Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-20)

Dynamic Guard Patrol in Stealth Games

Wael Al Enezi, Clark Verbrugge
McGill University
Montréal, Canada
wael.alenezi@mail.mcgill.ca, clump@mcgill.ca

Abstract

Guard patrols are common in stealth games, both for a sense of realism and to form a basic challenge to acting out covert behaviour. Automated patrol route construction, however, is difficult, as simple approaches based on randomized movement or goal selection tend to result in unrealistic and overly repetitive guarding motion, and poor area coverage. In this work we introduce an approach based on modelling location "staleness" in a manner similar to *occupancy maps*, but applied to both a grid representation and a dynamically constructed coverage-visibility mesh. The latter provides a continuous abstraction of the game environment which can be efficiently used and maintained for real-time decision making. We use Unity3D and multiple game levels to compare our grid and mesh-based results, showing the advantages of the visibility mesh over the grid-based approach.

Introduction

A basic challenge in stealth games requires players avoid detection by enemy agents (guards). The choice of guard motion depends on context, but guards are often expected to *patrol*, or continually traverse a given area in search of enemy (in this case player) agents. The guard behavior can be hard-coded by the designers for the game level, but crafting such routes to guarantee coverage represents is a design concern, and in the case of procedurally generated levels, a hard-coded patrol design may be impossible. Dynamically constructing patrol routes is thus desirable, reducing design effort and allowing even hard-coded game levels to have novel guard behavior in each replay.

Patrol route construction can be formalized in terms of "staleness." Given a game-level partitioned into n regions or points, we associate a "staleness" value s_i with each region. Staleness increases over time, and is reset when the region is observed by a patrolling agent. The quantified goal then, is to create a patrol route that minimizes overall staleness.

In this work we examine two approaches to automated patrol route construction. Our design attempts to give guard AI a simplified representation of the world by modelling relative staleness of locations, encoded in either a typical, discrete grid-based context, similar to the use of *occupancy maps* in approaches to exploration problems (Moravec 1989; Isla 2005; Campbell and Verbrugge 2019), or in a continuous, mesh-based spatial decomposition. The latter has the advantage of eliminating discretization artefacts and providing generally better performance, but introduces complexity in mesh construction and in maintaining staleness during mesh transformations or recomputations.

We evaluate our approaches experimentally, testing different pathing decision processes with both world representations over multiple game levels modelled in Unity3D. Our results show that both approaches have similar behavioural properties, although the mesh-based approach (or *VisMesh*) has significantly better runtime performance, and has better motion characteristics with fewer artefacts. These results demonstrate that full-level, dynamic patrol construction can be efficiently realized in a game context.

The major contributions of this work consist of:

- We describe two approaches to dynamic, runtime patrol route construction, using either a simple grid-based model or a more flexible space decomposition[1]. Our techniques are applicable to both defining patrol routes for good level coverage, as well as for solving exhaustive exploration problems.
- We experimentally evaluate and compare these approaches using multiple (greedy) decision heuristics applied to multiple game maps. Implementation within Unity3D shows both approaches are effective, and that a mesh-based approach can be efficiently realized in a real-time game environment.

Related Work

Computing patrol routes is a problem related to area coverage. This problem has been had relatively little investigation from a games perspective, but has received attention in terms of algorithm complexity and in the context of robotics.

Theoretical Work

Perhaps the most well known theoretical work in area coverage is the *Art Gallery Problem* (AGP) (O'Rourke 1987),

[1] A real-time video example of the result can be viewed at https://streamable.com/bmsr00

where Chvátal showed that $\lfloor n/3 \rfloor$ cameras with 360° infinite field of view are sufficient, and sometimes necessary to cover an n-vertex simple polygon (Chvátal 1975). Mobile guards have been considered as an extension to this base problem, and is sometimes referred to as the *Watchmen Route Problem* (WRP). The goal is to calculate the shortest path in a polygon such that the whole polygon can be covered from any point on that path. Optimally solving this problem is proven to be NP-hard (Chin and Ntafos 1986), although for the general case of being able to start from any point in the polygon there is a $O(n^5)$ solution (Tan 2001).

