# Strategies for Multi-Asset Surveillance

William M. Spears
Dimitri Zarzhitsky
Suranga Hettiarachchi
Wesley Kerr
Computer Science Department
University of Wyoming
Laramie, Wyoming 82071
Email: wspears@cs.uwyo.edu

***Abstract*— This paper describes our "sandbox" for the study of multi-asset surveillance, and explores the performance of rule-based control strategies on this task. In order to maximize the probability of detection of targets of interest, it is assumed that the team of unmanned air vehicles (UAVs) must provide maximum sensory coverage of the terrain. We demonstrate, however, both through simulation and mathematical analysis, that this is not always the case.**

## I. Introduction

The focus of our research is to design and build rapidly deployable, scalable, adaptive, cost-effective, and robust networks of autonomous distributed assets. The general purpose for deploying tens to hundreds of such assets can be summarized as "volumetric control." Volumetric control means monitoring, detecting, reporting, and responding to environmental conditions within a specified physical region.

In this paper we concentrate on the task of multi-asset surveillance. We assume there are $\alpha$ UAV assets flying at a constant altitude over terrain that contains areas of forest and non-forest. The assets have sensors to detect targets of interest, and they can determine whether they are over areas of non-forest or forest. The target sensor cannot penetrate the foliage – thus, regions of forest are of lower interest than those without forest. Each asset has a target sensor with a field of view of $\pi {r_t}^2$, and may also have a foliage sensor with a field of view of $\pi {r_f}^2$, where $r_f > r_t$. The total area of the region is much greater than $\alpha \pi {r_t}^2$. The target sensor also has a probability of detection $P_d$ (which is assumed to be 1.0 in this paper). The goal is to locate $T$ targets as reliably as possible, within some time period $t$. Targets can be stationary or mobile. When they move they can take advantage of the forest to provide cover for themselves. Our principal focus will be on whether providing maximum sensory coverage of the terrain implies maximum target detection.

## II. Surveillance Sandbox

In order to study multi-asset surveillance in a methodical manner, we have created a sophisticated simulation tool called "SURVE". SURVE has several modules, including forest generation, target controllers, asset controllers, a genetic algorithm, and modules for data collection. It can be run in either of two modes. In learning mode the genetic algorithm can be used to optimize parameters that are part of some asset controller. In performance mode SURVE collects data on the performance of the optimized controller. Our current implementation is in Java, to allow for cross-platform compatibility.[1]

### A. Forest Generation

Our "world" is a square with sides of length $L$, and $L^2$ discrete grid points. Grid point ($i$,$j$) can contain a tree or remain barren. We designed the algorithm to mimic a real forest, such that there are large clusters of trees, yet there can be holes in the middle. These holes allow assets to view targets.

The algorithm begins by placing an initial number of trees in the environment, and then uses the natural process of "seeding" in order to generate the next phase of the forest. The process continues until the desired amount of forest is reached. The controlling parameters of the algorithm are:

- $n_f$ - The initial number of trees in the forest.
- $R_s$ - The distance that seeds can travel.
- $S_s$ - The amount of seeds spread during seeding.
- $d_s$ - The decay rate of the seeds.
- $F_c$ - The desired forest coverage.

The algorithm begins by initializing $n_f$ trees. The placement of these trees is uniformly distributed about the world. Rather than just initializing one square for a clump of trees, we initialize a cross section, to aid in the development of the forest, see Figure 1(a). Once the initial trees have been placed, the next generation begins. For a given generation the current trees in the population are allowed to seed the areas surrounding them with a distance of $R_s$. The amount of seed laid at each position within this circle is defined as $S_s$. The seeding process can be seen in Figure 1(b). Once everything has been seeded, we begin selecting positions ($i$,$j$) that are seeded and determine if they become trees, by selecting a uniformly distributed random number. If the value selected is less than that of the seeds in the area, then the seeds have taken hold, and grow to become trees. Using this procedure not all seeds will become trees; therefore, we apply a decay to those seeds that have not become trees. Notice that as more trees surround an area, the greater the probability of growing a

[1]The learning aspect of SURVE will not be discussed in this paper.

