Exhaustive Exploration Strategies for NPCs

Muntasir Chowdhury and Clark Verbrugge
School of Computer Science
McGill University
Montréal, Québec, Canada
muntasir.chowdhury@mail.mcgill.ca
clump@cs.mcgill.ca

ABSTRACT

Automated, exhaustive exploration of game levels is typically based on simple, greedy coverage heuristics, or is directed more at the problem of locating dynamic targets. In this paper, we present and analyze a method for creating an exploratory tour guaranteed to uncover all parts of a 2D map. First, a set of camera points that collectively ensure full coverage are chosen. A tour visiting these cameras is then constructed using navigation graphs and by employing heuristics based on distance and visual coverage. We identify the different factors that affect this methodology through experiments on maps from commercial games. The strategies proposed can be used by both hostile and non-hostile NPCs in different spatial search scenarios that benefit from full coverage, or to help a player uncover a map in *fog-of-war* settings.

Keywords

Artificial Intelligence, Exploration, Agent Behaviours

INTRODUCTION

Non-player characters (NPCs) are sometimes required to explore the game map, either as a means of ensuring dynamic enemies eventually and naturally encounter players, or contentually, in their role of guards in stealth games. Exhaustive designs are useful in simulating thorough search, and for ensuring trivial, never-observed spots do not exist. Automatic solutions to fully observing a space are also important to helping players, and may be found in the use of non-hostile NPCs that assist or guide players through *fog-of-war,* or as a non-trivial augmentation to player commands provided in order to reduce the tedium of manual, exhaustive exploration of large game levels. In complex, obstacle-rich and procedurally generated game environments, however, exploration strategies that guarantee efficient and complete coverage, and ideally without excessive repetition are not obvious.

In this work we explore a simple and novel approach to exhaustive exploration in full information (known), 2D game maps. We use geometric principles to first establish a set of sites that collectively ensure complete coverage, and then consider different heuristics for connecting these points, building routes that guarantee a character will eventually, deterministically observe the entire game map. Beyond reaching full coverage, important considerations in algorithmic approaches to exploration are in doing so quickly, and in maximizing the amount of map space revealed at each point—the latter is especially useful in mimicking a human interest in having continual, high novelty throughout exploration. Experimental

Proceedings of 1st International Joint Conference of DiGRA and FDG
7th Workshop on Procedural Content Generation

evaluation of our approach under a wide variety of configurations allows us to observe the impact of different heuristic choices on speed and the pace of map-reveal, and verify that our approach is also robust to different game maps, choices of starting and ending positions, means of navigation, and granularity in performing observations.

Specific contributions of our work include the following.

- We present and implement an approach to exhaustive exploration of 2D maps, suitable for complex, known game geometries. Our design is heuristic and modular in several ways, but is able to guarantee complete, deterministic coverage.
- We perform detailed experimental work for different parametrizations of our design, using 6 different maps taken from modern games to establish the degree of variance induced by different factors. In this we show that simple greedy approaches are effective, but that a more complex, but still feasible heuristic balancing path cost and coverage benefit has generally better performance.

In the next section we discuss related work on exploration, focusing on theoretical results from computational geometry, as well as more practical approaches in robotics and games. We then present our methodology and implementation framework, including the theoretical basis for our result and different, modular heuristics that can drive it. Experimental results show performance of our designs under different permutations of assumptions.

RELATED WORK

Our work relates in a theoretical sense to problems guaranteeing (visual) coverage of a polygon. Perhaps the most well known problem in this area, and which we also make use of in our design, is the *Art Gallery Problem* (AGP) which asks for the minimum number of cameras or stationary guards required to cover the interior of an n-walled art gallery, modelled as an n-sided 2D polygon (O'Rourke 1987). For dynamic coverage, the main variation of this is the *Watchmen Route Problem* (WRP), where the task is to find the smallest route in a polygon, such that every point inside the polygon is visible from some point on the route. This particular problem statement is very similar to ours, except that we are interested in reducing redundancy in the observations, as well as trying to balance performance and efficiency in a practical design.

It has been shown that WRP starting from a fixed point is NP-hard for polygons with holes (Chin and Ntafos 1986). The best known solution for this problem in the case of simple (no holes) polygons is an $O(n^3 \log n)$ algorithm (Dror et al. 2003). The WRP without any restrictions currently has an $O(n^5)$ solution (Tan 2001), and there exists a linear time 2-approximation algorithm as well (Tan 2007). There have also been variations of both AGP and WRP using limited visibility (Kazazakis and Argyros 2002; Wang et al. 2010). These theoretical approaches are interesting but quite complex, and as mentioned the problem is NP-hard in the context of polygons with holes, which are required to model many game maps.

Some exploration-related work in the games area can be found as part of part of more complex, overall strategies, such as in bot design for *Real-time Strategy* (RTS) games. Hagelbäck and Johansson (2008), for instance, describe a design for automated discovery and

combat based on potential fields. Attraction and repulsion forces are applied to terrain, bases, enemies, and unknown locations to guide a bot through terrain, avoiding obstacles and reaching appropriate locations for engaging enemies. A different approach was taken by *Third Eye Crime,* which used dynamically updated discrete probability distributions to guide enemy search in an interactive stealth game (Isla 2013). The distribution is then exposed to the player, who can use that information to avoid discovery. These works combine gameplay with exploration in interesting ways, but either do not guarantee exhaustive coverage or do so inefficiently, as a probabilistic limit.

In games of certain genres (e.g. roguelike, turn-based), player tedium is reduced in the presence of repetitive level exploration through automated exploration. This automation is usually player controlled and within a gridworld setting, acting as a macro-command that (repeatedly) moves the player character to the nearest discrete, unexplored location, stopping when enemies or other important elements are encountered. In the case of *Civilization 5* (Various contributors) automated scouting takes into account terrain cost and visibility benefit, and in the case of *Dungeon Crawl Stone Soup* (DCSS Devteam) there are heuristics that affect the choice of which unobserved area is chosen, but these are essentially all minor variations on a trivial, greedy approach. These simple approaches can also sometimes be quite inefficient, with unsatisfactory results a frequent concern for players, as is often reported in game forums (Various contributors).