Literature also shows a variety of multi-agent scenarios that apply to the coverage problem (Laguna and Bhattacharya 2019; Ashok and Reddy 2019). Guards in these studies are also assumed to have unlimited range and a 360° view angle. More applicable to most games are the class of "lawnmower problems," which assume a more restricted field of view (Arkin, Fekete, and Mitchell 2000), and address coverage as a problem similar to finding the shortest route for a lawnmower to cut all grass in a field. Game guards with limited field of view can then be considered similar to lawnmowers. Efficient solutions to these problems, however, tend to result in a dense, zig-zag patterns (Falaki et al. 2020), which while good for coverage are not realistic for guards simulating human observation, even with a limited field of view.

Robotics Work

A common representation used for coverage in robotics research is a grid-based world representation—the space is decomposed into a grid of uniformly distributed nodes (Moravec and Elfes 1985). In this representation, coverage is accomplished by traversing all reachable nodes, deterministically or probabilistically (Elfes 1987). Grid-based approaches are necessarily resolution-dependent, with completeness relying on the granularity of the grid. Although one of the simplest approaches to problems based on space decomposition, for larger spaces, this method becomes computationally expensive (Thrun 1998).

Using grid-based representation in robotics allowed for the emergence of a common algorithm for exploring an environment (Moravec 1989). *Occupancy map* algorithms, for example, utilize the existing grid representation of the environment, associating each node with a probability that the corresponding location is occupied, or has other properties, such as explored, traversable, etc. (Moravec and Elfes 1985).

Games Work

The use of occupancy maps in pursuit and patrol expands to the field of games as well. A study used this method to predict opponent position in *Counter-Strike: Source*. It showed a human-like behaviour even in the mistakes it committed in the prediction task (Hladky and Bulitko 2008).

Occupancy maps were also used as a core mechanic in the AI of a commercial game as well. *Third Eye Crime* is a game that introduced an interesting method that incorporates occupancy maps to create an improved knowledge representation for agents to create a more realistic pursuit and search behavior in stealth scenarios (Isla 2013). In Third Eye Crime, after the intruder is discovered and the line of sight has broken, a probability distribution is diffused to the nearby locations. The probability represents the likelihood of the intruder to be in that location (Isla 2005). This resulted in an intelligent guard behavior, however, the algorithm implementation was limited to take place after an intruder has been detected. Before that, guards were either static or patrolling in a prescripted pattern, and did not have intelligent patrol logic to cover the game world.

Occupancy maps were also used to create dynamic, exploratory behavior for an NPC in NetHack, a turn-based roguelike game (Campbell and Verbrugge 2019). Since the game's representation is already a grid-based, the algorithm utilized that information and created an occupancy map to guide the exploration to unseen areas by following a simple greedy approach. Another grid-based approach to exploration used *potential fields* to guide an NPC in navigating an open real-time strategy (RTS) game space while considering the fog of war (Hagelback and Johansson 2008). Occupancy maps and related systems provide a simple yet powerful architecture for the task of coverage and patrol. A major drawback, however, is in relying on simple discretizations of the game world. This method can be costly in larger and obstacle-rich game worlds, where more fine-grained grids are required to give flexibility in movement choices and to best conform to the level geometry.

Very little research has been directed specifically at generating guard patrols in games. Work by Xu, Tremblay, and Verbrugge aimed at guard movement and patrol patterns, using a generated roadmap in the game level and adding a grammar-based route and behaviour construction (2014). This approach produced visually interesting results, but offered only relatively simple, short guard paths, with no attempt to ensure overall coverage. Other work by the same research group showed an approach to exhaustive exploration, a companion problem to patrol, based on constructing a tour of camera locations generated by solving the AGP (Chowdhury and Verbrugge 2016).

