Solving the Take-Down and Body-Hiding Problems

Jorge Morales Díaz and Clark Verbrugge
McGill University
Montréal, Canada
jorge.moralesdiaz@mail.mcgill.ca and clump@cs.mcgill.ca

Abstract

Stealth games challenge players to complete tasks while remaining unnoticed by enemies. Modern versions often provide players with the ability to silently eliminate enemies, inducing a puzzle-like nature to the task which combines finding a path to the goal with the need to discover an order in which enemies can be eliminated. This "take-down" problem is made more complex by the need to hide the resulting enemy body from future observation. We present a version of a sample-based search that can find paths respecting the stealth constraints of both the *take-down* and *body-hiding* mechanics, including multiple variations that address different ways these problems are expressed in games. As a perhaps surprising outcome of our analysis, we show that an additional body-hiding problem can improve the success rate, despite the apparent increase in task complexity.

Introduction

Many modern stealth games mix the basic stealth challenge of remaining undetected with light combat, allowing and sometimes requiring a player to simplify the problem by selectively eliminating enemies that may otherwise prevent them from completing the task. The latter forms a mechanism in its own right, with non-trivial difficulty resulting from enemies observing each other, and thus requiring players find a correct *take-down* order. This task is made more complex by the need to hide the evidence after the fact—*body-hiding* is necessary to stash knocked out enemies in places where other enemies will not observe them.

In this work we explore algorithmic solutions to both the take-down and body-hiding problems in a stealth context. As with other, prior work on stealth games, we build on the *Rapidly Exploring Random Tree* (RRT) search algorithm for heuristically finding stealthy paths (Tremblay et al. 2013). Critical choices like take-down events stress the ability of a basic RRT search to solve such problems, and so we consider different extensions to RRT that bias the search in this respect, and which simulate the use of different degrees of player knowledge of enemy motion. Body-hiding is then layered on these designs. Reflecting different ways body-hiding is integrated into games, we consider two approaches to the body-hiding problem, one based on the need to use designer-specified hiding spots, and one focused purely on ensuring the player task is accomplished prior to any potential evidence discovery.

We perform extensive analysis of different game level mock-ups, considering several factors that affect success, including the distance at which take-down can occur, and relative movement speeds. In simple levels the former parameter dominates, although the relative impact is not uniform in more complex scenarios. Interestingly, and despite the additional complexity, success can be improved by the addition of body-hiding—the need to deposit bodies in hiding spots not seen by enemies also moves the player away from potential observation, reducing chances of search failure from overly aggressive goal or take-down searching.

Specific contributions of this work include,

- We formalize the take-down and body-hiding problems as important features of a stealth game puzzle. This extends the core stealth puzzle model developed in previous work on stealth games.
- We describe several variations on search-based solutions to solving both take-down and body-hiding puzzles. Our solutions improve upon a naive search, and allow for incomplete information scenarios as well.
- Using a non-trivial implementation in the Unity framework, we show feasibility in applying our techniques to models of stealth puzzles found in actual stealth games.

Background

A stealth problem involves pathfinding from one location to another while remaining unseen. Following previous approaches to analyzing stealth problems, we represent the search space as a 2D polygonal game level extruded into a third time dimension. Figure 1 shows an abstraction of this representation. In a stealth game an enemy field of view (FoV) can be modelled as a polygonal area, changing position over time as the enemy moves, forming in our 3D space another structure, more complex but similar to any other obstacle. This allows us to treat stealth in terms of 3D pathfinding.

The *Rapidly exploring Random Tree* (RRT) algorithm can be efficient in 3D pathfinding, and has the advantage of computing a random, heuristic solution. The latter is useful for

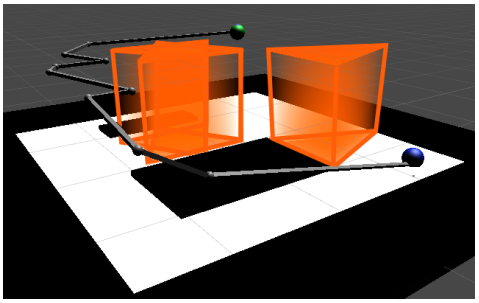


Figure 1: State-space representation of a stealth problem where a 2D scenario is augmented with an extra dimension (vertical) representing the time. A solution path is shown from a starting location (blue sphere) to the goal (green sphere), skirting obstacles (black areas) and enemy fields of view (orange triangular prisms).