The field of robotics contains perhaps the most mature and significant research on autonomous exploration. Many different approaches have been used, including ones based on constructing efficient coverage tours (Xu et al. 2014), although these tend to be strongly constrained by environmental or vehicular factors. Exploration approaches more generally drift into solutions to simultaneously mapping an unknown environment and figuring out a robot's position within that environment, or the *Simultaneous Localization and Mapping* (SLAM) problem (Thrun 2002). Certain strategies employ moving the agent in a fixed path such as concentric circles (Sim and Dudek 2003), while others use some sort of evaluation function to determine the next best point to move to (Tovey and Koenig 2003), and experimental analysis of different strategies that can be followed have been performed (Amigoni 2008). Unlike exploration in a video game context, these algorithms have to account for hardware issues, and often focus on problems created by faulty sensor readings or a sensor's information extraction capabilities.

METHODOLOGY

Our implementation framework consists of multiple phases, largely incorporated into the *Unity3D* game engine. For greater efficiency in analysis and additional flexibility, we also have the ability to export specific phases to external programs; a schematic of the complete workflow is shown in figure 1.

The process we follow can be roughly divided into 3 main phases. First we prepare *maps* to experiment on. Then we find a set of points which collectively provide complete visual *coverage* of the map, and establish methods of accurately measuring coverage. Lastly, we connect these cameras via navigation graphs or *roadmaps*, and construct different exploratory tours that visit the cameras, such that each point inside the map is guaranteed to be seen from some point on any such tour. The subsections below expand on these core steps.

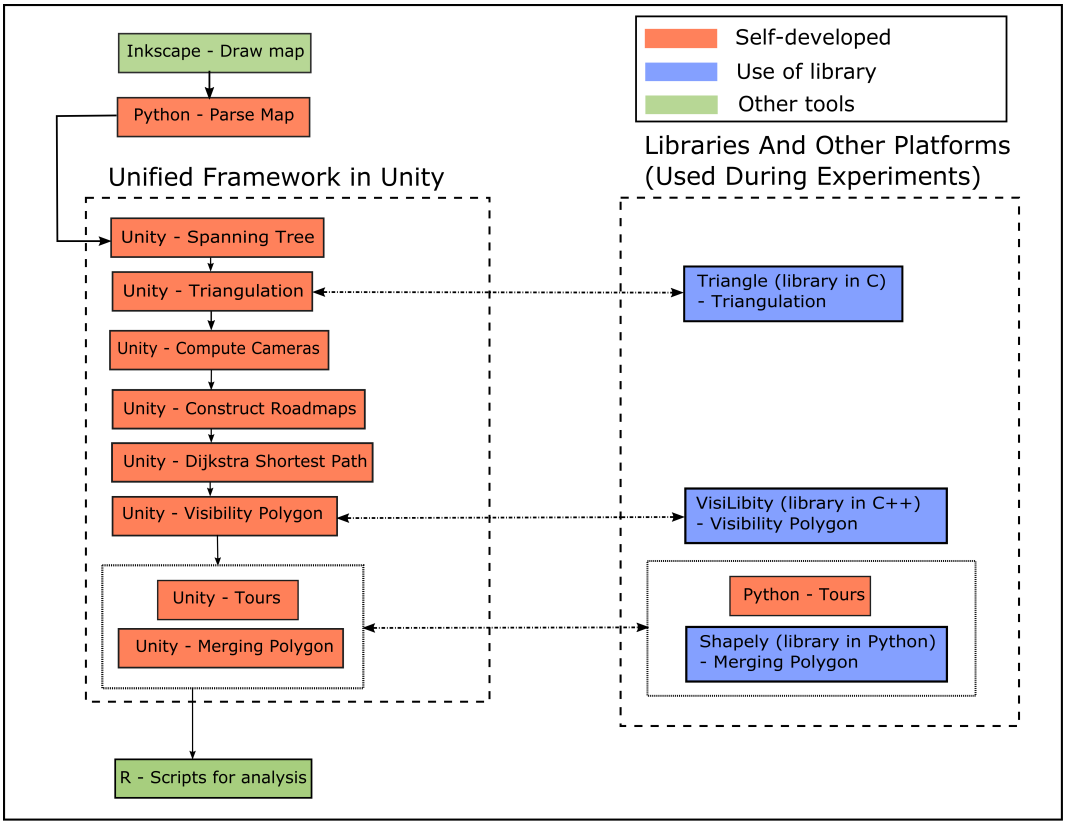


Figure 1: Platform architecture

Map Modelling and Collection

Since exploration is primarily affected by visibility and mobility, and these in turn are affected by structural properties, our essential requirement for modelling maps is the geometric shape of the game world. We use a 2-dimensional (2D) polygons with holes, where the obstacles and map border are represented by the holes and the boundaries of the polygon (exterior), respectively. The NPC is a point inside the polygon and can move about in every direction. This model is quite abstract, and can be applied to a large number of scenarios.

As games are proprietary products, most developers and publishers do not make their levels readily available for use outside the game. Map sets suitable for academic research can be found online (Sturtevant 2012), but are mainly aimed at evaluating pathfinding, and expressed as heavily discretized gridworlds. For more modern maps with scalable, geometric representations we thus perform a manual extraction based on tracing over map screenshots, easily found in various online sources. For 3D images we always trace along the top of structures as such a view usually obscures some floor-level details. Transformations are applied to isometric traces to correct any perspective view. The tracing is done in *Inkscape*, which saves the map in *.svg* format, which we can then parse in order to extract node and edge coordinates required to build a polygonal representation. This manual process does not of course necessarily result in perfect mimicry of the actual game map, but minor distortions do not affect our overall process or results, and the approximations inherent in this process could be improved through the use of direct, in-game representations, if available.