However, this approach is not designed to handle agents with partial field of view. A recent study by Seiref heuristically solved WRP on a grid-based environments, however, it does not address real-time scenarios which is a requirement in most commercial games (Seiref et al. 2020).

Methodology

Our method aims to provide guards with a more sophisticated knowledge representation that adjusts to different game scenario parameters, such as the guard's field of view's radius or angle, so that even when used with a very simple decision making AI it produces more intelligent behavior.

The main idea behind this representation is to associate game areas with a measure of how often they were covered by the guards' vision. In stealth games, the guard's vision is usually depicted in the form of a cone of a fixed radius and angle, which is referred to as the **field of view (FoV)**. In our implementation, to keep track of the covered region, we geometrically union the area seen by the guard at a time step t with the FoV at time step $t + 1$. We refer to this as **the covered region**. Figure 1 shows an example of how the

covered region expands. The game space is explored once the covered region equals the map region.

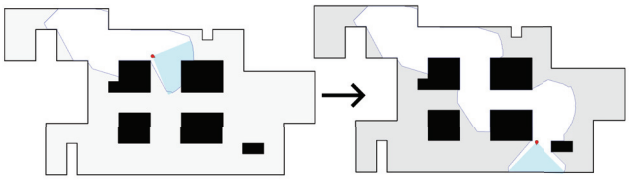


Figure 1: The progression of the covered region in a patrol route

In our study we consider two variations on this idea based on two world representations: grid-based and space decomposition-based, which we refer to as the *VisMesh* approach. For navigation, both representations use the NavMesh.

Grid-Based

In this approach, the space is discretized into a grid of nodes, and each node is either walkable or non-walkable. We build on this by associating each node with a "staleness" value representing how long it has been since that area (node) was last seen. At the beginning of the patrol shift, the staleness of each node is set to 0. Then, at a fixed time-step, the staleness increases for each node by a fixed rate until it reaches a max value.

Once a node is in the covered region, its staleness value is reset and as long as that node remains in the covered region its staleness is fixed to 0.

Space Decomposition-Based (VisMesh)

In this approach, the world is modelled by a polygonal mesh. To create the VisMesh, we consider the overall space geometry which consists of an outer simple polygon with holes (obstacles). Then, we incorporate the covered region into the overall space geometry by taking the geometric difference between the overall space and the covered region. The result will be one or more simple polygons with or without holes in them. The interior of the result is the **uncovered region**. Figure 2 further explains this step.

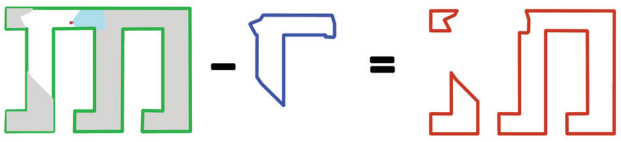


Figure 2: The covered region (blue) is subtracted from the overall space geometry (green) and result in the uncovered regions (red)

After that, we partition the covered and uncovered regions into a mesh of convex polygons. For the decomposition, any convex decomposition algorithm will work, however, for our implementation we used Hertel-Mehlhorn (1985). The result is a mesh of convex polygons that are either contained in the covered region, referred to as **seen polygons**; or which lies outside the covered region, the **unseen polygons**. Convexity is not strictly required in our design, but it simplifies the construction and allows us to build on existing algorithms.

After building the mesh, we associate each convex polygon with a staleness value. In seen polygons, the staleness is set to 0, and in unseen polygons, the staleness is calculated by the weighted staleness sum of the intersecting polygons from the VisMesh of the previous time step. This will be further explained in the next section. Lastly, the unseen polygons staleness is incremented by a fixed rate.

The process of reconstructing the VisMesh and incrementing the staleness of the unseen polygons is potentially expensive if done every frame. In our implementation we compute it at every waypoint in the guard's generated path. An alternative approach would be to compute it over a fixed number of frames, which allows one to scale the approach by trading accuracy for performance.