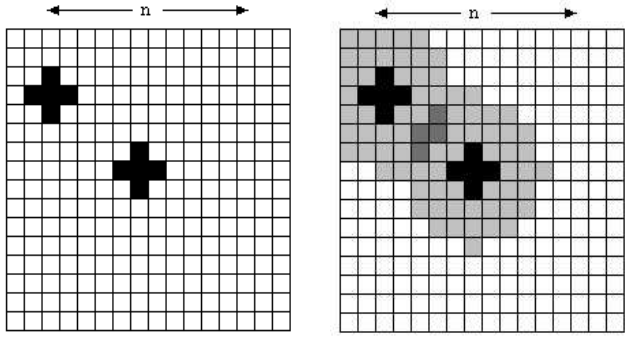


Fig. 1. Forest seeding algorithm. *(a)* The initial forest configuration. Notice that the trees do not represent exactly one square. *(b)* The seeding portion of the algorithm. Notice that the overlap receives more seeds and is more likely to create a new tree.

tree, since it has more seeds. We continue the algorithm until the desired forest coverage $F_c$ has been achieved.

Setting the parameters is relatively easy, knowing that at full seed strength, we will have perfect circles of trees of size $R_s$. As we decrease $S_s$ we begin getting more sparsely populated forests because our seeds are not as hearty and don't survive. The decay rate provides a mechanism to remove old seeds from the population of seeds, again protecting us from realizing perfect circled forests. Figure 2 provides a nice illustration of our forest generator. In this paper our world is $200 \times 200$ in size.

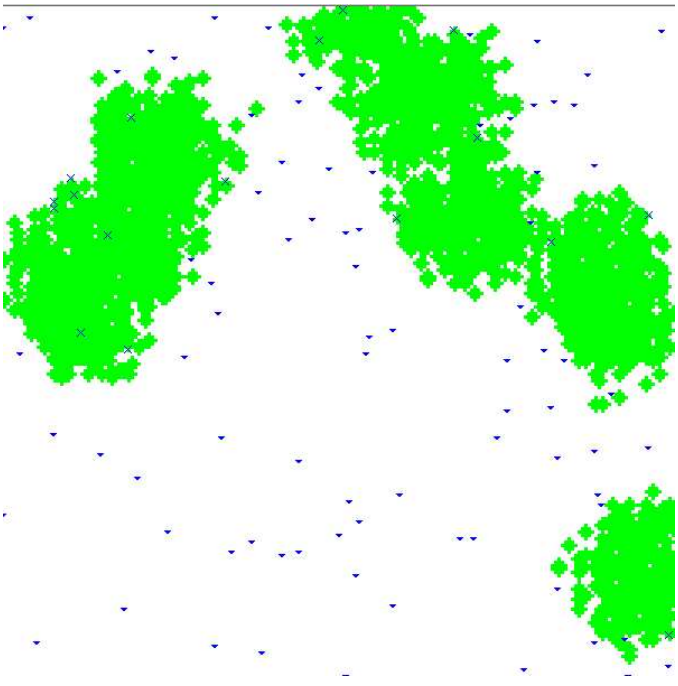

Fig. 2. An example forest with 100 targets of interest.

### B. Target Controllers

Targets can either be stationary or mobile. When they move they try to cross the environment by moving from the left to the right. We are currently creating controllers modeled after real-life scenarios, as found in the Department of the Army's Field Manual "Tank and Mechanized Infantry Company Team" [12]. In this paper we will focus on one target controller, called "Gollum".

Gollum is a "sneaky" target controller that attempts to cross the environment by greedily (and locally) choosing a path with maximum foliage, so that the time a target spends exposed to the assets' sensors is minimized. To achieve the effect, each target has a foliage sensor with two attributes, a foliage radius and a foliage angle, which define how far and how wide the foliage sensor scans the environment ahead of the target to find foliage. In the absence of foliage, each target will move horizontally toward the right edge. However, when foliage is detected, the target will alter its course to reach the foliage cover. Once in the foliage, the target will again continue moving toward the right edge, and attempt to stay within the foliage as long as possible. In this paper, the target foliage sensor radius is 10, and the foliage sensor angle is $60°$, so that the sensor scans $+30°$ and $-30°$ away from the horizontal. These values yield a good trade-off between seeking foliage cover and avoiding large detours in target paths.