exploring a design space or better simulating the variability of human behaviour. Originally presented by LaValle (Lavalle 1998), the RRT algorithm is an iterative process for growing a search tree in a given space by attempting to attach a randomly sampled state from the 3D, non-obstacle state-space to its nearest neighbour in the tree. Nodes successfully attached grow the tree, while invalid samples are discarded. The algorithm terminates when either the goal state is reachable from a sampled state, in which case a path from start to goal can be extracted from the tree, or the given time or sampling budget is exhausted, in which case the algorithm reports failure.

Additional features besides just position information can be added as dimensions to the state space to allow RRT to solve move complex problems. The same, random selection process can then be applied, although with an increased number of (often sparse) state dimensions the ability to grow the tree effectively is reduced. Less uniform selection of random states can alleviate this, provided a suitable biasing strategy can be found.

Take-Down Model

Take-down problems occur in stealth scenarios in which a covert path is impossible, actually or apparently. Players are required to use (violent or non-violent) combat to eliminate an enemy, while still preserving the overall non-detection property. This differs from *distractions*, which change enemy behaviour (Borodovski and Verbrugge 2016), but do not require active enemy engagement.

Quietly eliminating an enemy can be an optional part of any stealth scenario. The more interesting puzzle-like nature of the problem becomes evident when enemies are observing each other. Figure 2 shows an example. The player needs to reach the goal (green dot), but the corridor entrance is protected by the field of view of guard 3 (orange FoV). In order to eliminate guard 3, however, the player must first remove guard 2 (yellow FoV), which requires removing guard 1 (pink FoV). The apparent necessity to eliminate a guard as part of a stealth solution, and a non-trivial chain of such dependencies in particular is the basis for what we call a take-down problem.

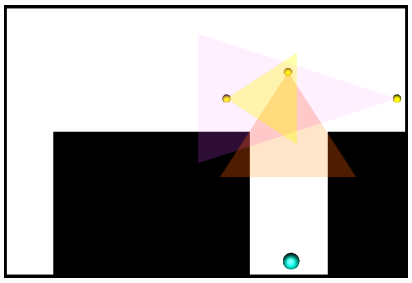


Figure 2: Take-down problem example. Here a guard observes the corridor entrance, while two other guards observe the first and each other. In order to enter the corridor, a player must eliminate the guards in a specific order.

This is more complex in actual game contexts due to the dynamic nature of agent movements, which changes the ordering or introduces temporal limitations on when take-downs can occur. Different game designs can also require the player to eliminate all enemies (whether strictly necessary for covert pathing or not) as a progress condition, making take-down the primary goal.

Basic Take-down Using RRT

In order to analyze the take-down problem we augmented the RRT representation previously used for the pure stealth problem (Tremblay et al. 2013). Each RRT node is now required to include the state (alert or taken down) of each enemy. The random sampling is still performed in the base x, y, t space, with newly attached nodes assuming enemy status is inherited from their parent node. Connectivity between the new node and parent can then use the parent's enemy states in order to determine whether an enemy is alert, and thus whether their FoV inhibits connecting the new node to the search graph.

To enable actual take-downs a player must reach an enemy position undetected. It is of course unlikely that an exact enemy position will be sampled by the RRT, and so an action radius is added that determines how far the player can be from an enemy while still being able to take it down. Even within this action radius, since take-down order is important, it is also necessary to allow a node to choose not to take-down an enemy. For this we use an action probability, changing the enemy state from alert to taken down only with some chance. Note that we assume taking an enemy down is instantaneous, although adding a duration to this event would be straightforward.

We can thus summarize the modified RRT algorithm in terms of the following steps.

1. Sample a new node n from the reachable, non-obstacle x, y, t state space.
2. Find the closest neighbor m in previously sampled nodes. Verify no alert enemy FoV intersects n or the path from m to n. Copy all enemy status from m into n.
3. If n is inside the take-down radius of an alert enemy, based on the action probability either proceed with the

take-down or not. If so, the enemy changes state from alert to taken down.