Updating Knowledge Representation

The update is straightforward for the grid-based approach. We increase the staleness of the nodes outside the covered region at the defined rate per time unit, and we set the staleness of the nodes inside the covered region to be 0.

As for the VisMesh approach, when the guard moves we repartition the space from scratch. This results in a new VisMesh that contains seen polygons and unseen polygons. Staleness of seen polygons is trivially set to 0; however, for the unseen polygons, we need to map the staleness values in the previous mesh decomposition to the new mesh. Staleness is thus calculated from the weighted staleness of intersecting polygons from the previous VisMesh, then the staleness of the unseen polygons is increased by the defined rate. Algorithm 1 further explains the Staleness calculation for the VisMesh.

Algorithm 1 VisMesh staleness calculation

Require: $V_{previous}$, the previous VisMesh.
Require: $V_{current}$, the current VisMesh.
for each $v_c \in \mathcal{V}_{current}$ **do**
  $Stale(v_c) \leftarrow 0$
  if $IsUnseenPolygon(v_c)$ **then**
    for each $v_p \in \mathcal{V}_{previous}$ **do**
      if $DoIntersect(v_c, v_p)$ **then**
        $aDiff \leftarrow intersectArea(v_c, v_p)/Area(v_c)$
        $Stale(v_c) \leftarrow Stale(v_c) + Stale(v_p) * aDiff$
      end if
    end for
  end if
end for

As the guard patrols the area, the covered region expands. Once the covered area equals the overall area the guard has explored the whole map. However, the task of the guard is to continue patrolling the area, not just to explore it. To do that, the covered area must be reset in some fashion so subareas can again become stale. The condition of resetting the covered region can result in two patrol modes.

Patrol Modes

We defined two main patrol modes, differing in how often the covered area is reset. The first is an **explorative** patrol; the covered region is reset when it equals the overall area. The second is a **simplified** patrol; in it, the covered region is reset when it crosses a fixed percentage of the overall space area.

Our goal of creating two modes is to explore various possible patrol behaviors. For example, In the case of the **explorative** patrol, the guards are guaranteed to cover the overall space at least once during a patrol shift. This is because the covered space will be reset once the overall area is explored, and thus the unseen area will be precisely calculated. Once the overall space is covered, the guards will re-explore the space until it is covered again and so on. While this method guarantees complete coverage. It might be more suited for exploration scenarios where the agent is searching for a static object.

In the **simplified** patrol, the covered area is reset more often. This will restrict the guard's covered space not to exceed a fixed percentage of the overall space, which in turn will allow the unseen area to be diffused when re-decomposed. When the vision mesh is updated, recently seen polygons might be merged with unseen polygons with high staleness. This diffuses the corresponding sub-space and distributes the staleness over it. We refer to this as staleness diffusion. Figure 3 shows an example of how staleness is diffused when updating the VisMesh. In our setup, we considered the reset threshold to be 25%. We expect this value to provide a good balance of the staleness diffusion in the map.

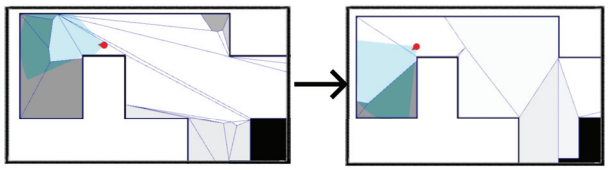


Figure 3: Example of how staleness is diffused after updating the VisMesh. The darker the region the staler it is (solid black for obstacles).

After updating the knowledge representation, whether it is the grid-based or the VisMesh, the guard plans a path to investigate a certain location.

Decision-Making

One of the major advantages of our approach is that it allows the implementation of various decision-making strategies in constructing patrol routes. Qualitatively, we observe that even the most basic behavior of simply finding the path to (the centroid of) the stalest polygon in the space already demonstrates a very believable patrol behaviour, better than simple randomized approaches, and without the cost and complexity of manual scripting.