Figure 2 shows areas of forest, and 100 targets. The triangle represents a target that has not yet been seen but is visible, and the "$\times$" represents a hidden target that has not been seen. The simulation also uses a "+" to represent a target that has been seen and is visible, and a "|" to represent a target that is currently hidden but has been previously seen (the latter two are not shown in this figure).

### C. Rule-Based Asset Controllers

We have implemented three rule-based asset controllers. The first, called "Straight Line" (SL), is our simple base controller that moves in a straight line at velocity $v$. When the environment wall is hit, the asset rebounds such that the angle of reflection equals the angle of incidence. This extremely simple asset controller does not have a foliage sensor.

To illustrate the behavior of the SL controller, we monitored the "asset map" achieved by this asset strategy. This map shows locations that were seen by at least one of the assets during the given surveillance period. Areas with frequent sensor coverage are denoted by bright white areas, and areas that were observed less frequently are shown in darker hues, with black areas receiving no target sensor coverage at all. Since the target sensor can not penetrate foliage, the forested areas are also black, regardless of whether assets flew over those areas. The sample environment used for this set of experiments is shown in Figure 2.

As can be seen in Figure 3, SL-controlled assets can achieve uniform world coverage, given a moderate number of assets (ten assets appears to be enough to achieve this effect for many sets of initial conditions). However, one weakness is that assets spend time over foliage, which is not productive.

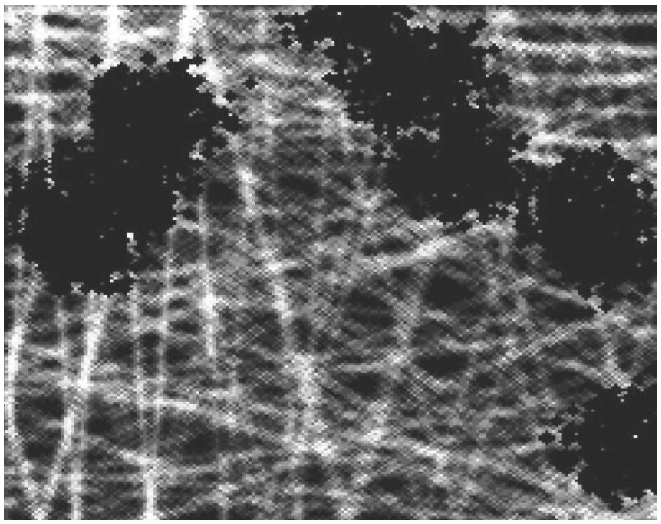

Fig. 3. Sensor coverage with the SL strategy after 1,000 steps.

As a consequence we also implemented a "Straight Line Avoid Forest" (SLAF) controller. This controller extends SL by presuming the presence of a foliage sensor, which allows the asset to avoid spending valuable surveillance time over forested areas, where target sensors are ineffective. The only information that is assumed to be available from this simple foliage sensor is the percentage of the sensor view field occupied by the foliage. Once that percentage exceeds a certain threshold (currently set at 50%), the asset will reverse its direction and will move away from the edge of the forest. Should the asset be initially deployed directly over the foliage, it will continue moving in its initial direction until it leaves the foliage, or an interaction with a world boundary causes it to turn around; in either case, given sufficient time, the asset will end up in a foliage-free location.