Coverage

As we are treating the NPC as a point, we can use point visibility to determine what it sees from a specific location. Given a polygon P (with holes) and a point p conceptually simple algorithms exist to compute the *visibility polygon*, or subset of P that can be seen from p (Ghosh 2007). If we can obtain a set of visibility polygons whose aggregate area covers all of the area within P, we can solve the problem of seeing the entire map. This more abstract problem was solved in the 1970's when posed as the *Art Gallery Problem*, which involves placing *cameras* in a 2D polygon, so as to guarantee complete observation of the space. A true minimum solution is expensive to compute, but solutions exist for simple polygons (O'Rourke 1987), and ones with holes (Hoffmann et al. 1991) which can guarantee sufficiency in camera placement.

Our process actually follows a slightly modified version of Fisk's (1978) solution to the simple polygon version of the Art Gallery Problem, as a computationally straightforward approach. This requires we first *triangulate* the polygon, then compute a 3-colouring of the triangulation vertices, designating the set of points with the minimum occurring colour as the set of cameras. Being convex, triangles can be fully observed from any of their vertices, a 3-colouring guarantees each triangle has a vertex of every colour, and so this guarantees every point in the polygon is covered. A slight inefficiency is introduced into this approach to handle holes while still guaranteeing 3-colourability, basically by connecting every hole to the exterior and so reducing our polygon with holes back to a simple polygon. Practically, this is done by forming a spanning tree whose nodes are the exterior and the holes.

Our exploratory tours will visit the cameras incrementally. To understand how well any such tour, or method used to generate the tour, works we have to determine the exact amount of space the NPC sees from its path. For this we use visibility polygons, which have the advantage of letting us compute relative coverage through polygon overlap and difference. The final NPCs exploration path, however, will be dynamic, in the form of edges, and therefore it would seem to make more sense to consider visibility from an edge instead of a point. We do not use edge visibility algorithms because they are highly complicated and are computationally expensive (Suri and O'Rourke 1986), and moreover do not let us consider the impact of frequency or granularity in NPC observations. We thus instead use point visibility from each endpoint of an edge as an approximation to edge visibility, computed using Asano's (1985) planar-sweep algorithm. To calculate aggregate coverage as an NPC moves from one camera to another we merge individual visibility polygons to form larger polygons.

Roadmaps and Tours

The process for visiting cameras involves navigating between individual camera locations. We use a roadmap construction for this, both for the improved efficiency and to follow the practical and common use of roadmaps in actual games. We experiment with two kinds of roadmaps that provide different pathing properties: triangulation roadmaps, and shortest path roadmaps. Either can serve as the basis for constructing routes between cameras by first joining start and end points to the roadmap, and then using a search process (A*, Dijsktra, *etc*) on the resulting graph to compute the final path.

Triangulation roadmaps are trivially computed from a triangulation, joining the centroid of each triangle to the midpoint of each of its edges that is shared with another triangle.

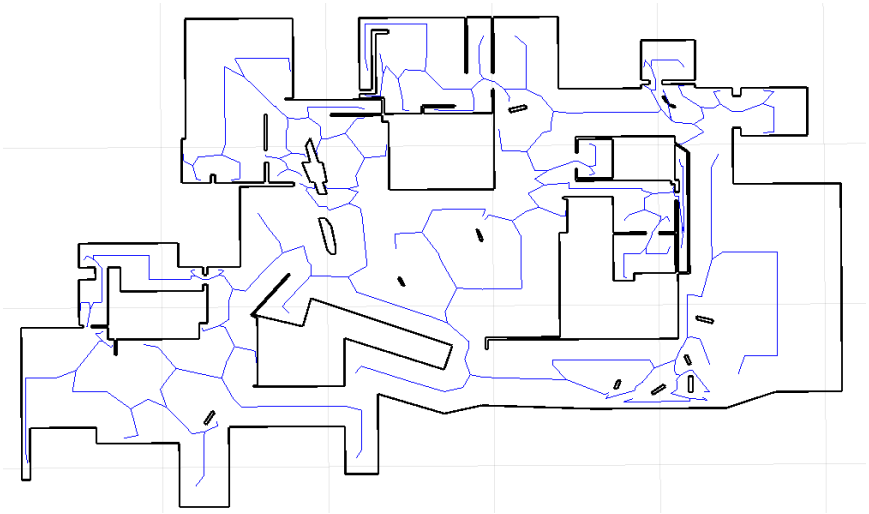


Figure 2: A triangulation roadmap on the map *Crash* from *Call of Duty 4*

Start and end points are then just connected to the centroid of their respective, enclosing triangles. The result is not optimal in general, with the potential for significant inefficiences due to variation in triangle size/proportions. It is, however, easy and efficient to compute, especially (as in our case) if a triangulation is already computed for other purposes, and tends to reasonably approximate a path that avoids close contact with obstacle edges, as is common in human-like movement. Figure 2 shows an example of the result.

Shortest-path roadmaps provide for more efficient pathing results. These roadmaps are built by finding a subset of polygon (and obstacle) vertices that are *reflex* vertices (interior angle strictly greater than 180°), and which can be connected by a *bitangent* edge, a line segment that does not strictly intersect the polygon along its length, or if slightly extended past the connecting points in either direction. Insertion of the start/end points is more complicated as well, requiring they be connected to all visible roadmap nodes, potentially including each other. LaValle (2006) gives the full construction. A shortest path roadmap, as the name suggests, guarantees shortest paths between points, but has the disadvantage of tending to include segments that overlap with or "hug" polygon edges, and so may not well represent realistic character movement. Figure 3 shows the result on the same game map as our triangulation roadmap example.