4. Check if the goal can be reached from n, and if so connect it (and terminate the search).

It is important to notice that the correctness of this basic solution depends on having full information on the game level, including geometry and enemy positions. Success also strongly depends on fortuitous sampling of take-down nodes, which even with an action radius can be unlikely.

Improving the Basic Approach

We improve the low likelihood of sampling take-down nodes through three different techniques, *biasing*, *prediction*, and *correction*. These approaches offer different trade-offs between improvements in success rate and the amount of ahead-of-time, deterministic knowledge of enemy movements that is required.

Biasing An RRT search for a basic pathing problem encounters a similar issue, in that it is unlikely to randomly sample the goal node. A typical solution is to *bias* the search, immediately and artificially sampling the goal node after each newly connected node to test for a direct connection to the goal. A similar approach can thus be applied to aggressively locate take-down nodes.

Specific biasing designs may choose to add new nodes, or to simply replace or modify a sampled node. Our approach to biasing follows the latter strategy: with some probability, a valid, newly sampled node n that would otherwise fail to take-down an enemy e for being outside of the action radius is modified, creating instead a node n' with a geometric position randomly selected to be outside e's FoV but inside its action radius. This approach is not trivially successful, and as with normal sampling we still need to verify that n' is both a valid position in geometry (and time as well), and can still connect to the same parent p in the search tree. The latter is overly conservative—we can search for a new parent as well, but as these checks may be performed for each new node a low cost solution is preferable.

Prediction The bias approach requires full knowledge of enemy positions, both for the basic RRT search and for validating and selecting bias nodes. For players this knowledge is usually heuristic, and take-downs would be more often performed opportunistically, based on an ad hoc judgment made upon observing an enemy.

Our prediction approach attempts to model take-downs in terms of this partial knowledge. Similar to the bias approach, with some probability a node n may be changed to n', a node located at a different, in this case future position. We determine the position of n' by estimating the enemy's motion, dead-reckoning into the future based on its current and previous position one time interval prior. For this we use a simple linear estimation, although the design extends naturally to more complex position estimates, potentially incorporating knowledge of previous positions and level geometry.

Correction Players without advance knowledge of enemy motions will need to react to potential observation by enemies. Upon encountering an enemy (assuming they are not immediately seen), a player will either change course to avoid detection, or make the heuristic choice to perform a take-down. Translated into our RRT search context, we can use RRT nodes normally discarded during the basic stealth pathing as additional sources of knowledge that opportunistically inspire take-down decisions. When a node is sampled such that the path to its parent intersects an enemy FoV, we treat it as a potential player observation: the player has seen an enemy in the way, and now heuristically adjusts their strategy to instead perform a take-down.

Figure 3 illustrates the design. A sampled node (red) is created and a suitable parent node (green) found. Since the node-parent connection intersects an enemy FoV (orange triangle), the new node would normally be rejected. In this design, we instead locate the intersection point (black), and translate it following the FoV edge as a reference to a point outside the FoV but within the enemy action radius (grey dot). Note that we limit this process to intersection points on the left or right "arm" of the enemy FoV—"correcting" paths attempting to enter the FoV from the 3rd side could also be found, but would require more complex pathing.

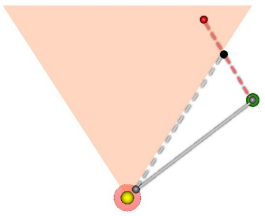


Figure 3: A potential take-down position can be estimated from an otherwise failing sample node.

Body-Hiding Model

Our approach so far has naively assumed that taking down an enemy deletes it from the environment. In many stealth games, however, the take-down problem implies an additional concern in needing to dispose of the evidence—knocking out an enemy leaves a body behind, which if seen by other enemies also results in stealth failure.

Several variations on this challenge exist. In many games players have the ability to move a body, allowing them to hide it from future observation. The main complexity is then in finding an appropriate hiding spot after a take-down while still remaining unobserved by the remaining enemies. Hiding spots can be offered in a few common ways, either based on *preset* or easily identified safe locations provided by the designers—large trash containers, closets or heavily shadowed areas—or through player knowledge, understanding enemy motions to identify or *learn* locations not covered by other, remaining enemies prior to taking them down as well and/or completing the level.