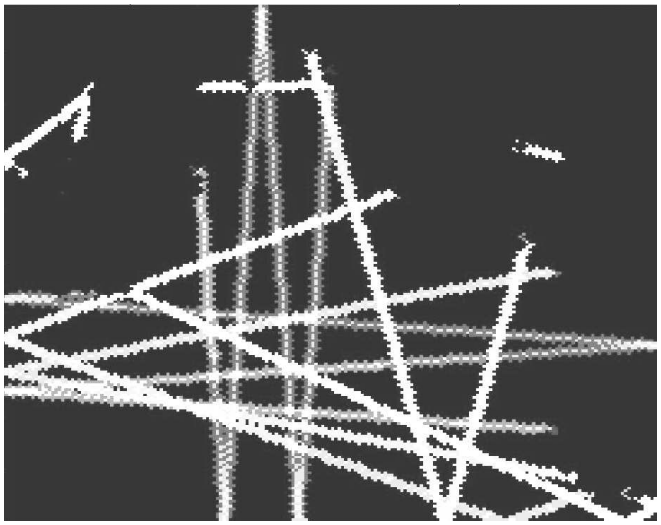

Fig. 4. Sensor coverage with the SLAF strategy after 1,000 steps.

As can be seen in Figure 4, SLAF-operated assets frequently move in a short, localized pattern, and fail to achieve uniform sensor coverage regardless of the amount of time the surveillance activity is carried out.

To address the poor coverage of SLAF, we finally implemented a "Super Straight Line Avoid Forest" (SSLAF) controller that utilizes a more sophisticated foliage sensor. Upon detecting an increased amount of foliage within its sensor (current threshold is set at 10%), the asset will enter an "avoid forest" state, computing the mass distribution of the sensed foliage and moving in the direction exactly opposite the center of foliage mass. Once the detected foliage drops below the threshold, the asset will switch into its normal surveillance state, continuing to move in its current direction until an encounter with another patch of foliage or a bounce from an environment boundary occurs. This strategy was inspired by Reynolds [8] and Balch [2].

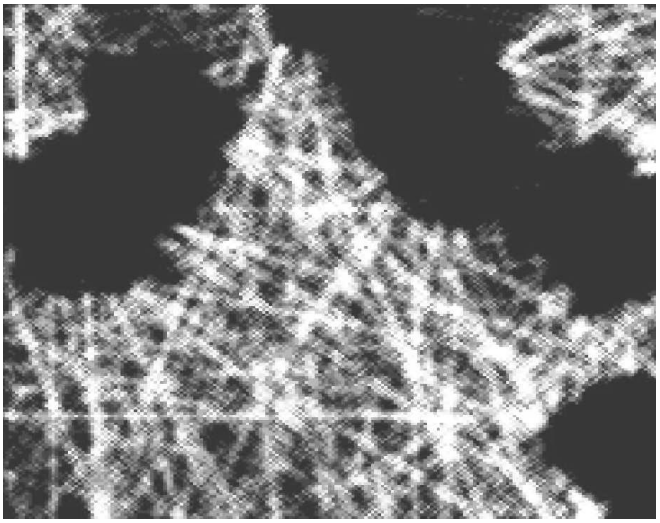

Fig. 5. Sensor coverage with the SSLAF strategy after 1,000 steps.

As can be seen in Figure 5, SSLAF-controlled agents tend to achieve a far more more comprehensive sensor coverage than SLAF, and they achieve this faster than do SL-based assets (as indicated by the brighter coverage of SSLAF versus SL).

## III. Results with Stationary Targets

We first tested the SL, SLAF, and SSLAF asset controllers with environments containing 100 stationary targets. The target radius $r_t = 2$, while the foliage radius $r_f = 5$. Regardless of controller, all assets moved at speed $v = 3$. The percentage of foliage varied from 0% to 70% in increments of 10%. The performance metric is the percentage of targets spotted within $t = 200$ times steps, given that these targets were in fact visible. For each percentage of foliage 100 forests were randomly generated, and for each of the forests the results were averaged over 50 runs with different target and initial asset placements.