Preprocessing for Tours

Once the roadmap is computed, we connect each camera point to the roadmap as the possible start or end point of a path segment. We can begin from a given camera, compute a path to another camera, go to it via the computed path, and from there repeat the process until all cameras have been visited. This combination of paths is the NPCs exploratory tour, and by the use of the Art Gallery theorem, guarantees full coverage.

To judge a given tour we consider a) its total length, and b) the rate at which the map is seen or uncovered. The latter is required to figure out when the tour stops as it is complete when the whole map has been seen, regardless of whether all cameras have been visited. The length of the shortest path between two points in our roadmap is precomputed using Dijk-

Figure 3: A shortest-path roadmap

stra's single-source shortest-path algorithm, and we also precompute the visibility polygons of each node (camera and non-camera) on the roadmap. Figure 4 shows an example of a tour based on the shortest path roadmap.

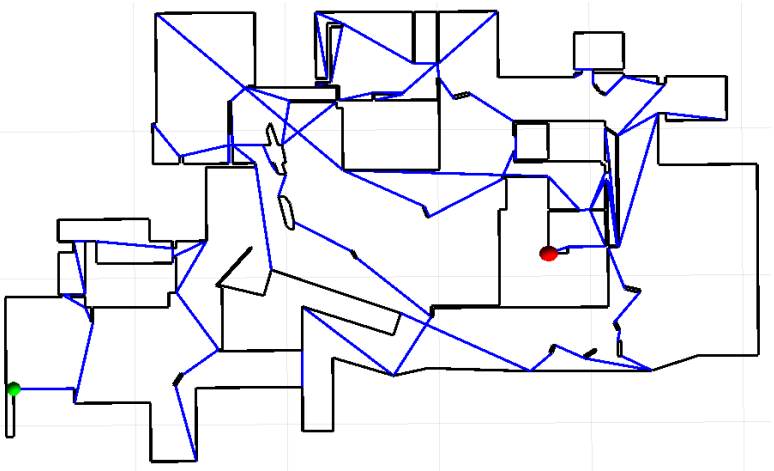


Figure 4: An exploratory tour covering the whole map i.e. the final result of our process. Blue lines indicate the tour, starting at the green node and terminating at the red node. Some edges in the image are traversed in both directions making it appear as though the path ends at several points.

Tour Types

The tour is formed by visiting cameras one by one. At each camera there exists the option of choosing one of several unvisited cameras as the next destination. The way this is decided strongly affects the overall length of the tour, minimization of which implies a travelling salesman problem. It also affects the speed/rate of coverage, with different choices resulting in more or less redundancy in revealed area. Our approach is thus to evaluate several

heuristics, focusing on simple, greedy choices that at least intuitively should tend to short routes and rapid discovery.

Nearest (N) - In nearest tours, we always try to get to the camera that is nearest to the current point based on Dijkstra distance values between cameras. The aim is to visit all the cameras in one region first so that the NPC does not have to return to it later. Although this is simple and avoids visibility calculations, it has the disadvantage that increase in coverage is not guaranteed, and can be nonexistent while in the same overall convex region.

Farthest (F) - This is the opposite of the N tour. Although this approach is not likely to be efficient in terms of distance travelled, the idea is that the farthest camera likely covers a region that shares the least overlap with the current camera, and, therefore, will yield the largest increase in coverage.

Nearest Non-Visible (NNV) - Instead of considering all cameras like the N tour, we consider only the ones that are not directly visible to our current camera. We are attempting to get a larger increase in coverage by going to a region that has less overlap with the current one, and at the same time minimize the increase in tour length by visiting a camera that is nearby.

Maximum Union (MU) - In this tour type we choose the camera that will provide the maximum increase in coverage. At all times the area of the map that has been seen up to this point in the tour is known (by merging the visibility polygons of all visited camera and non-camera nodes). The increase in coverage an unvisited camera provides is measured by merging the area it sees with the area of the map seen so far.

Maximum Union: Distance (MUD) - The maximum union:distance tour accounts for both coverage and distance. We now take the increase in coverage that a camera would provide and divide it by the increase in tour distance that would be incurred when travelling to that particular camera, selecting the camera that gives the maximum ratio. This makes sense since we want to maximize coverage and minimize distance travelled.

RESULTS

Our goal in experimental analysis is to better understand the way our algorithm works in real game contexts. For this we thus chose a number of relevant game maps, and primarily measured total tour distance necessary for complete coverage, using the different heuristics presented in the previous section. This is of course not the only optimization criterion. Tours may also differ, and be more or less preferable, in how the map is revealed—ideally, for human guidance, we would combine a short tour with continuous, large increases in coverage, and so limit the time and area which is repeatedly re-covered in the course of exploration; NPC exploration may, alternatively, aim for a high novel coverage gradient either early or late in the search, giving the player less or more chance of evading detection respectively. Note that as we are mainly interested in properties of the tour result, and all the information necessary to perform a tour can be computed at map construction time, we do not measure computational performance of tour decisions.