We expect that the guard's performance can be further enhanced by implementing more sophisticated decision-making techniques like behavior trees, GOAP, or HTNs.

Experimental Context

To test our approach, we recreated, using Unity3D, a simplified top-down representation of several maps from commercial stealth games, as well as some synthetic tests (not all of which we can show here). Each scenario involves a fixed-time for a patrol simulation, beginning with the guard located randomly in the game space and tasked with patrolling the game space according to our different representations, patrol modes, and decision strategies. In our setup, we tested multiple episodes for each combination of the following variables:

- **World Representation**: Grid-based or VisMesh.
- **Patrol Mode**: Explorative or Simplified.
- **Strategy**: This is the strategy the guard used to patrol the game space. It greedily chooses the next map unit and pathfinds to it using a navigation mesh; In VisMesh, it will go to the centroid of the target polygon. In grid-based, it will go to the position of the node. We consider two variants, one based on absolute staleness, and one weighting distance and staleness:
  - **Stalest Unit**: The guard goes to the map unit with the highest staleness.
  - **Weighted Distance Stalest Unit (WDS)**: In this strategy the guard considers both staleness and (Euclidean) distance. We assess each unit (node or polygon-centroid) with the fitness value shown in equation 1.

$$WDS(x) = \frac{1}{D(x)} + Stale(x) * c \qquad (1)$$

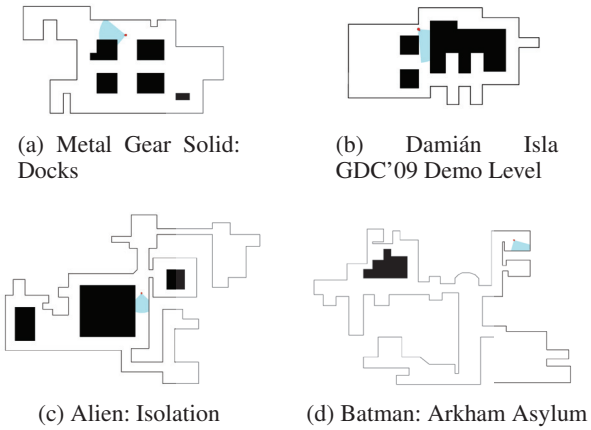
    Where $WDS(x)$ is the fitness value for the map unit x, $D(x)$ is the unit's Euclidean distance from the guard, $Stale(x)$ is the staleness value of x, and c is a coefficient to reduce the weight of staleness compared to the distance. To give more weight to the distance, and assuming fairly large *Stale* values, we set c to be 0.01.
  - **Random**: We consider this as a baseline strategy. The agent randomly chooses a polygon in the NavMesh independent of staleness, and plans a path to a random point in that polygon.
- **Game Map**: We chose several maps to use in testing. Most are derived from stealth-based commercial games. Figure 4 shows the layouts of these maps. For larger maps we selected a representative excerpt to reduce testing time. The maps are:

(a) The "Docks" level from *Metal Gear Solid 1*.

(b) A map created by Damián Isla in his GDC'09 Demo (Isla 2014 accessed June 11 2020).

(c) An excerpt of the "San Cristobal Medical Facility: Basic care unit" from *Alien: Isolation.*

(d) An excerpt of of the "Arkham mansion" from *Batman: Arkham Asylum.*



(a) Metal Gear Solid: Docks

(b) Damián Isla GDC'09 Demo Level

(c) Alien: Isolation

(d) Batman: Arkham Asylum

Figure 4: Game Maps

Results

For evaluating the performance of our methods, we ran 60 episodes of each parameter combination on each map, considering different measures of overall patrol quality. For all maps we find common observations on the performance, and thus shown only a selection of results here.

Average Staleness

The staleness of a location indicates the degree of how often it is surveyed, and thust the goal of the guard is to keep the staleness of an area as low as possible. To measure that, we consider the average staleness of the unseen region; a low average staleness represents a well-covered area.

