Optimal Positioning of Surveillance UGVs†

Ulrik Nilsson, Petter Ögren and Johan Thunberg
Department of Autonomous Systems
Swedish Defence Research Agency (FOI)
164 90, Stockholm, Sweden

Abstract— **Unmanned Ground Vehicles (UGVs) equipped with surveillance cameras present a flexible complement to the numerous stationary sensors being used in security applications today. However, to take full advantage of the flexibility and speed offered by a group of UGV platforms, a fast way to compute desired camera locations that cover or surround a set of buildings e.g., in response to an alarm, is needed.**

In this paper we focus on two problems. The first is how to create a line-of-sight perimeter around a given set of buildings with a minimal number of UGVs. The second problem is how to find UGV positions such that a given set of walls are covered by the cameras while taking constraints in terms of zoom, range, resolution and field of view into account. For the first problem we propose a polynomial time algorithm and for the second problem we extend our previous work to include zoom cameras and furthermore provide a theoretical analysis of the approach itself. A number of examples are presented to illustrate the two algorithms.

I. Introduction

As the market for automated surveillance continues to grow, UGV-mounted cameras are becoming a natural complement to stationary cameras and manned patrolling. The flexibility offered by UGVs is particularly important in cases of alarm response, temporary replacement of stationary cameras, or e.g. when some valuable containers are stored overnight in a large harbor terminal with only perimeter surveillance.

Fig. 1. This Surveillance UGV testbed, developed by Rotundus AB (www.rotundus.se), will be used in real-world experiments.

In this paper we investigate how small scale UGVs, such as the one depicted in Figure 1, can be used in surveillance and security applications. In particular we will address the problem of autonomously finding a set of reachable camera locations that satisfy the requirements for a given surveillance task prompted by e.g. an alarm. We will address two kinds of tasks, the first is how to create a line-of-sight perimeter around a given set of buildings, and the second is how to achieve good stationary coverage of a given set of walls. Before going into details we will give an overview of the previous work in this field.

When studying camera positioning problems in general, the choice of camera model is an important part of the problem statement. While many papers on coverage consider only occlusion constraints, some also deal with explicit range constraints, such as [1]–[5] and others incorporate limited field of view, [2], [3], [6]–[8]. Finally, resolution constraints are dealt with in [2] and [3].

There is a large amount of work done on various aspects of the perimeter surveillance problem, [1], [9], [10]. However, they all assume that the actual perimeter region is already given and that only the sensor placement within some belt remains to be decided. Thus we have found no results on the line-of-sight part of our problem in the literature.

The wall guard problem that we are interested in is very similar to the *The Minimum point guard problem*, that was defined by Eidenbenz, [11], as the problem to find the minimal set of points that guard an area. This problem is known to be NP-hard, [12], and furthermore even finding a set of guards whose cardinality is at most $1+\epsilon$ times the optimum is also NP-hard, [13]. Greedy approximation schemes have been proposed, [14], [11], and analyzed using a transcription to the *so-called Minimum set cover problem*, [11]. These greedy schemes need a set of candidate guard positions to choose from and the choice of such positions have been studied using convex covers, [15]. Other ways of finding candidate positions are proposed in [16], using vertex coloring, and [5], using an approximate *so-called* visibility index. In [11], the tentative observer positions are found by partitioning the whole 3D-space using a huge set of planes, and a similar approach was used in [7] to find areas within which the number of visible vertices do not change.

Alternatives to the greedy approach can be found in [17], where a randomized search approach is proposed instead of a greedy solution and random sampling is furthermore suggested in [18]. Finally, an algorithm for viewpoint computation considering an arm-mounted stereo camera was presented in [3]. The authors propose a number of interesting

† All three authors were funded by the Swedish defence materiel administration (FMV) and the Swedish armed forces through the *Technologies for Autonomous and Intelligent Systems* (TAIS) program, 297316-LB704859.

constraints to be incorporated and perform the optimization using a genetic algorithm.

In this paper, we present a new approach to the line-of-sight perimeter problem as well as extend the ideas presented in [19] on the wall guard problem. The extension includes cameras with zoom capabilities, a better way to handle walls of different lengths, a more compact reformulation of the algorithm itself, and a theoretical result on its properties.