Tables I and II give the performance of the three asset controllers, with $\alpha = 10$ and $\alpha = 30$ assets. With 0% foliage, all controllers perform equivalently, since the foliage sensors have no function to perform. Also, the performance of SL

TABLE I

PERCENTAGE OF STATIONARY TARGETS FOUND (10 ASSETS)

| Foliage (%) | SL | SLAF | SSLAF |
|---|---|---|---|
| 0 | 41.18 | 41.18 | 41.18 |
| 10 | 39.05 | 22.47 | 39.87 |
| 20 | 41.25 | 20.09 | 41.11 |
| 30 | 39.25 | 17.43 | 43.12 |
| 40 | 38.98 | 15.83 | 44.53 |
| 50 | 40.67 | 17.73 | 50.19 |
| 60 | 41.35 | 18.19 | 54.57 |
| 70 | 39.98 | 20.13 | 57.58 |

TABLE II

PERCENTAGE OF STATIONARY TARGETS FOUND (30 ASSETS)

| Foliage (%) | SL | SLAF | SSLAF |
|---|---|---|---|
| 0 | 79.52 | 79.52 | 79.52 |
| 10 | 79.71 | 54.30 | 75.73 |
| 20 | 79.33 | 48.07 | 77.67 |
| 30 | 78.14 | 44.06 | 77.63 |
| 40 | 78.54 | 43.01 | 79.55 |
| 50 | 78.86 | 43.78 | 81.25 |
| 60 | 79.07 | 44.22 | 84.05 |
| 70 | 77.27 | 47.46 | 84.49 |

is independent of the percentage of foliage, since it has no foliage sensor. In general, SSLAF outperforms SL, especially as the percentage of foliage increases. This is a consequence of the fact that SL wastes time over areas of foliage. Finally, as expected, SLAF performs poorly, due to its lack of ability to provide uniform coverage of the environment.

## IV. RESULTS WITH MOVING TARGETS

For the next set of experiments we allowed the targets to move according to their "Gollum" controller. Target speed was 0.4, with all other experimental variables fixed as before.

TABLE III

PERCENTAGE OF GOLLUM TARGETS FOUND (10 ASSETS)

| Foliage (%) | SL | SLAF | SSLAF |
|---|---|---|---|
| 0 | 32.28 | 32.28 | 32.28 |
| 10 | 29.78 | 29.63 | 31.35 |
| 20 | 25.46 | 26.50 | 29.04 |
| 30 | 23.09 | 25.60 | 28.47 |
| 40 | 18.59 | 22.72 | 24.02 |
| 50 | 17.45 | 23.10 | 24.85 |
| 60 | 14.54 | 21.93 | 22.39 |
| 70 | 13.61 | 23.11 | 23.10 |

Tables III and IV give the performance of the three asset controllers, with $\alpha = 10$ and $\alpha = 30$ assets. Again, with 0% foliage, all controllers perform equivalently, since the foliage sensors have no function to perform. However, in general, the task is clearly more difficult. This is a consequence of the Gollum controller, where targets actively seek out foliage for cover. As expected, SSLAF still outperforms SL. However, the results with SLAF look anomalous. With moving targets SLAF performs roughly equivalently to SSLAF (and in fact

TABLE IV

PERCENTAGE OF GOLLUM TARGETS FOUND (30 ASSETS)

| Foliage (%) | SL | SLAF | SSLAF |
|---|---|---|---|
| 0 | 65.02 | 65.02 | 65.02 |
| 10 | 64.66 | 64.10 | 65.57 |
| 20 | 55.55 | 56.25 | 57.27 |
| 30 | 51.13 | 54.79 | 54.52 |
| 40 | 45.01 | 51.05 | 50.55 |
| 50 | 39.50 | 48.46 | 47.36 |
| 60 | 36.61 | 48.47 | 46.67 |
| 70 | 30.78 | 45.75 | 40.90 |

outperforms SSLAF when there are a large number of assets and the percentage of foliage is high)!

Recall that our initial motivation for creating the behavior of our asset controllers was predicated on the belief that maximum sensor coverage of the domain implies maximum target detection. For stationary targets, this belief was validated. However, as soon as the targets start to move (with a speed that is quite slow relative to the asset speed), an asset controller with extremely poor domain coverage provides very good target detection. The next section provides a mathematical analysis of why this occurs.