In general, body-hiding induces a separate pathfinding problem after a take-down event. In many games, this search requires the use of a different movement model, since only one body can be carried at a time, often with reduced mobility or other action limitations. To handle this our RRT

search needs a more detailed record of the state of each enemy, which can be *Alive*, *Unconscious* (and assumed carried), *Dropped*, or actually *Hidden*.

Preset

As with the take-down problem, relying on random selection to find a drop location is unlikely (depending the relative area of the preset locations), and so we also use a biased approach. After each node n added to the RRT, if an unconscious enemy is being carried, with some probability we check if any hiding spots are reachable. If so, a random position in a hiding spot will be selected and we attempt to connect it to n and update the enemy state to *Hidden* within n. If a hiding spot is not connected the search continues as normal, with any further connections to n inheriting the fact that an enemy is in the *Unconscious*, carried state. This model allows us to ensure only one body is carried at a time, but does not restrict the ability of a player to move arbitrarily while carrying it.

Learned

Hiding spots may also need to be identified by players heuristically, based on discovering which spots are seen by enemies or not, often over repeated game-plays. For this we add a probability factor to control whether an unconscious, carried enemy is dropped or not, in this case assuming that drops can happen arbitrarily at any (valid) RRT node. If no enemy subsequently observes the drop-site, the body-hiding is successful. If it is seen at some point during the RRT execution, a radius around that position is marked as an unsafe spot (we use an area rather than a position to allow some conservative uncertainty). As the RRT iterates, unsafe spots are better identified, allowing for improved decisions in future iterations, although this also depends on the order and timing of the take-downs, since a safe spot might be found because the enemy capable of observing that spot is already inactive. This motivates the use of a separate *Dropped* enemy state, as we need to check whether *Dropped* enemies are seen during other enemy movements, while *Hidden* ones can be safely ignored.

Experimental Results

Initial tests were performed on a synthetic scenario to determine efficacy and analyze parameter configurations. We then extended this to more complex scenes based on level designs from actual games. Data was gathered using the *Unity 3D* framework on a Windows 10 machine, with an i5-2320 CPU operating at 3.0GHz, and 8GB of RAM. Unless otherwise stated, tests shown in this section were performed applying a value of 100% for both the take-down action probability p_k and the bias probability p_b.

Synthetic Test

Figure 4 shows a simple stealth scenario in which there is a dependency in the order in which the enemies must be taken down: two static enemies are guarded by a third enemy that rotates back and forth to cover the other two with its FoV, forcing the player to take down the enemies in the order $A \rightarrow B \rightarrow C$ (taking down C is optional under the right timing).

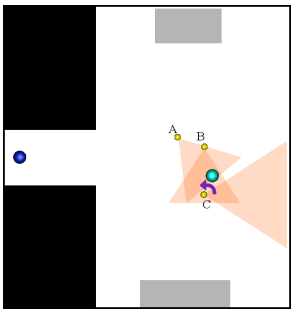


Figure 4: Test Scenario. Starting from the blue circle on the left, the player must take down enemies (yellow) while avoiding their FoVs (orange) to reach the goal (green) in the center. Grey shapes represent designated body-hiding sites.

We used this scenario to investigate the impact of the take-down radius r and the enemy speed v_e on the success rate of the search. Using 100 iterations of the RRT algorithm, each with a maximum of $7,000$ nodes, we examined radius values between 1 and 10, keeping enemy speed constant at $v_e = 10$. At $10\times$ the player speed this ensures a significant challenge even at higher radii. We also examined v_e in the range 1 to 10 while keeping the radius constant at $r = 3$, as a reasonable, intermediate radius value. Under these parameters we only observed changes in the success rate of the naive and correction approaches; both the prediction and bias approaches achieved 100% success with all the configurations.

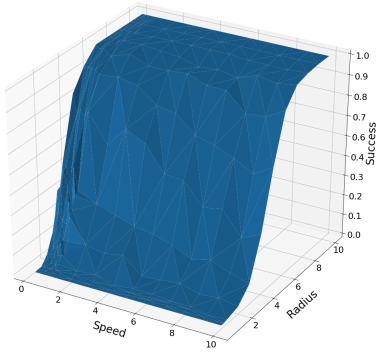


Figure 5: Average success in the Test Scenario for all pairs of parameters with the naive approach.