Many other factors are also relevant to our design and may have significant influence. Table 1 lists 5 different feature categories that could affect a tour in a map, along with the

Table 1: Features and Metrics

Roadmaps	Tour Types	Starting Point	Granularity	Target	Metrics
Shortest-path	Nearest	Arbitrary	Regular	Camera	Distance
Triangulation	Farthest	Extreme	Double	All	Coverage
	Nearest non-visible	Centre	Triple		
	Max Union				
	Max Union: Distance				

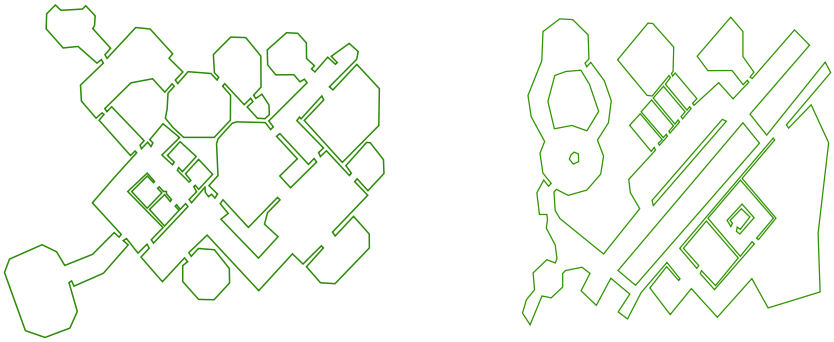
metrics that we use to indicate the quality of a tour. We independently vary the choice of roadmap, path heuristic (tour type), starting position in the map, granularity of observations, whether the path heuristic includes only camera positions or all points (target), and compare the combinations in terms of both distance and coverage properties. Not all combinations are interesting of course, and space constraints prevent us from showing all possibilities, so here we focus on independently evaluating the impact of each factor, leaving other, dependent factors set to their default value, which is the first value shown in each column. Starting positions begin at a camera location, giving us a maximum of n different tours for n cameras, with aggregate results summarizing over these sets.

Maps

The experiments were carried out on maps from popular commercial games. Six different maps were taken from online sources (wikis and/or game-maps.com), three from the role playing game *Pillars of Eternity* (figures 5a, 5b and 5c), and one each from *Witcher 3* (figure 6), *Call of Duty 4* (figure 4), and *Arkham Asylaum* (figure 7). These maps were selected for their availability in terms of accurate and traceable screen images, their relative size and geometric complexity, and for including both building and dungeon interiors. Note that we analyze only the base 2D representation (we model the stairs between levels in the *Pillars of Eternity: Doemenel Manor* as a long corridor). We did not include exterior maps, as these tended to have ill-defined boundaries and obstacles, making a geometric representation more arbitrary.

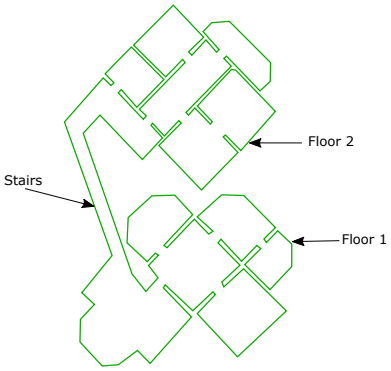
Coverage Measures and Tour Instances

Figure 8 shows two graphs that measure relative coverage (y-axis, percentage of the map's full area) versus tour length (x-axis, in Cartesian units), for each of our 5 tour types. All the tours are shown in full except for the Farthest (F) tour, which is truncated, as it takes considerably longer to finish than the other tours. The left graph shows progression of the tours when starting from one camera point, and the right shows tours constructed from a different random starting point on the same map. These coverage graphs are interesting in showing how individual tours vary in the speed at which coverage is achieved (slope), with horizontal segments indicating periods in which nothing new is discovered. In this we can see that despite good initial coverage the very long tail of F is probably undesirable, and that N and NNV seem to alternate periods of large discovery with long redundancy. There is high variance though, and so we now consider other factors on aggregate results, concentrating on tour length.



(a) Raedric's Hold Dungeons

(b) Copperlane Catacombs



(c) Doemenel Manor

Figure 5: *Pillars of Eternity* maps

Aggregated Comparison of Tour Types

To properly compare the tour types we have to run each tour type (e.g. N tour) on all starting points (cameras), then note the length of each tour instance (e.g. each individual N tour), and contrast the behaviour of the aggregated data to the data from other tour types (e.g. all N versus all MU). This aggregated behaviour is represented as a *violin plot* which is simply a probability distribution of tour lengths. Figure 9 shows the violin plots obtained from *Copperlane Catacombs*. The height of each region represents the range of the length for tours of that type and the width at any point is proportional to the likelihood of that particular value. The widest region in each plot is the most likely value of the length for that tour type. Here we can see that despite the high variance in individual behaviours, in a general sense the MUD tour performs better than other tours for all starting cameras. N is always better than MU, and both overlap with NNV. NNV seems to have the widest variety in tour lengths, a fact that maybe helpful if a developer wants different NPCs to perform at different kinds of efficiency over a wide range. The relative ranks of the the tour types are the same in 3 other maps (*Doemenel Manor*, *Crash*, *Arkham Mansion*), and MUD performs the best in all maps. The ranking between MU, N and NNV have some variance in the other cases, indicating that finer aspects of this ranking are also map dependent.

MUD's dominance suggests that distance and coverage work better as heuristics when used in combination rather than individually. In trying to get to the camera with the best coverage increase, MU likely goes to points that are far away, ignoring nearby cameras that were