The organization of this paper is as follows. In Section II, we state the two problems and the solutions are proposed in Section III. Examples illustrating the approaches are presented in Section IV. Finally, the paper is concluded in Section V.

II. Problem Formulation

The two problems we investigate both involve finding static camera positions for getting good situational awareness of a scene. The first problem is the line-of-sight perimeter problem and generally requires less UGVs than the second problem, that aims at achieveing concurrent high quality images that cover all walls or buildings selected by the user.

A. Line-of-Sight Perimeter

Problem 2.1: (Minimum line-of-sight perimeter) Let $B = \{b_i : b_i \in \mathbb{R}^2\}$ be a set of points, one located inside each building or item that needs to be surrounded. Let furthermore $O \subset \mathbb{R}^2$ be the union of all obstacles. The problem *minimum line-of-sight perimeter* is the problem of finding a minimum set of points on the ground plane, $S = \{s_i : s_i \in \mathbb{R}^2\}$, such that every point b_i in B lies inside the polygon $\{s_1, \dots, s_m\}$, there is a free line of sight between the pairs (s_i, s_{i+1}), and the distances $||s_i - s_{i+1}|| \leq R$ for some given maximal range R. Above, s_{m+1} is to be interpreted cyclicly as s_1.

An example problem and the corresponding solution can be found in Figure 2.

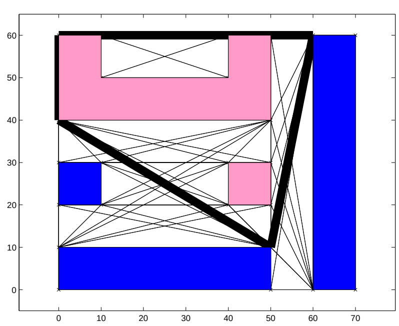

Fig. 2. An example of the solution to a line-of-sight perimeter problem. All buildings are part of the obstacle set O. Setting $B = \{(50, 40), (50, 30)\}$ corresponds to the upper left and the central building being surrounded. The solution perimeter is shown in thick lines.

B. Wall Coverage

The second problem we study is closely related to the *Minimum point guard* problem defined by Eidenbenz in [11]. Here however, we incorporate explicit constraints on camera field of view, as well as image resolution and allow for vehicles equipped with zoom cameras. This problem is motivated by situations where one wants to either make sure that no one exits the surveyed buildings, or when movements in windows needs to be monitored to e.g., look for possible snipers. Formally we define the problem as follows

Problem 2.2: (Minimum wall guard with resolution and field of view constraints) Let $W = \{w_i = (p_i, q_i)\}$ be a set of line segments, $p_i, q_i \in \mathbb{R}^2$, corresponding to the walls that needs to be surveyed. Let furthermore $O \subset \mathbb{R}^2$ be the union of all obstacles. The problem *minimum wall guard* is the problem of finding a minimum set of points on the ground plane, $S = \{s_i : s_i \in \mathbb{R}^2\}$, such that every wall w_i in W is *guarded* by a point s_j in S. By *guarded* we mean that s_j and w_i satisfy the constraints in Definitions 1, 2 and 3.

Definition 1 (Visibility constraint): A wall $w_i = (p_i, q_i) \in W$ is visible to a point guard $s_j \in S$ if the interior of the triangle (s_j, p_i, q_i) does not intersect the interior of the obstacle set O.

Definition 2 (Resolution constraint): Given a camera field of view α and image quality constraints stating that every segment of length δa of the wall being surveyed must cover at least a fraction $k \in (0, 1)$ of the width of the image. A wall $w_i = (p_i, q_i) \in W$ and a point guard $s_j \in S$ satisfies the *resolution constraint* if

$$||r - s_j|| \leq \frac{\delta a \cos(\phi)}{k\alpha} \tag{1}$$

for all points r in w_i, where ϕ is the angle of inclination, i.e. the angle between the the line (r, s_j) and the normal of w_i.
The definition is illustrated in Figure 3, where it can be seen that this constraint will force the camera to be inside two circular arcs.