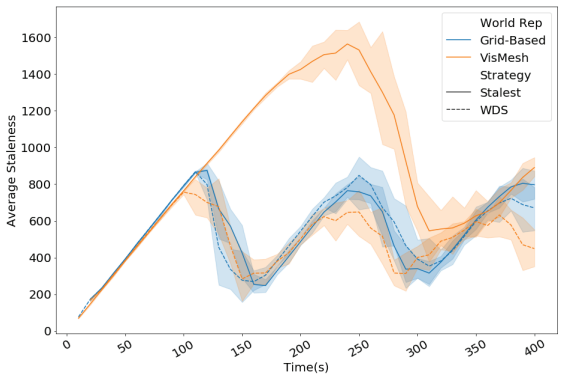


Figure 5: Average Staleness progress for the **explorative** patrol in Alien: Isolation. Orange represents VisMesh, and blue grid-based, with dotted lines for the Stalest strategy and solid lines for WDS. Shaded area represent two standard deviations. The performance of Random is excluded, as it is unable to explore exhaustively.

Figure 5 shows results for the explorative mode. Grid-based and the VisMesh WDS strategy have similar performance, with a see-sawing average staleness as exploration episodes are reset. The VisMesh Stalest unit strategy performs notably worse—as the guard patrols, they sometimes leave small shards of unseen polygons, and the relative uniformity of staleness values with a lack of a distance factor in target selection means these are overlooked as the guard moves to other unseen polygons.

Figure 6 shows the simplified patrol performance. The drawback of VisMesh guard using Stalest unit strategy is overcome, since staleness is diffused which results in larger polygons with a higher difference in the staleness between them. Eventually, this creates a larger difference between the staleness and distances to the centroids between polygons. The random guard, performed the worst since it left several areas unseen.

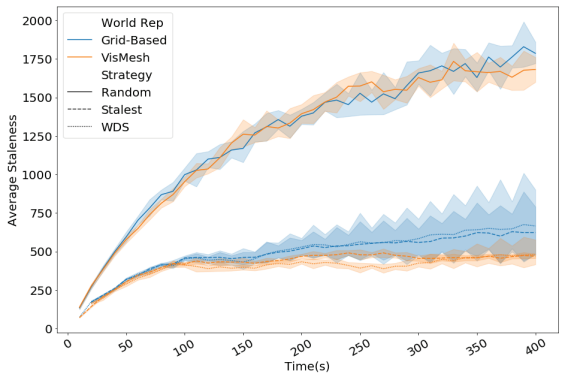


Figure 6: Average Staleness progress for the **simplified** patrol in Alien: Isolation

Treasure Hunt

A patrol route should also be effective at finding objects (or other agents) hidden in the space. To evaluate this, at the beginning of each patrol shift, we randomly distribute 50 "pellets" in the walkable area. The guard does not know of their locations or count. Each time a pellet lies inside the guard's FoV, it is considered to be found and will be removed, and randomly relocated to any walkable location outside the current FoV. This setup provides a quantitative measure of how good the guard is at continually investigating unseen areas.

Figure 7 shows the results, measuring number of pellets found over real-time. We focus on Simplfied patrol results. Here we observe the VisMesh guard finds pellets at a faster rate than other combinations. In VisMesh, the guard is encouraged to spend more time traversing to different locations, while in the other modes the guard focuses more on covering the surrounding area and spends a longer portion of the patrol shift rotating to cover the nearby locations. This results in the guard moving a smaller distance. This is further confirmed in figure 8, where we replace time with distance travelled on the x-axis, showing the VisMesh and Random guards travelled longer distances.

Scalability of World Representation

Performance is a major concern in games, and it is important to evaluate the scalability of our model, which we measure in terms of of memory cost and frames per second (FPS) in relation to map size.