Results for the naive approach are shown in figure 5. We observed a moderate downward trend as we increase enemy speed, with success rates differing by at most 30%, but a proportionally much larger change due to radius, extending from around 2% when $r = 1$ (take downs require being right behind an enemy), up to 100% at a fairly distant radius of 8. The correction approach improves on this but shows similar trends. The low point occurs at $r = 1$ and $v_e = 10$, with around 40% success, reaching 100% success at $r = 7$. Enemy speed has a slightly larger impact, differing up to

40%—the increased rotation speed of enemy C has more consistent coverage of the other enemy FoVs, resulting in more invalid node choices from the correction bias.

The body hiding parameters where also tested in a similar manner by adding two hiding areas, one on top and one on the bottom area of the scenario. Here we focused our analysis in the body-drop probability p_d and the bias towards hiding zones p_b. We set $r = 3$ and $v_e = 1$ for all the tests, and proceeded as before: we set the bias $p_b = 1.0$ and let p_d range from 0 to 1 in steps of 0.1, and then fixed $p_d = 0.5$ and let p_b range similarly. The results, shown in figure 6, suggest that the success improves in the naive approach mainly when the user is more likely to drop the bodies: naive sampling will not always select nodes inside a hidden zone, and therefore it is better to have a high body-drop probability to take advantage of the times that such nodes are selected.

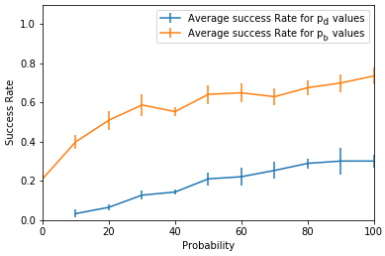


Figure 6: Effect of the body hiding parameters in the success of the *Preset* hiding zones for the Test Scenario using a *naive* search.

Success on Complex Scenarios

Our second set of tests used more complex level designs adapted from popular stealth games. These levels include 2D representations of more complex 3D elements, including overhead traversal and visual-only occlusion. We focus here on analyzing performance and success rate of the different approaches, fixing other parameters ($r = 1, v_e = 1, p_b = 1$, and $p_d = 1$).

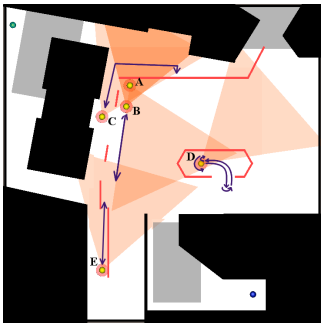


Figure 7: Aventa Station scenario. The player must get from the bottom right (blue dot) past the station (large central obstacle) to the upper left (green dot).

The first scenario, shown in figure 7, is a model of the Aventa Station scene from *Dishonored 2* (Bethesda Softworks, 2016). Thin obstacles (red) represent 3D features that hinder movement but do not occlude FoV—some fences, and a guard observation kiosk. The player is required to enter the station; since we did not model the station interior, we place the goal behind the station.

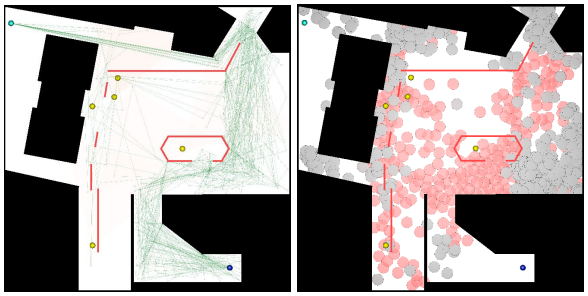
	Success Rate		Nodes		Time	
	$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ
Take Down Model						
Random	19.4%	0.029	5713	728	2.64	0.141
Bias	47.8%	0.048	5360	301	2.27	0.138
Prediction	47.3%	0.043	5615	452	2.43	0.131
Correction	37.5%	0.057	5256	526	2.62	0.143
Preset Hidden Zones						
Random	24.9%	0.045	5662	418	3.22	0.246
Bias	60.89%	0.037	4864	137	3.36	0.321
Prediction	56.4%	0.044	5261	311	3.01	0.163
Correction	46.5%	0.038	5090	351	3.21	0.099
Learned Hidden Zones						
Random	23.6%	0.037	5818	611	3.01	0.148
Bias	44.7%	0.056	5419	306	3.04	0.123
Prediction	42.1%	0.057	5645	338	3.32	0.179
Correction	34.6%	0.048	5197	368	3.49	0.171

Table 1: Results for the Aventa Station scenario for the simple take down model and the body hiding model with different hidden zones strategies. Results based on 10 experiments with 100 iterations using a maximum of 10,000 RRT nodes.