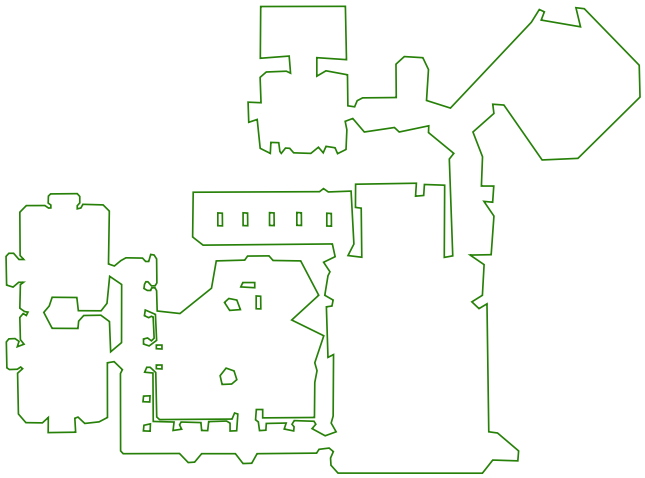


Figure 6: *Witcher 3:* Palace of Vizima

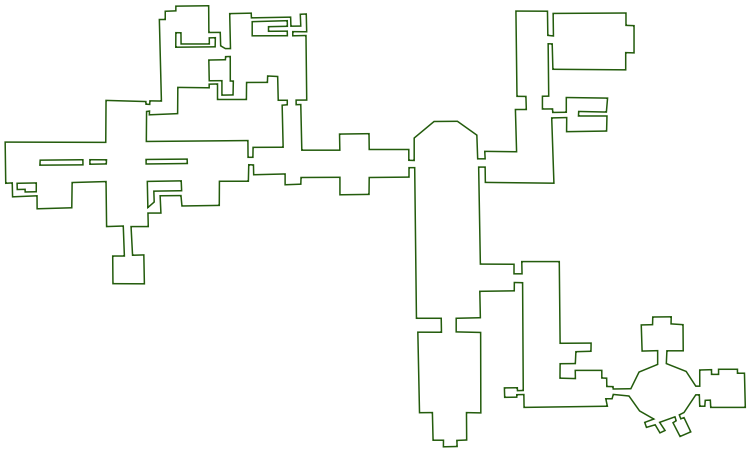


Figure 7: *Arkham Asylum:* Arkham Mansion

crucial to achieving complete coverage in the local area. This means it will have to come back to the same area later, an inefficiency which is a major issue in player disastification with automated exploration. N is likely to ensure the local area is covered, since it is trying to move to the nearest points, but will have some redundancy, as in an area densely populated by cameras it is not necessary to visit all of them.

Target

At each decision point in the tour we choose among unvisited camera nodes in the roadmap. The roadmap, however, contains many more nodes than just camera points, and one of these may provide better increase in coverage than a camera point at that given decision point. Figure 10 shows results from *Crash*. Here, we compare each tour type in two modes, a) "All", which means all roadmap points are possible candidates, and b) "Camera", our default strategy, which selects only from camera points.

Results for all the maps, except *Palace of Vizima*, displayed common patterns. When using cameras as the target, N and NNV perform better, although sometimes only slightly so. MU

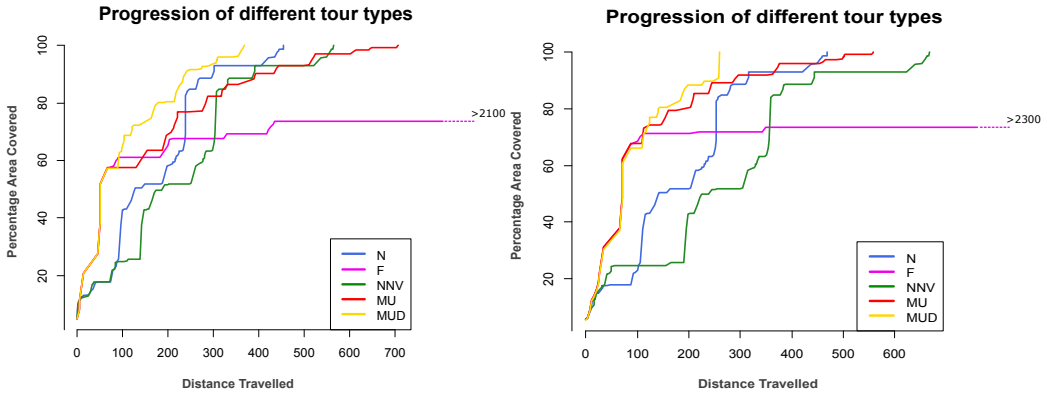


Figure 8: Relative coverage over distance for different strategies. Left and right graphs are from two different camera starting points on *Copperlane Catacombs*.

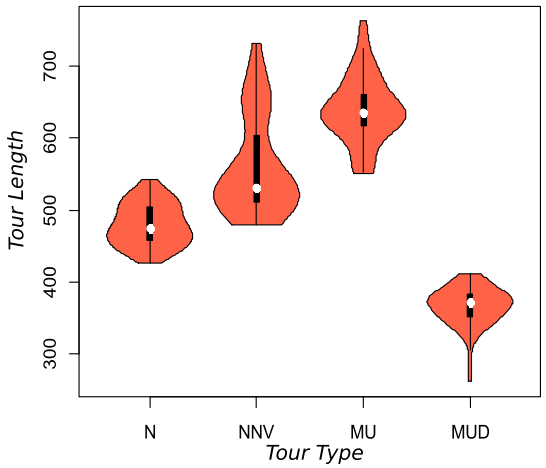


Figure 9: Comparison of tour lengths in *Copperlane Catacombs*.

and MUD generally worked better when all nodes were targeted, otherwise they had similar performance under both settings. In *Palace of Vizima*, all except NNV are better in the "All" mode.

In these results we can observe that distance appears to be a better heuristic when cameras are used as the sole target, whereas coverage is better when all nodes can be tested. MUD is observably improved when all nodes are considered. Unlike points of good proximity, points of improved coverage are not spread about symmetrically. Hence, more choices increases the likelihood of a better decision.

Starting Positions

As shown previously in figure 8 when analysing individual tours, we noticed that starting positions have a definite effect on the progression and total length of a tour. This can be seen purely as a (large) source of variance, but it is also possible that, viewed coarsely, some starting areas are better than others. In particular, starting from near the center of a map should give relatively equal access to other locations, while starting close to an edge of the map will bias the cost (and benefit) of some locations over others.