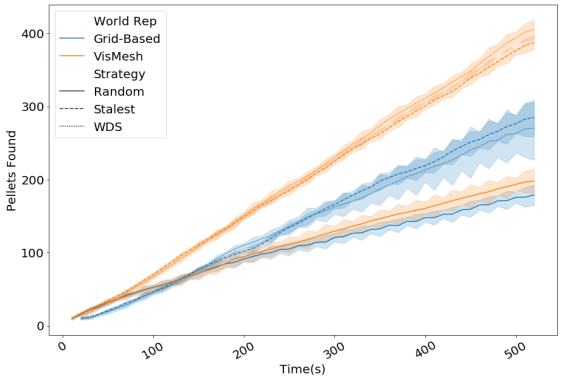


Figure 7: Found pellet progress over **time** for the strategies and World Representation for Batman: AA map using Simplified patrol.

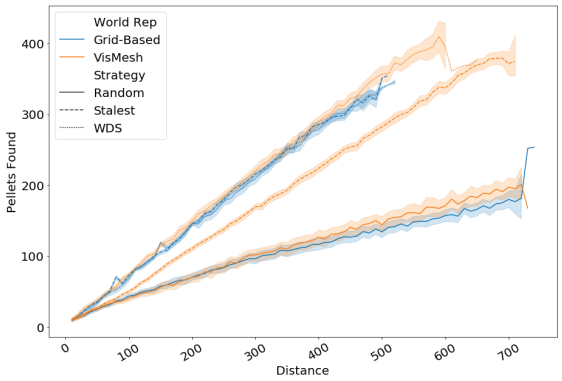


Figure 8: Found pellet progress over **distance** on Batman: AA map using Simplified patrol.

Figure 9 shows results in relation to grid-based approach on the MGS: Docks map, collected using Unity3D's profiler.

As the map scales higher, VisMesh gains both in a relatively better frame-rate, and a reduction in memory requirements. The latter is unsurprising, as the grid-based implementation grows based on the number of grid points, which depends on the size of the map. Coarser grid resolutions are cheaper, albeit at the cost of increasing discretization concerns. VisMesh relies only on map geometry—memory is the same as long as the overall level shape stays the same.

Patrol HeatMaps

Qualitatively, we compared the performance of Simplified patrol between a Random guard and the two methods. Figure 10 shows heatmaps of a single patrol for VisMesh and Random guard patrol, with brightness indicating the relative amount of time an area is observed. We can see a more uniform coverage in VisMesh method, while the Random mainly covered a limited region.

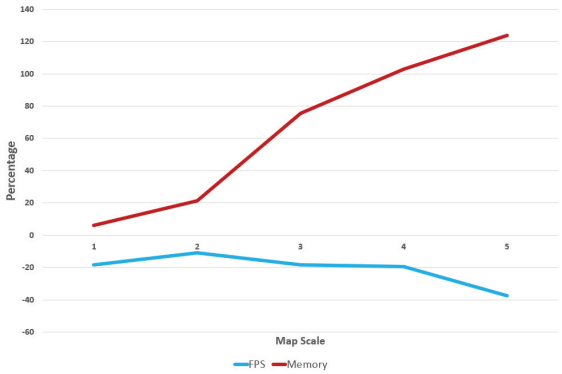
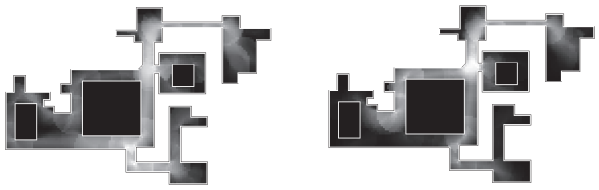


Figure 9: Effect of a scaling the map size on performance. The y-axis represents the average relative percentage of VisMesh to the grid-based representation, and the x-axis is the scale of the map. Grid-based performance for the default map size is 45 FPS, 12 MB and for the scale of 5 it is 34 FPS, 31 MB.



(a) VisMesh (b) Random

Figure 10: HeatMaps for Simplified patrol in Alien: Isolation

Conclusions & Future Work

Guards are an intrinsic element in stealth games. Realistic and effective guarding behaviour has value, in offering greater challenge to a stealth problem, and for improving the sense of immersion.