Remark 1: The above definition is motivated by the fact that computer vision algorithms often need at least some given number of pixels across a given object in order to do recognition with reasonable accuracy.

To handle field of view limitations we make the following definition.

Definition 3 (Field of view constraint): Given a camera view angle limit α, the wall $w_i = (p_i, q_i) \in W$ and a point guard $s_j \in S$ satisfies the *field of view constraint* if

$$\angle(p_i, s_j, q_i) \leq \alpha, \tag{2}$$

where $\angle(p_i, s_j, q_i)$ is the angle between (p_i, s_j) and (q_i, s_j).

This constraint will force the camera to be outside a circle segment, as depicted in Figure 3.

Remark 2: Note that the field of view limitation α in both Definition 2 and Definition 3 can be varied in a zoom camera. This is illustrated for two different α in Figure 3 below.
Having defined the two problems we are trying to solve, we now go on to describe the solutions we propose.

III. Proposed Solution

We first present a polynomial time algorithm solving Problem 2.1 and then an approximation algorithm solving Problem 2.2. Note that in both algorithms, O is the set of obstacles, including buildings, walls to be surveyed and everything else that obstructs vision and mobility.

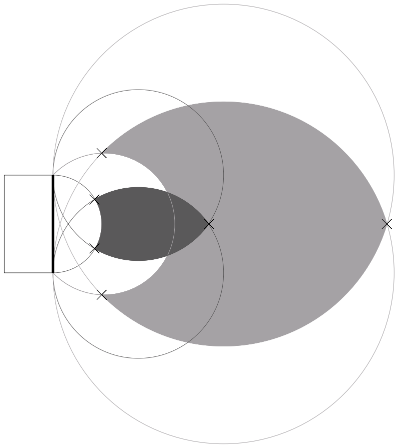

Fig. 3. The sets satisfying Definitions 2 and 3 for two different zoom settings. The dark gray region corresponds to zoom setting $\alpha = 90°$ and the light gray region to zoom setting $\alpha = 45°$.

A. *Line-of-Sight Perimeter*

We begin with stating the proposed algorithm.

Algorithm 3.1:

1) Create a vertex set $V = \{v_i : v_i \in \mathbb{R}^2\}$, either from the corners of the obstacles O, or by any other method, such as random placement or using an equidistant grid.
2) Create a visibility graph $G = (V, E)$, where the edge set $E = \{(v_i, v_j) : ||v_i - v_j|| \leq R, line(v_i, v_j) \cap O = \emptyset, v_i, v_j \in V\}$, i.e., the edges correspond to vertex pairs that have a free line of sight between them of length less than the range R. Assign a unit cost to all edges of G, $c(e) = 1, \ \forall e \in E$.
3) If $B = \{b_i\}$ contains more than one point, remove all edges in $G = (V, E)$ that crosses one of the line segments (b_i^*, b_{i+1}^*). This will ensure that the line-of-sight perimeter encloses all points in B.
4) Pick a random point $\hat{b}$ in B and draw a line to infinity in a random direction. Let $\hat{E} \subset E$ be the edges that intersect this line segment.
5) For each edge $\hat{e}_i = (v_{i1}, v_{i2}) \in \hat{E}$, let L_i be the length of the minimum cost path in $\hat{G} = (V, E \setminus \hat{E})$ from v_{i1} to v_{i2}.
6) Find
$$\min_i (L_i + c(v_{i1}, v_{i2})),$$
this is the minimum number of UGVs needed to create the line-of-sight perimeter. The vertices of the corresponding optimal path are the UGV positions.

Remark 3: For UGVs with two cameras, such as the one in Figure 1, the line-of-sight perimeter can be maintained by positioning a UGV at every other vertex of the path, as noted in the caption of Figure 5.

Remark 4: The above algorithm only finds solutions in the graph G. If the chosen graph is considered too sparse, any number of vertices can be added to the obstacle vertices before creating the visibility edges. More vertices give potentially better solutions at the cost of longer computation times.