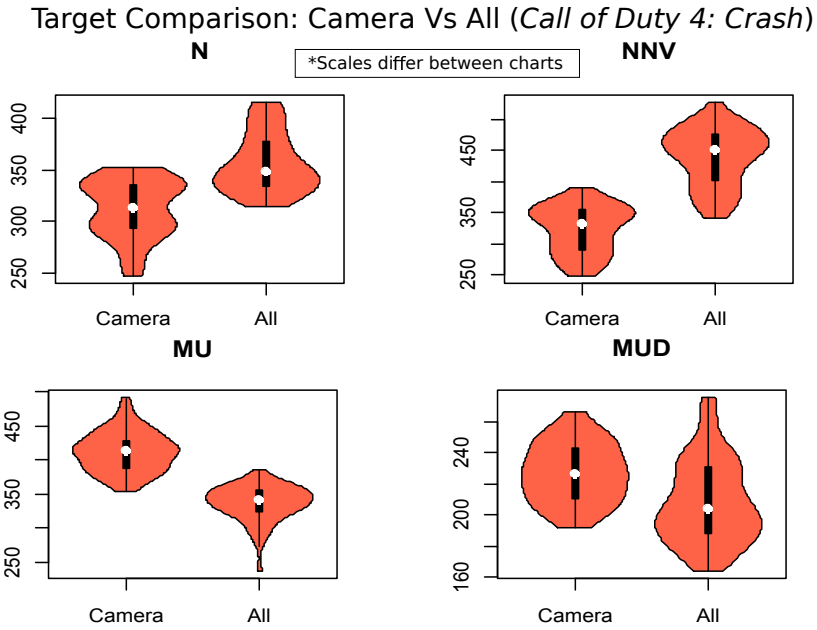


Figure 10: Tour length given different target choice sets.

We therefore tried to see if tours starting from central and extreme (edge) regions of the map, computed based on overall map boundary, differed in any way from all other tours or a random tour. We do not show plots of this, as no predictable pattern could be found. This implies that either the starting position is not itself an important factor, or that the occlusion and pathing constraints imparted by the geometry of a map are more important than the starting point's relation to the map boundary. As future work it would be interesting to explore whether this is still true for other interpretations of center and extreme positions, such as by using a network centrality measure on the roadmap (Freeman et al. 1991) designed to represent a center (or extreme position) with respect to path distance.

Granularity

Currently we are only checking visibility at the endpoints of path edge segments as a way of approximating everything that the NPC sees as it walks along a line. Other than using full, and expensive edge visibility algorithms, a scalable way to improve this approximation is to check visibility at additional points between the endpoints, *i.e,* checking points at higher granularity. This increases the coverage generated by each line of movement, changes decisions made based on coverage, and also possibly allows a tour to stop prematurely if full coverage is achieved earlier. We experiment with granularity by uniformly dividing up the edges using a fixed interval. The interval is the average edge length of the roadmap. At double and triple granularity the interval is the average distance divide by 2 and 3, respectively.

Figure 11 shows the results of granularity experiments on *Palace of Vizima*, with other maps showing generally similar trends. In almost all cases there is a marked improvement in tour distance when checking visibility polygons for points at a higher granularity or frequency. Choice of coarse or fine-grained observation points can thus be a useful scaling factor to weigh against performance, although the benefit to tour distance is not always in proportion:

the improvement sometimes happens at triple granularity or stops after double, suggesting there are performance plateaus or even thresholds beyond which increased granularity might not offer further significant improvement.

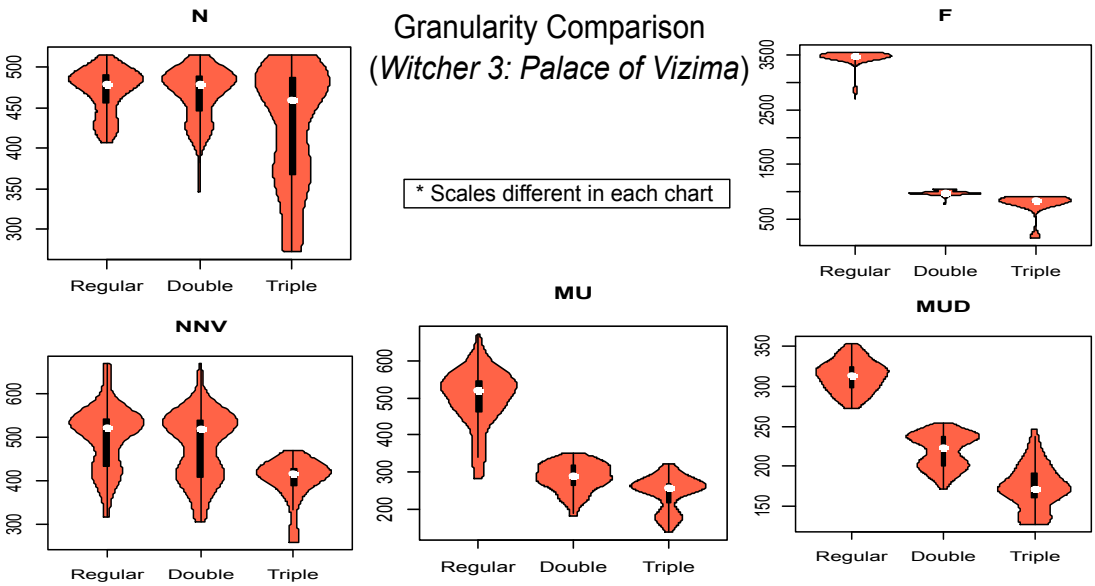


Figure 11: Tour length given granularity choices.

Roadmaps

A roadmap heavily influences a tour's geometric nature by defining the set of possible movements from a point *A* to a point *B*. All the results above were based on tours using shortest-path roadmaps. We thus repeated the experiments using a triangulation roadmap on three of the maps (*Copperlane Catacombs*, *Doemenel Manor*, *Crash*). We do not show specific results, as overall we did not observe many interesting differences in the weighing of our different factors due to the roadmap choice. Tour types maintained similar ranks, and while target and granularity did not have as significant an effect on the triangulation roadmap, improvements and lack thereof were in the same general directions.

