can occur where a room seems to be covered based on the roadmap nodes covering that room which may leave a large portion of the room uncovered (Figure 4(b)). If more complete coverage is desired, a more densely sampled map can be generated as in Figure 4(c).

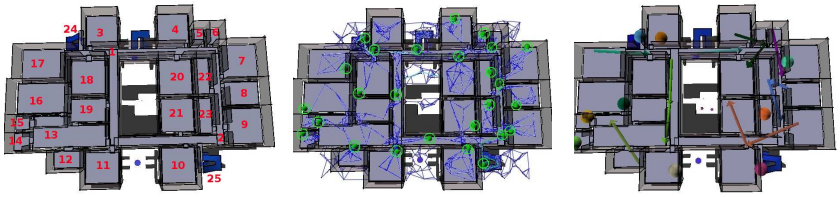
We are able to use clearing paths in this complex environment to clear each level. It can be seen in Table 2 that fewer agents are needed when using the clearing paths strategy. This is true at each level of the environment but especially when the agents have the more limited view radius (VR 30) where at the first level close to 30 fewer agents are needed to clear the level. An example of the clearing paths generated can be seen in Figure 3(c).

7 Conclusion

In this paper we describe a level search algorithm to clear a building. This problem is inspired by art gallery and pursuit-evasion problems. Our heuristic approach offers guarantees within our roadmap-based approach. This is an extension to existing approaches however we have the ability to solve problems in complex environments. These 3D environments have received less attention but are as important to solve as simple, single level environments. This approach is tunable to allow for different levels of difficulty to be supported.

Table 2. Search results in 3 Level Office Building

	Strategy 1 Coverage Locations			Strategy 2 Clearing Paths		
Level	1	2	3	1	2	3
VR: 30 - Avg. number of searchers needed to clear	68	41	41	39	27	26
VR: 60 - Avg. number of searchers needed to clear	39	26	26	23	16	16
VR: 300 - Avg. number of searchers needed to clear	34	22	22	19	13	13

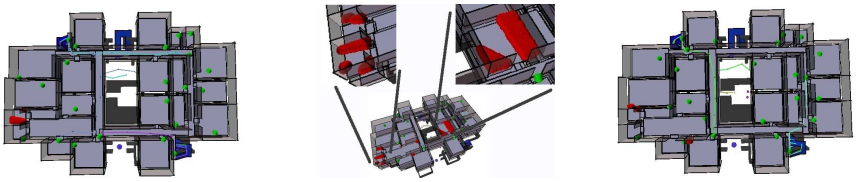


(a) Coverage Locations (By Simple Inspection) (b) Roadmap-Based Coverage Locations (VR 60) (c) Clearing Paths (VR 60)

Fig. 3. Office Example.

References

1. Amato, N.M., Bayazit, O.B., Dale, L.K., Jones, C.V., Vallejo, D.: OBPRM: An obstacle-based PRM for 3D workspaces. In: Robotics: The Algorithmic Perspective. pp. 155–168. A.K. Peters, Natick, MA (1998), proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998
2. Amit, Y., Mitchell, J.S.B., Packer, E.: Locating guards for visibility coverage of polygons. In: Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX) (2007)
3. Bayazit, O., Lien, J.M., Amato, N.M.: Better group behaviors in complex environments using global roadmaps. In: Artif. Life. pp. 362–370 (Dec 2002)
4. Bayazit, O., Lien, J., Amato, N.M.: Roadmap-based flocking for complex environments. In: Proc. Pacific Graphics. pp. 104–113 (Oct 2002)
5. Bopardikar, S., Bullo, F., Hespanha, J.: Cooperative pursuit with sensing limitations. In: American Control Conference (ACC). pp. 935–953 (July 2007)



(a) Good coverage with Basic Map (b) Missed Coverage (c) Good Coverage Oversampled Map

Fig. 4. Coverage examples of 3D office example. Goal locations shown in green. Missed samples (out of 100,000 samples) shown in red for third floor.

6. Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). pp. 476–481 (2000)
7. Durham, J.W., Franchi, A., Bullo, F.: Distributed pursuit-evasion with limited-visibility sensors via frontier-based exploration. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). pp. 3562–3568 (2010)
8. Gerkey, B.P., Thrun, S., Gordon, G.: Visibility-based pursuit-evasion with limited field of view. Int. J. Robot. Res. 25(4), 299–315 (2006)
9. Guibas, L.J., claude Latombe, J., Lavalle, S.M., Lin, D., Motwani, R.: Visibility-based pursuit-evasion in a polygonal environment. In: International Journal of Computational Geometry and Applications. pp. 17–30. Springer-Verlag (1997)
10. Isler, V., Kannan, S., Khanna, S.: Randomized pursuit-evasion with limited visibility. In: Proc. ACM-SIAM Symposium on Discrete Algorithms. pp. 1060–1069 (2004)
11. Isler, V., Sun, D., Sastry, S.: Roadmap based pursuit-evasion and collision avoidance. In: Proc. Robotics: Sci. Sys. (RSS) (2005)
12. Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Automat. 12(4), 566–580 (August 1996)
13. Kolling, A., Kleiner, A., Lewis, M., Sycara, K.: Solving pursuit-evasion problems on height maps. In: IEEE International Conference on Robotics and Automation (ICRA 2010) Workshop: Search and Pursuit/Evasion in the Physical World: Efficiency, Scalability, and Guarantees (2010)
14. Kolling, A., Carpin, S.: Cooperative observation of multiple moving targets: an algorithm and its formalization. Int. J. Rob. Res. 26, 935–953 (September 2007)
15. Kolling, A., Carpin, S.: Multi robot pursuit evasion without maps. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). pp. 3045–3051 (May 2010)
16. Kolling, A., Carpin, S.: Pursuit-evasion on trees by robot teams. Trans. Rob. 26(1), 32–47 (2010)
17. Lavalle, S.M., Lin, D., Guibas, L.J., claude Latombe, J., Motwani, R.: Finding an unpredictable target in a workspace with obstacles. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). pp. 737–742 (1997)
18. O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, New York (1987)
19. Parsons, T.D.: Pursuit-evasion in a graph. In: Theory and Applications of Graphs. Lecture Notes in Mathematics, vol. 642, pp. 426–441. Springer (1978)
20. Rodriguez, S., Denny, J., Mahadevan, A., Vu, J., Burgos, J., Zourntos, T., Amato, N.M.: Roadmap-based pursuit-evasion in 3d environments. Transactions on Edutainment (to appear) (2011)
21. Rodriguez, S., Denny, J., Zourntos, T., Amato, N.M.: Toward simulating realistic pursuit-evasion using a roadmap-based approach. In: Proc. of the 3rd Intern. Conf. on Motion in Games, 2010, Lecture Notes in Computer Science (LNCS). pp. 82–93 (November 2010)
22. Wilmarth, S.A., Amato, N.M., Stiller, P.F.: MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). vol. 2, pp. 1024–1031 (1999)
23. Yamauchi, B.: Frontier-based exploration using multiple robots. In: International Conference on Autonomous Agents (Agents '98). pp. 47–53 (1998)