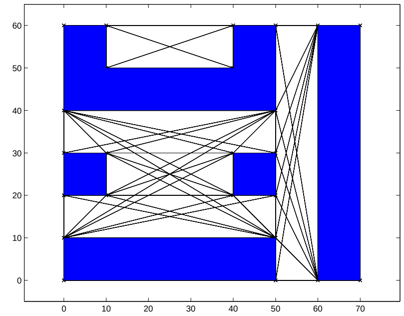

Fig. 4. The visibility graph G, with $R > 100$ created from the obstacle vertices.

Lemma 1 (Algorithm 3.1 is complete and runs in $\mathcal{O}(n^4)$): The algorithm above is polynomial, i.e., it terminates in time $\mathcal{O}(n^4)$, where n is the number of vertices in G. The algorithm is furthermore complete for the one building case, in the sense that if there is a solution to Problem 2.1 in the graph G, the algorithm will always find the optimal one.

Proof:

We begin with showing that the algorithm is polynomial. Creating the visibility graph G involves checking every pair of vertices and is thus $\mathcal{O}(n^2)$. Removing edges in step 3 and 4 involves iterating through the edges, which is also $\mathcal{O}(n^2)$. Step 5 is $\mathcal{O}(n^4)$, since finding a shortest path takes $\mathcal{O}(n^2)$. and this has to be done at most $\mathcal{O}(n^2)$ times. To conclude, the algorithm terminates in $\mathcal{O}(n^4)$.

To show completeness we let $P = (v_1, \ldots, v_n)$ be an optimal solution to problem 2.1. Since P surrounds B, some edge of P must be intersected by the line segment from $\hat{v}$ to infinity and thus be included in $\hat{E}$. The rest of P is then a feasible path when the algorithm iterates through $\hat{E}$ trying to connect v_{i1} and v_{i2} with a shortest path. The algorithm thus finds a path that is at least as good as P, but as P is assumed to be optimal the returned path must be exactly as good as P. Which proves that the algorithm is indeed complete. ■

Remark 5: For the multi building case, there is a small chance that step 4 removes a part of P and then fails to find a solution as good as P in the proof above. Therefore, the algorithm is not complete in the multi-building case.

Remark 6: If the more elaborate edge cost of $c((v_i, v_j)) = 1 + ||v_i - v_j|| / (n^2 \max_{k,l} ||v_k - v_l||)$ is used. The corresponding line-of-sight perimeter is not only the shortest in number of UGVs, but among those, the shortest in actual length.

B. *Wall Coverage*

In this section, we describe an extension of our earlier work presented in [19]. The extension includes zoom cameras, improved handling of walls of different lengths, a more compact formulation, and results on the theoretical properties of the proposed algorithm. The main idea of the algorithm is to carefully construct a set of candidate guard positions, and then choose a subset of these by transcribing the problem to a set cover problem, which in turn is solved by an approximation method.

We propose the following algorithm to find a solution to Problem 2.2.

Algorithm 3.2:

1) Find the candidate guard set S as defined below.
   Definition 4 (The Candidate Set S): Denote by $S_1(w_i) \subset \mathbb{R}^2$ the set of points satisfying the Visibility constraint in Definition 1 for wall w_i and similarly for $S_2(w_i) \subset \mathbb{R}^2$ (Definition 2) and $S_3(w_i) \subset \mathbb{R}^2$ (Definition 3). Let $S \subset \mathbb{R}^2$ be the set of points in $\cup_i(S_1(w_i) \cap S_2(w_i) \cap S_3(w_i))$ that lie on the boundary of at least two of the sets $S_j(w_k)$.
2) Calculate the walls guarded by each $s \in S$, using Definitions 1, 2 and 3. If all walls that can be guarded by a single position do not fit into one single camera view, due to the field of view limitation, this candidate position is duplicated and stored with all possible maximal combinations of guarded walls and corresponding viewing directions. This is repeated for all different zoom settings. Thus to each $s \in S$ a viewing direction $\psi(s)$, a zoom setting $z(s)$, and a set of guarded walls $W(s)$ are assigned.
   At this point it might occur that there are walls in W that are not guarded by any point in S. This happens when the feasible set is empty, i.e., no UGV can survey the whole wall with sufficient image quality. This can be caused by either a wall that is too long, or an obstacle that is too big and close to the wall. Both these problems are solved by dividing the wall $w_i \in W$ into a set of smaller walls. Thus we propose a recursive strategy where a wall that is not guarded by any point in S is divided into two smaller walls. The computations regarding w_i are then re-iterated for both the smaller walls. If again one of the walls is not guarded it is split once more and so on.
3) Select a subset of S guarding all walls in W. Since the problem is NP-hard, we use an approach similar to the greedy one described in [11]. Iteratively, we pick the position $s \in S$ that maximizes the following
   $$\max_s \Sigma_{\mathrm{W'(s)}} f(l, d, \phi),$$
   where $W'(s) \subset W(s)$ are the yet uncovered walls, l is the length of the corresponding wall, d is the distance to the wall, and ϕ is the angle of inclination to a point in the middle of the wall. For the function $f(l, d, \phi)$, we propose the following three options
   $$f(l, d, \phi) = \begin{cases} 1 & \text{or} \\ l & \text{or} \\ l\left(2 - \frac{d}{d_{max}\cos\phi}\right) \end{cases} \qquad (3)$$
   With the first option, the algorithm strives to cover as many walls as possible, as suggested in [19]. With the second option the algorithm instead covers as many meters as possible of the surveyed buildings. We have found that using the more elaborate third choice where d_{max} is the maximum distance from a candidate point in $R_{rf}(w_i)$ to the wall w_i (the diameter of C_{res}), does yield nice solutions. This reward function will favour candidates that are close to the wall and have small angles of inclination by giving a reward close to $2l$. Yet, candidates at less desirable locations will still be rewarded at least the value l. Using the greedy algorithm with this choice of reward function will generally produce attractive tradeoffs between image quality and the number of wall meters covered, as illustrated by Figures 9 and 10 below.

Before we go on to show some examples of guarding problems solved using the algorithm, we analyze the set S in somewhat more detail. Below we will show that for a single zoom setting, restricting the search of guard positions to S, instead of the whole of $\mathbb{R}^2$, yields the same optimal solution. This fact is stated in the following Lemma.

Lemma 2 (Candidate set contains optimal solution): Given a fixed α. If Problem 2.2 is feasible, then the candidate set S contains an optimal solution.

Proof: Let $S^* = \{s_i^*\}$ be an optimal solution to Problem 2.2. It is enough to show that for each $s_i^* \in S^*$ we can find an $s_j \in S$ such that s_i^* and s_j guards the same walls.

Let $W_i^* \subset W$ be the walls guarded by s_i^*, and $S_i^* \subset \mathbb{R}^2$ be the maximal connected region such that all $s \in S_i^*$ guards W_i^*. It is now enough to show that there is an element of S in S_i^*.

S_i^* is bounded by straight occlusion lines resulting from Definition 1 and circular arcs resulting from the resolution and field of view constraints in Definitions 2 and 3. Since S_i^* satisfies Definitions 2 and 3 for all walls in W^*, it is clear that S_i^* is not bounded by a single circle. Thus S_i^* must be bounded by two or more arcs or line segments. The intersections of these arcs and line segments lie in S. Thus an element of S is indeed found within each S_i^*. ■

This proof concludes the algorithm section.

IV. Example Problems

In this section we will apply the proposed algorithms to some example problems.

A. Line-of-Sight Perimeter

Using the visibility graph in Figure 4 we will solve two different problems. If the user wishes to surround the upper left building we can set $B = \{(50, 40)\}$. The corresponding solution can be found in Figure 5. If the central building is also to be surrounded, we set $B = \{(50, 40), (50, 30)\}$ and get the solution in Figure 2. We will now go through the steps in Algorithm 3.1 to see how this result was found. In step 1 the obstacle vertices were used for the set V. In step two we set the max range $R = 100$ and the visibility graph of Figure 4 was created. Then, in step 3 we note that B contains two points, $b_1 = (50, 40)$ and $b_2 = (50, 30)$ and thus we draw a line segment from b_1 to b_2 and remove all edges in E that intersect this segment in order to ensure that both b_1 and b_2 are surrounded by the perimeter. In step 4 we pick a random point in B, say b_2 and draw a line to infinity in a random direction, say to the right in the figure. $\hat{E}$ are now the edges