CONCLUSIONS & FUTURE WORK

The need for good exploration algorithms is important to many modern games, giving NPCs more realistic behaviours, and reducing player frustration with ad hoc heuristics. We proposed a simple method with a strong guarantee of being exhaustive, and evaluated the impact of a variety of practical factors on its behaviour using multiple, realistic game maps. Heuristics that balance coverage and distance generally do best, with improvements also possible through increased granularity of observation, at least in terms of overall length of an exploratory tour.

An ideal next step would be a qualitative assessment of our method by implementing it in existing games and using human testing to gauge impact on player experience. This would provide useful data on the relative importance of redundancy and novelty in exploration, giving more weight to a quantitative evaluation of a tour from that perspective, in addition to overall tour length. There are also, of course, many additional complexities in how

games present or model observation. Limited field-of-view, lighting issues, and use of full, truly 3D spaces would be interesting to explore, and although these complexities can easily change the underlying theoretical problem, it may be possible to use a similar approach based on constructing a tour of sites that collectively provide a coverage guarantee. Starting from a known solution of any form, tour optimization—introducing short-cuts that bypass intermediate or final goals if coverage can still be guaranteed, or the inverse—degrading a tour through side-excursions, might be a feasible and scalable means of modifying tours to achieve different gameplay effects, while still ensuring an exhaustive result. Finally, it would be interesting to extend our design to contexts that do not assume prior knowledge of the map, both to mimic more realistic exploration behaviours, and to address multiplayer environments that require global map knowledge be restricted in order to reduce opportunities for cheating.

ACKNOWLEDGEMENTS

This work supported by the Natural Sciences and Engineering Research Council of Canada, Application ID #249902.

BIBLIOGRAPHY

Amigoni, Francesco. 2008. “Experimental evaluation of some exploration strategies for mobile robots.” In *IEEE International Conference on Robotics and Automation,* 2818–2823. IEEE.

Asano, Takao. 1985. “Efficient algorithms for finding the visibility polygons for a polygonal region with holes.” *Transactions of IECE of Japan* E-68:557–559.

Chin, Wei-Pang, and Simeon Ntafos. 1986. “Optimum watchman routes.” In *Proceedings of the second annual symposium on Computational geometry,* 24–33. ACM.

DCSS Devteam. *Dungeon Crawl Stone Soup.* https://crawl.develz.org/.

Dror, Moshe, Alon Efrat, Anna Lubiw, and Joseph S.B. Mitchell. 2003. “Touring a sequence of polygons.” In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing,* 473–482. ACM.

Fisk, Steve. 1978. “A short proof of Chvátal’s watchman theorem.” *Journal of Combinatorial Theory, Series B* 24 (3): 374.

Freeman, L. C., S. P. Borgatti, and D. R. White. 1991. “Centrality in valued graphs: A measure of betweenness based on network flow.” *Social Networks* 13 (2): 141–154.

Ghosh, Subir. 2007. *Visibility Algorithms in the Plane.* Cambridge University Press.

Hagelbäck, Johan, and Stefan J. Johansson. 2008. “Dealing with fog of war in a real time strategy game environment.” In *IEEE Symposium On Computational Intelligence and Games,* 55–62. IEEE.

Hoffmann, F., M. Kaufmann, and K. Kriegel. 1991. “The art gallery theorem for polygons with holes.” In *Symposium on Foundations of Computer Science,* 39–48. October.

Isla, Damián. 2013. “Third Eye Crime: Building a Stealth Game Around Occupancy Maps.” In *Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* 206–206. AAAI.

Kazazakis, Giorgos D, and Antonis Argyros. 2002. “Fast positioning of limited-visibility guards for the inspection of 2D workspaces.” In *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 3:2843–2848. IEEE.

LaValle, Steven M. 2006. *Planning algorithms.* Cambridge University Press.

O’Rourke, Joseph. 1987. *Art gallery theorems and algorithms.* Vol. 57. Oxford University Press.

Sim, Robert, and Gregory Dudek. 2003. “Effective exploration strategies for the construction of visual maps.” In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003,* 4:3224–3231. IEEE.

Sturtevant, Nathan R. 2012. “Benchmarks for Grid-Based Pathfinding.” *IEEE Transactions on Computational Intelligence and AI in Games* 4 (2): 144–148.

Suri, Subhash, and Joseph O’Rourke. 1986. “Worst-case optimal algorithms for constructing visibility polygons with holes.” In *Proceedings of the second annual symposium on Computational geometry,* 14–23. ACM.

Tan, Xuehou. 2001. “Fast computation of shortest watchman routes in simple polygons.” *Information Processing Letters* 77 (1): 27–33.

———. 2007. “A linear-time 2-approximation algorithm for the watchman route problem for simple polygons.” *Theoretical Computer Science* 384 (1): 92–103.

Thrun, Sebastian. 2002. “Robotic mapping: A survey.” Chap. 1 in *Exploring artificial intelligence in the new millennium,* edited by Gerhard Lakemeyer and Bernhard Nebel, 1–35. San Francisco, CA: Morgan Kaufmann.

Tovey, Craig, and Sven Koenig. 2003. “Improved analysis of greedy mapping.” In *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 4:3251–3257. IEEE.

Various contributors. *Automated exploration versus manual exploration.* https://www.reddit.com/r/civ/comments/18wtqz/what_is_better_auto_explore_or_doing_it_yourself/.

———. *Civilization Modding Wiki.* http://modiki.civfanatics.com/.

Wang, Pengpeng, Ramesh Krishnamurti, and Kamal Gupta. 2010. “Generalized watchman route problem with discrete view cost.” *International Journal of Computational Geometry & Applications* 20 (02): 119–146.

Xu, Anqi, Chatavut Viriyasuthee, and Ioannis Rekleitis. 2014. “Efficient complete coverage of a known arbitrary environment with applications to aerial operations.” *Autonomous Robots* 36 (4): 365–381.