## V. ANALYSIS

Informally, one can start to understand the error in our belief if one notices that the belief was grounded in spatial reasoning. Clearly, for stationary targets, uniform coverage of the space is necessary (if one assumes the targets also are uniformly distributed throughout the space). The asset map shown in Figure 4 clearly indicates that vast portions of the domain are unexplored, leading to poor target detection.

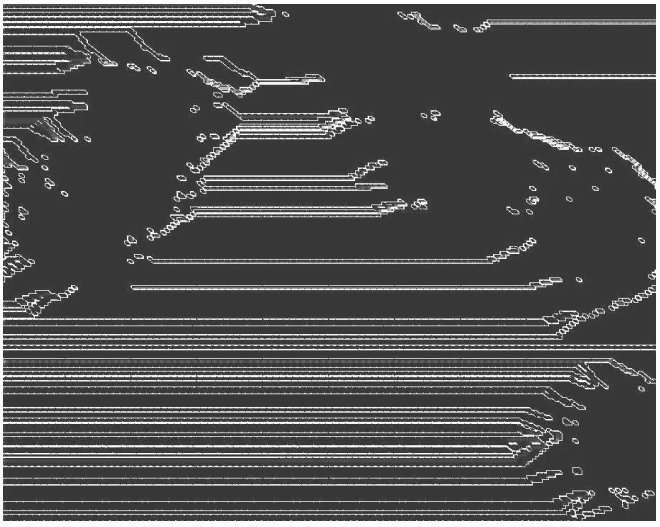

Fig. 6. Gollum target coverage over 1,000 steps.

However, once targets move, spatial reasoning must be combined with temporal reasoning. To illustrate this, Figure 6 shows a "target map" illustrating the movement of the Gollum targets over time, as the targets move from left to right through

the environment. The passage of targets through foliage are not shown, since they are not visible to the assets during that time. One can notice that a clump of foliage provides a "focusing" effect that changes the density of targets (along a column) from uniform to highly non-uniform.

For the sake of target detection, it can be seen that if one mentally overlays Figures 4 and 6, all that is really required for detection is an intersection between the target paths and the asset paths, *at the same moment in time*. Although the target and asset maps provide spatial information, their intersection in time is not represented. For this we will require some probabilistic analysis.

First, for the sake of tractability, we assume that the $L \times L$ environment contains no forest, and that $T$ targets are crossing the environment horizontally from left to right. We consider four situations: (A) an asset is bouncing back and forth in the domain along the horizontal, (B) an asset is bouncing along the vertical, (C) an asset is bouncing along the major diagonal, and (D) an asset is bouncing in such a way as to uniformly cover the space. We will then compute the *expected number of targets* that the asset will detect. Note that the situations are specifically designed to differentiate between a highly uniform coverage of the space versus a highly non-uniform coverage.

### A. Horizontal Movement

This situation is the easiest to analyze. If one asset has target radius $r_t$, then it will sweep a row with height $2r_t$. If $T$ targets are uniformly distributed along the left-most column as they start their movement, then the asset will detect $Tr_t/L$ targets. If there are $\alpha$ assets with non-overlapping horizontal paths, then the expected number of targets detected is:

$$\frac{\alpha T r_t}{L} \tag{1}$$

Note that the velocity of the asset is not relevant for this particular situation. The velocity of the target must be greater than zero.

### B. Vertical Movement

Again, assume the asset has target radius $r_t$. It will sweep a column of width $d = 2r_t$. Assume for the sake of simplicity that the velocity of the asset $v = 2r_t$, so that overlap does not occur. Then the number of steps required for the asset to traverse the column $S_\alpha = L/d - 1$. Thus, any grid point in the column is hit with a probability approximated by $1/S_\alpha$, and is not hit with probability approximated by $(S_\alpha - 1)/S_\alpha$. Finally, we need to calculate how long a *target* will be within that column. If the speed of the target is $v_t$, then the target will require $S_t = d/v_t$ steps to cross the column. Hence the probability of that target not being detected is approximately $((S_\alpha - 1)/S_\alpha)^{S_t}$. Finally, the expected number of targets detected is approximately:

$$T\left[1 - \left(\frac{S_\alpha - 1}{S_\alpha}\right)^{S_t}\right] \tag{2}$$

If there are $\alpha$ independent assets then the number of targets detected is:

$$T\left[1 - \left(\frac{S_\alpha - 1}{S_\alpha}\right)^{\alpha S_t}\right] \tag{3}$$