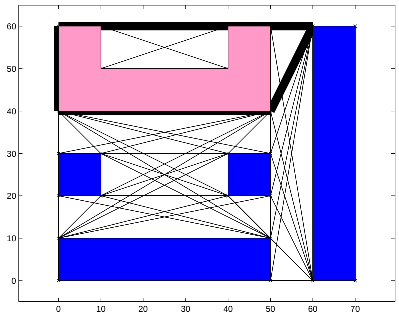

Fig. 5. A four node line-of-sight perimeter surrounding the upper left building. Note that this perimeter can be guarded by either four one-camera UGVs, or by two two-camera UGVs such as the one depicted in Figure 1.

that intersect this segment. In our case $\hat{E}$ are all the edges inbetween the central building and the right-most building, as well as the edges to the right of this building. Now, in step 5, for each of the edges $(v_{i1}, v_{i2}) = \hat{e}_i \in \hat{E}$ we look for the shortest path from v_{i1} to v_{i2} in the graph $\hat{G} = (V, E \backslash \hat{E})$, i.e. the original graph with $\hat{E}$ and all edges between b_1 and b_2 removed. The resulting path must by construction surround all of B. Finally, in step 6 we find for the shortest, in terms of number of nodes, see Remark 6, of these surrounding paths and get the UGV positions shown in Figure 2.

B. Wall Coverage

In this section we will see how the proposed algorithm applies to three examples, one fairly straightforward with two rectangular buildings, another somewhat more difficult problem with one building of complex shape, and finally a problem with four irregularly shaped buildings with a total of 27 walls.

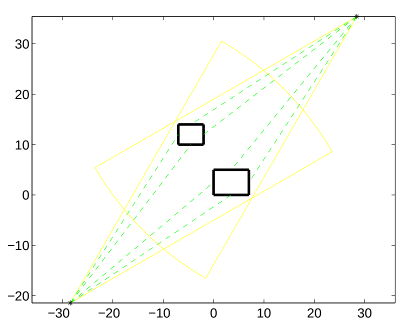

Fig. 6. Two rectangular buildings being surveyed by two UGVs. The dashed green lines illustrates what walls are covered by what UGV while the solid yellow lines denote the field of view cones.

The first example with corresponding solution is depicted in Figure 6. As can be seen the algorithm finds the optimal solution of two UGVs surveying all walls of the two buildings.

The problem becomes slightly more complex if we increase the image quality constraint, in terms of pixels per meter of surveyed wall. This corresponds to increasing the constant k in equation (1). Increasing k reduces the size of the constraint sets, and we get the situation depicted in Figure 7. The constraint sets are shown as well as occlusion lines. Note that the given resolution constraints makes a 2 UGV solution infeasible, since there is no overlap between e.g. the constraint sets of the eastmost wall and the northmost wall. In fact, these two sets are separated by only a few meters near (20,20). The solution with the tighter image quality constraints is further illustrated in Figure 8. This Figure shows the assignment of the different walls to the different guards, as well as the field of view limitations of the cameras.

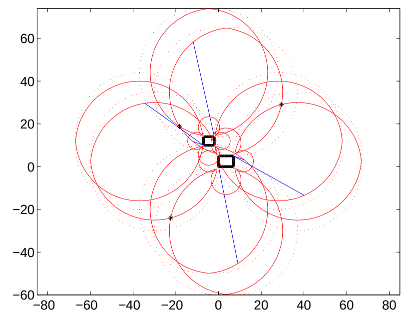

Fig. 7. Two rectangular buildings. The constraint sets are shown in red and the blue lines show occlusion boundaries. The solution involves three guard positions, denoted by asterisks (*). Note that even though the guards at (-22,-24) and (29,29) does indeed see all walls, they do not satisfy the resolution constrains of the west and north walls of the northern building.