The goal of our work is to enable efficient and effective dynamic guard patrol. We consider a grid-based model and a continuous, space-decomposition based approach (VisMesh). Both variations can produce appropriate behaviour, although the VisMesh method tends to have better patrol results in terms of being effective at searching and limiting staleness, at least with a suitable decision strategy. It also has the notable advantage of much better memory efficiency, and is thus much more appropriate for larger maps.

For our future work, we are interested in implementing more sophisticated decision-making techniques to work with our world representations. We are also expanding this work to multi-guard patrol scenarios, where multiple guards cooperate, sharing their updates to a global VisMesh in order to partition the patrol space effectively. Player experience is also a concern for real applications of course, and user studies would be useful in determining the extent to which players note and are impacted by different, improved patrol behaviours.

Acknowledgments

This work supported by the COHESA project, through NSERC Strategic Networks grant NETGP485577-15.

References

Arkin, E. M.; Fekete, S. P.; and Mitchell, J. S. 2000. Approximation algorithms for lawn mowing and milling. *Computational Geometry* 17(1-2):25–50.

Ashok, P., and Reddy, M. M. 2019. Efficient guarding of polygons and terrains. In *International Workshop on Frontiers in Algorithmics*, 26–37. Springer.

Campbell, J., and Verbrugge, C. 2019. Exploration in NetHack with secret discovery. *IEEE Transactions on Games* 11(4):363–373.

Chin, W.-p., and Ntafos, S. 1986. Optimum watchman routes. In *Proceedings of the second annual symposium on Computational geometry*, 24–33.

Chowdhury, M., and Verbrugge, C. 2016. Exhaustive exploration strategies for NPCs. In *Proceedings of the 1st International Joint Conference of DiGRA and FDG: 7th Workshop on Procedural Content Generation*.

Chvátal, V. 1975. A combinatorial theorem in plane geometry. *Combin. Theory Ser. B* 18:39–41.

Elfes, A. 1987. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation* 3(3):249–265.

Falaki, P. M. M.; Padman, A.; Nair, V. G.; and Guruprasad, K. 2020. Simultaneous exploration and coverage by a mobile robot. In *Control Instrumentation Systems*. Springer. 33–41.

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

Hertel, S., and Mehlhorn, K. 1985. Fast triangulation of the plane with respect to simple polygons. *Information and control* 64(1-3):52–76.

Hladky, S., and Bulitko, V. 2008. An evaluation of models for predicting opponent positions in first-person shooter video games. In *2008 IEEE Symposium On Computational Intelligence and Games*, 39–46.

Isla, D. 2005. Probabilistic target-tracking and search using occupancy maps. In *AI Game Programming Wisdom 3*. Charles River Media.

Isla, D. 2013. Third Eye Crime: Building a stealth game around occupancy maps. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Isla, D. 2014 (accessed June 11, 2020). *GDC '09: Intro to Knowledge Representation, Occupancy Map Demo*.

Laguna, G. J., and Bhattacharya, S. 2019. Adaptive target tracking with a mixed team of static and mobile guards: deployment and activation strategies. *Autonomous Robots* 1–13.

Moravec, H., and Elfes, A. 1985. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, 116–121.

Moravec, H. P. 1989. Sensor fusion in certainty grids for mobile robots. In *Sensor devices and systems for robotics*. Springer. 253–276.

O'Rourke, J. 1987. *Art gallery theorems and algorithms*. Oxford University Press Oxford.

Seiref, S.; Jaffey, T.; Lopatin, M.; and Felner, A. 2020. Solving the watchman route problem on a grid with heuristic search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 249–257.

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

Thrun, S. 1998. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence* 99(1):21–71.

Xu, Q.; Tremblay, J.; and Verbrugge, C. 2014. Generative methods for guard and camera placement in stealth games. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.