### C. Diagonal Movement

This situation is a small transformation of vertical movement. Since moving along the diagonal is a factor of $\sqrt{2}$ longer than moving along the column, both $S_\alpha$ and $S_t$ must be renormalized. In this situation $S'_\alpha = \sqrt{2}L/d - 1$ and $S'_t = \sqrt{2}d/v_t$. If there are $\alpha$ independent assets then the number of targets detected is:

$$T\left[1 - \left(\frac{S'_\alpha - 1}{S'_\alpha}\right)^{\alpha S'_t}\right] \tag{4}$$

### D. Uniform coverage

Finally, if an asset is uniformly covering the domain then $S''_\alpha = L^2/\pi {r_t}^2$ and $S''_t = L/v_t$. If there are $\alpha$ independent assets then the number of targets detected is:

$$T\left[1 - \left(\frac{S''_\alpha - 1}{S''_\alpha}\right)^{\alpha S''_t}\right] \tag{5}$$

### E. Theory versus Simulation

For confirmation of the theory, we ran SURVE with 100 Gollum targets, no forest, asset speed $v = 10$, target radius $r_t = 5$, and target speed $v_t = 0.4$. The number of assets were one, two, and four.

TABLE V
EXPECTED NUMBER OF TARGETS DETECTED

| $\alpha$ | Horizontal | Vertical | Diagonal | Uniform |
|---|---|---|---|---|
| 1 | 5 | 74 | 73 | 62 |
| 2 | 10 | 93 | 92.9 | 86 |
| 4 | 20 | 99.6 | 99.5 | 98 |

TABLE VI
ACTUAL NUMBER OF TARGETS DETECTED

| $\alpha$ | Horizontal | Vertical | Diagonal | Uniform |
|---|---|---|---|---|
| 1 | 5 | 73 | 77 | 63 |
| 2 | 10 | 91 | 93 | 86 |
| 4 | 18 | 98 | 99 | 98 |

Table V shows the theoretical results from the above equations, for the four situations. Table VI shows the empirical results, averaged over 1000 runs. The empirical results agree very well with the theoretical results, and are quite informative. First, as might be expected, horizontal movement of the asset yields the poorest performance (assuming that targets are also moving horizontally). However, column movement and diagonal movement definitely outperform uniform coverage! Hence, although uniform coverage is excellent for stationary targets, this strategy is suboptimal for moving targets. If one

re-examines the asset map shown in Figure 4, one can clearly see the long vertical movements that are yielding the good performance of SLAF. The diagonal asset control strategy is especially interesting, since it will work well regardless of whether targets cross the domain horizontally or vertically.

## VI. Summary

This paper presents the SURVE toolkit for experiments in multi-asset surveillance. SURVE has several modules, including forest generation, target controllers, asset controllers, a genetic algorithm, and modules for data collection. Due to the efficiency of implementation, SURVE can be run with hundreds of assets and thousands of targets. We then present evidence that although the goal of uniform coverage of a domain is excellent for detecting stationary targets, it is suboptimal for moving targets. Theoretical analysis confirms the empirical results, indicating that specialized patterns of surveillance will outperform more generalized uniform coverage.