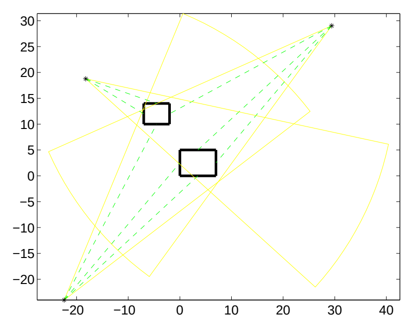

Fig. 8. The detailed solution to the problem of Figure 7. The guards are denoted by asterisks (*) and dashed green lines are drawn between the guards and the walls they guard. Furthermore, the field of view cones are illustrated in yellow.

To illustrate the different choices of objective function proposed in Algorithm 3.2, a problem with only two UGVs and a single house with 20 walls and 140 meters of wall is shown in Figures 9 and 10.

In Figure 9, the objective is to maximize the number of walls being surveyed, i.e. the first option. The corresponding solution focuses on the north part of the building, with 10 walls and 50 meters of wall surveyed. Using the second objective function of equation (3) we get the results in Figure 10. As can be seen, the UGVs guard both sides of the house resulting in a total of 8 walls and 85 meters of wall being surveyed.

In Figure 11, a more complex scenario is depicted. Furthermore, the guards in this problem have three different zoom settings to choose from. Thus it can be possible to fit more walls into the same field of view by moving away while

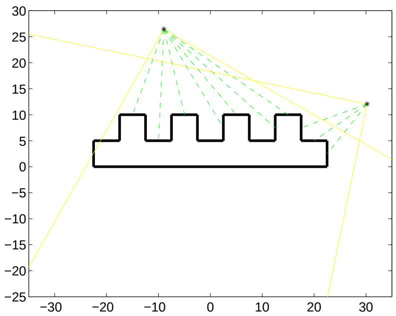

Fig. 9. A building with many wall segments is surveyed by two UGVs. The depicted solution corresponds to maximizing the number of guarded walls.

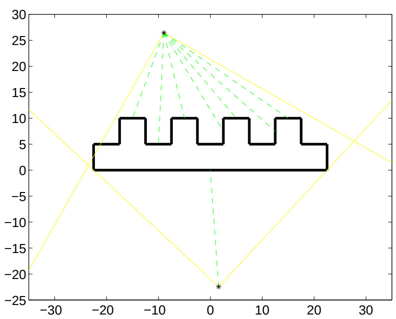

Fig. 10. The same problem as in Figure 9, but with the second objective function of equation (3). The UGV covering the southern wall was placed first.

zooming in to satisfy the resolution constraint, as illustrated in Figure 3. This however increases the risks of running into occlusion problems. As can be seen in the Figure, the guards monitoring the scene from the outside are generally using a higher zoom setting (tele) than the ones viewing the inward facing walls. In this Figure, the solution is obtained using the third row in equation (3). This example concludes the simulation section.

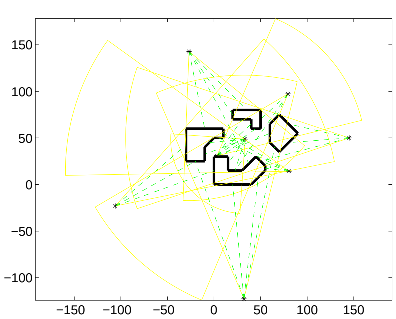

Fig. 11. A complex scenario with 27 walls to be guarded. The solution requires 8 guards to guard all walls while satisfying occlusion, resolution and field of view constraints. As in Figure 8 above, asterisks (*) are guard positions, dashed green lines show what walls are guarded by whom, and yellow cones illustrate field of view limitations.

V. Concluding Remarks

In this paper, two variations on the problem of positioning a team of UGVs to get a good situational awareness were studied, the line-of-sight perimeter problem and the wall guarding problem with realistic camera constraints. For the first we were able to find a polynomial time algorithm that was shown to be complete in certain instances. For the second problem, which is known to be NP-hard, we proposed searching in a carefully selected finite set of candidate positions that was shown to contain an optimal solution to the original problem. Both algorithms were illustrated by example problems.