RRT success rate and timing data is summarized in table 1 for our different approaches. *Bias*ing offers more improvement than correction over *random*, likely due to its more aggressive approach to selecting potential take-down spots. Interestingly, the addition of a body-hiding mechanic, at least for *preset* hiding zones ends up improving overall success rate. Designated hiding zones in this scene require the player move well away from the central, well guarded area, reducing the chances of body observation and moving the player into a position where a stealthy solution is easier to discover. This can be seen in Figure 8a, which shows solution paths, most of which pass through or are easily reachable from the preset hiding areas. *Learned* zones, shown in Figure 8b, are not as effective due to the discovery cost, although they do begin to converge to areas similar to the preset ones.

A second test scenario is based on an additional part of the level situated on the back side of the previous scenario. In this scenario illustrated in figure 9 a set of static enemies wait to ambush the player behind various objects. The player (initialized to the blue circle at the bottom of the scene) has two options to avoid being detected: they can go around the main building (black object on the right) to get to the upper section and take down the enemies from there, or use the building balconies (shown as purple sections next to the black obstacles) to move around while staying hidden.

Results of the experiments for this scenario contrast with results from the previous test with respect to body-hiding. Here the *learned* strategy performs better than the *preset* strategy, even with a quite large hiding area (shown in grey). Success rate goes from 33.4%, 77.4%, 79.5%, 57.9% for the *naive*, *bias*, *prediction* and *correction* approaches respectively. to 34.6%, 93.6%, 93.2% and 57.9%. This difference



(a) (b)

Figure 8: Solutions for the Aventa Station scenario for the *bias* approach. (a) Shows an overlay of 30 solutions to the take-down model represented as green lines. (b) Shows the set of learned hidden spots (grey) and exposed spots (red) after 1000 iterations of the body-hiding algorithm.

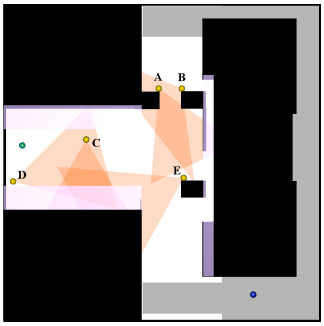


Figure 9: Aventa Alley scenario.

is related to the nature of the enemies in the scenario: all of them are static enemies, and thus the search is able to find consistent hidden spots near to the take-down positions instead of traveling back and forth between the hidden zones and the enemies positions.

Related Work

Stealth game analysis is dominated by the problem of finding a hidden or *covert* path through a game level. Early work on this is found in the robotics domain, where a variety of algorithmic approaches have been explored. Marzouqi and Jarvis, for example, construct a discrete *visibility map* of inter-visible areas, combining it with a *distance transform* to help search for a shortest, most covert path (2004). Visibility maps have been also combined with other techniques; *corridor maps*, for instance, provide a Voronoi-based partitioning strategy that when combined with visibility information allows searching for a path that minimizes visibility values (Geraerts and Schager 2010). Other approaches focus on the impact of known vs unknown environments, uncertainty in observer positions, etc.; Marzouqi and Jarvis survey the area (2011).

Probabilistic solutions have also been considered in game contexts. Johansson and Dell'Acqua construct a hierarchical scene model, using prior experience of an agent in observing enemy positions to define the likelihood of being observed at specific locations (2010). Searching through the resulting graph structure can then locate a path of lowest observation probability. Our work focuses on finding exact, guaranteed solutions, with full level knowledge. We build on Tremblay et al.'s approach to covert pathing (2013), using Lavalle's RRT algorithm for a fast, heuristic search that can generate multiple solutions, as a proxy for potential player choices. This work was later extended to include combat (Tremblay and Verbrugge 2015), and incorporate *distractions* (Borodovski and Verbrugge 2016). Distractions are often found in stealth games, allowing players to create a noise or visual disturbance that attracts guards away from their normal route, exposing additional opportunities.