Prior work in this area includes Wu et. al., who used the SAMUEL learning system to evolve rule sets that control a set of micro-air vehicles (MAVs) to provide maximum coverage of a region [13]. Wu expanded on this by combining chunking with a GA to increase performance [14]. Independently, Bugajska combined SAMUEL with a GA to find an optimal sensor suite and reactive rules [3], and combined SAMUEL with ACT-R to provide a cognitive model [4]. Sukhatme et. al. have used a behavior-based approach to surveillance with cooperative aerial and ground vehicles [11]. Albekord et. al. summarize a multi-tiered control strategy for multiple-asset surveillance, utilizing both air and ground vehicles [1]. Spears et. al. discuss a physics-based distributed algorithm for controlling swarms of UAVs for surveillance [9], [10]. Pack and Mullins propose a method for searching an area according to four basic search rules, in order to provide complete coverage of the domain [7]. Finally, Krishna et. al. provide a surveillance system based on multiple mobile sensors [6]. Interestingly, this latter paper uses an "X" formation of assets, where the assets are placed along both major diagonals of the environment. However, these papers do not provide theoretical justification for the behavior of the assets.

We are currently attempting to generalize our theoretical analysis to include forest distributions. The eventual goal is to formally merge the spatial asset and target maps into a temporal construct that is predictive of target detection probability, allowing us to construct optimal asset strategies for given target controllers.

## Acknowledgment

Thanks to Diana F. Spears for suggesting the use of asset and target maps for visualization of the system.

## References

[1] K. Albekord, A. Watkins, G. Wiens, and N. Fitz-Coy, *Multiple-Agent Surveillance Mission with Non-Stationary Obstacles*, Florida Conference on Recent Advances in Robotics, 2004.

[2] T. Balch and R. Arkin, *Behavior-based Formation Control for Multi-Robot Teams*, IEEE Transactions on Robotics and Automation, 1998 (14) 1–15.

[3] M. Bugajska and A. Schultz, *Co-Evolution of Form and Function in the Design of Autonomous Agents: Micro Air Vehicle Project*, Workshop on Evolution of Sensors, Genetic and Evolutionary Computation Conference, 2000, 240–244.

[4] M. Bugajska, A. Schultz, J. Trafton, S. Gittens and F. Mintz, *Building Adaptive Computer Generated Forces: The Effect of Increasing Task Reactivity on Human and Machine Control Abilities*, Genetic and Evolutionary Computation Conference Late Breaking Papers, 2001, 24–29.

[5] D. Goldberg and M. Mataric, *Design and Evaluation of Robust Behavior-Based Controllers*, Chapter 13 of Robot Teams (eds. T. Balch and P. Parker): A.K. Peters, 2002, 369–380.

[6] K. Krishna, H. Hexmoor, P. Rao and S. Chellapa, *A Surveillance System based on Multiple Mobile Sensors*, FLAIRS, Special Track on AI Techniques in Multi-sensor Fusion, 2004.

[7] D. Pack and B. Mullins, *Toward Finding an Universal Search Algorithm for Swarm Robots*, International Conference on Intelligent Robots and Systems, 2003, 1945–1950.

[8] C. Reynolds, *Steering Behaviors for Autonomous Characters*, Proceedings of Game Developers Conference, 1999, 763–782.

[9] W. Spears, D. Spears, R. Heil, W. Kerr and S. Hettiarachchi, *An Overview of Physicomimetics*, Lecture Notes in Computer Science - State of the Art Series, 2005, (3342).

[10] W. Spears, D. Spears, J. Hamann, and R. Heil, *Distributed, Physics-Based Control of Swarms of Vehicles*, Autonomous Robots, Volume 17(2-3), 2004, 137–162.

[11] G. Sukhatme, J. Montgomery, and R. Vaughan, *Experiments with Cooperative Aerial-Ground Robots*, Chapter 12 of Robot Teams (eds. T. Balch and P. Parker): A.K. Peters, 2002, 345–367.

[12] *Tank and Mechanized Infantry Company Team*, Department of the Army Field Manual 71-1, 2002.

[13] A. Wu, A. Schultz, and A. Agah, *Evolving Control for Distributed Micro Air Vehicles*, IEEE Conference on Computational Intelligence in Robotics and Automation, 1999, 174–179.

[14] A. Wu and H. Stringer, *Learning Using Chunking in Evolutionary Algorithms*, University of Central Florida, Computer Science Department, Technical Report, 2002.