References

[1] S. Kumar, T. Lai, and A. Arora, "Barrier coverage with wireless sensors," *Wireless Networks*, vol. 13, no. 6, pp. 817–834, 2007.

[2] H. González-Banos and J. Latombe, "A Randomized Art-Gallery Algorithm for Sensor Placement," *Proceedings of the 17th Annual Symposium on Computational Geometry*, pp. 232–240, 2001.

[3] S. Chen and Y. Li, "Automatic Sensor Placement for Model-Based Robot Vision," *Transactions on Systems, Man and Cybernetics, Part B, IEEE*, vol. 34, no. 1, pp. 393–408, 2004.

[4] W. Franklin and C. Ray, "Higher isn't Necessarily Better: Visibility Algorithms and Experiments," *Proceedings of the 6th International Symposium on Spatial Data Handling*, pp. 751–763, 1994.

[5] W. Franklin, "Siting Observers on Terrain," *Symposium on Spatial Data Handling, Ottawa*, pp. 109–120, 2002.

[6] J. Urrutia, "Art gallery and illumination problems," in *Handbook of computational geometry*, J.-R. Sack and J. Urrutia, Eds. North-Holland Publishing Co., 2000, pp. 973–1027.

[7] B. Gerkey, S. Thrun, and G. Gordon, "Visibility-Based Pursuit-Evasion with Limited Field of View," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[8] B. Speckmann and C. Tóth, "Allocating Vertex p-Guards in Simple Polygons via Pseudo-Triangulations," *Discrete and Computational Geometry*, vol. 33, no. 2, pp. 345–364, 2005.

[9] D. Kingston, R. Beard, and D. Casbeer, "Decentralized Perimeter Surveillance Using a Team of UAVs," *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, 2005.

[10] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, *et al.*, "A line in the sand: a wireless sensor network for target detection, classification, and tracking," *Computer Networks*, vol. 46, no. 5, pp. 605–634, 2004.

[11] S. Eidenbenz, "Approximation Algorithms for Terrain Guarding," *Information Processing Letters*, vol. 82, no. 2, pp. 99–105, 2002.

[12] D. Lee and A. Lin, "Computational Complexity of Art Gallery Problems," *IEEE Transactions on Information Theory*, vol. 32, no. 2, pp. 276–282, 1986.

[13] S. Eidenbenz, "Inapproximability Results for Guarding Polygons without Holes," *Proceedings of the 9th International Symposium on Algorithms and Computation*, pp. 427–436, 1998.

[14] Y. Amit, J. S. B. Mitchell, and E. Packer, "Locating Guards for Visibility Coverage of Polygons," in *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments*, ser. Proceedings in Applied Mathematics. SIAM, 2007.

[15] S. Eidenbenz and P. Widmayer, "An Approximation Algorithm for Minimum Convex Cover with Logarithmic Performance Guarantee," *SIAM Journal on Computing*, vol. 32, p. 654, 2003.

[16] M. Marengoni and B. Draper, "System to Place Observers on a Polyhedral Terrain in Polynomial Time," *Image and Vision Computing*, vol. 18, no. 10, pp. 773–780, 2000.

[17] A. Efrat and S. Har-Peled, "Guarding Galleries and Terrains," *Proceedings of the IFIP 17th World Computer Congress-TC1 Stream/2nd IFIP International Conference on Theoretical Computer Science: Foundations of Information Technology in the Era of Networking and Mobile Computing*, pp. 181–192, 2002.

[18] C. Fragoudakis, E. Markou, and S. Zachos, "How to Place Efficiently Guards and Paintings in an Art Gallery," in *Lecture notes in Computer Science: Advances in Informatics*, Bozanis, Panayiotis, Houstis, and Elias, Eds. Springer, 2005, pp. 145–154.

[19] U. Nilsson, P. Ogren, and J. Thunberg, "Towards Optimal UGV Positioning," in *Optimization and Cooperative Control Strategies*, ser. Lecture Notes in Control and Information Sciences, M. Hirsch, C. Commander, P. Pardalos, and R. Murphey, Eds. Springer Verlag, 2008.