Performing a take-down is related to the basic *interception* problem, where an agent is required to path toward the location of a dynamic agent. Again, prior work mainly stems from robotics, and is typically focused on probabilistic approaches. Garzón et al., for example, apply the *Risk-RRT algorithm*, a variant of RRT that incorporates uncertainty in avoiding collisions with dynamic obstacles to plan the motion of Unmanned Ground Vehicles (UGV) for surveillance purposes, adapting the algorithm to perform interception by aiming for collisions rather than avoiding them (2014). A covert approach to interception is described by Park et al., who use a predictive technique to select a specific point for interception, and then plan for a path that gets a robot as close as possible to its target while still being undetected (2009). A game context adds complexity in considering the need for multiple interceptions, while hiding the evidence as well, and as far as we are aware ours is the first attempt at extending stealthy pathing to include take-down and body-hiding problems.

Conclusion & Future Work

Stealth game scenarios are designed as a combination of different factors that can either reduce or increase the difficulty for the player in achieving their goals (Smith 2006). Adding mechanics or behaviours to both players and enemies increases the space of player actions and therefore, the complexity for analyzing the design and its solutions. Our expectation was that both take-down and body-hiding would reduce search success, but it is interesting to note that the additional effort of body-hiding can at least sometimes simplify the problem, indirectly forcing a player into locations more advantageous for continuing solution discovery.

Our work presents a few possible approaches to the take-down and body-hiding problems, but it has not explored all possible variations on them. Variants such as having a dynamic take-down radius based on available weapons, the effect of imperfect information, consideration of sound and other forms of detection, among many other stealth game features would be interesting to explore. Our current focus is on improving RRT search. Although reasonably effective, success rates even with just the take-down and body-hiding mechanics are not high, and approaches to biasing or partitioning the RRT search may offer additional improvements.

Acknowledgements: This work was supported by NSERC grant RGPIN-2019-05213.

References

Borodovski, A., and Verbrugge, C. 2016. Analyzing stealth games with distractions. In *Twelfth Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 129–135.

Garzón, M.; Fotiadis, E. P.; Barrientos, A.; and Spalanzani, A. 2014. RiskRRT-based planning for interception of moving objects in complex environments. In Armada, M. A.; Sanfeliu, A.; and Ferre, M., eds., *ROBOT2013: First Iberian Robotics Conference*, 489–503. Springer International Publishing.

Geraerts, R., and Schager, E. 2010. Stealth-based path planning using corridor maps. In *Computer Animation and Social Agents*.

Johansson, A., and Dell'Acqua, P. 2010. Knowledge-based probability maps for covert pathfinding. In Boulic, R.; Chrysanthou, Y.; and Komura, T., eds., *Motion in Games*, 339–350. Berlin, Heidelberg: Springer Berlin Heidelberg.

Lavalle, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Dept., Iowa State University.

Marzouqi, M., and Jarvis, R. A. 2004. Covert path planning for autonomous robot navigation in known environments. In *Proc. Australasian Conference on Robotics and Automation, Brisbane*.

Marzouqi, M. A., and Jarvis, R. A. 2011. Robotic covert path planning: A survey. In *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, 77–82.

Park, J.; Choi, J.; Kim, J.; and Lee, B. 2009. Roadmap-based stealth navigation for intercepting an invader. In *2009 IEEE International Conference on Robotics and Automation*, 442–447.

Smith, R. 2006. Level-building for stealth gameplay - Game Developer Conference. `http://www.roningamedeveloper.com/Materials/RandySmith_GDC_2006.ppt`.

Tremblay, J., and Verbrugge, C. 2015. An algorithmic approach to decorative content placement. In *Experimental AI in Games Workshop (EXAG 2015)*, 75–81.

Tremblay, J.; Torres, P. A.; Rikovitch, N.; and Verbrugge, C. 2013. An exploration tool for predicting stealthy behaviour. In *The Second Workshop on Artificial Intelligence in the Game Design Process*, 34–40.