Planning Aims for a Network of Horizontal and Overhead Sensors

Erik Halvorson and Ronald Parr
Department of Computer Science
Duke University
{erikh, parr}@cs.duke.edu

Abstract

The ever increasing capabilities and complexity of sensor networks have led to an increased interest in sensor placement and observation planning problems. Many sensor placement and planning problems, however, lead to instances of the intractable classical planning problems or (similarly intractable) partially observable Markov decision processes. We consider the problem of planning sensor actions for a network of overhead sensors which will resolve ambiguities in the output of a horizontal sensor network. More specifically, we address the problem of counting the number of objects detected by the horizontal sensor network, using the overhead network to aim at specific areas to improve the count. The main theme of our results is that, even though sensor planning is intractable for such a network, a simple, greedy algorithm for controlling the overhead sensors guarantees performance with bounded and reasonable suboptimality. Our results are very general and make few assumptions about the specific sensors used. As such, the techniques described in this paper can be used to plan sensor aims for a wide variety of sensor types and counting problems.

1 Introduction

The problems of sensor placement and observation planning have become increasingly relevant as sensor networks increase in both capability and complexity. Often, however, sensor placement and planning problems lead to instances of classical planning problems or partially observable Markov decision processes, both of which are intractable in general. Although there exist algorithms which give optimal solutions to these sorts of problems, the potentially enormous computational burden of using such approaches makes them undesirable.

Consider a horizontal network of sensors with the goal of counting the number of distinct objects it detects. Due to occlusion, the sensor network may not be able to sense all the objects and thus it may not be able to determine the exact count. This paper considers an observation planning problem where the goal is to plan the aims of a set of overhead sensors to resolve these ambiguities. The overhead sensors are used to resolve specific portions in the region of interest where the count is ambiguous. A concrete example of such a network would be a set of horizontal, fixed position cameras, with pan-tilt cameras mounted on unmanned aerial vehicles (UAVs) providing the overhead sensors.

Counting the number of objects within a region is a basic problem in the field of surveillance. Once determined, the number of objects has many potential uses, such as counting people moving across a border, identifying vehicle movements, or providing an accurate count of the people attending a sporting event or other outdoor gathering. Traditional (non computer-based) methods typically rely on manual head counting and would not work in these situations. We consider the problem of developing an accurate count with no human involvement.

Depending on the different kinds of sensors in the network, there are a wide variety of ways to count the distinct objects. This paper will use a geometric approach to counting, inspired by the previous work of Yang, et al. (2003), which used a visual hull to determine upper and lower bounds on the number of people in a scene viewed by horizontal cameras. Though the visual hull is typically associated with cameras, the concept generalizes to other sensor types which can detect occupancy. We also note that the counting method described by Yang, et al. is not specific to counting people and can be used to count any objects detectable by the sensor network.

The work of Yang et al. (2003) assumes that objects move, which helps reduce ambiguities as the patterns of occlusion change over time. Even if the objects are in motion, however, the gap between these bounds may not converge to zero or, depending upon the speed at which the objects are moving, may not converge at an acceptable rate. We consider the use of overhead sensors to supplement the ground network. Such sensors can provide a faster and more accurate count when object motion alone is not sufficient. Overhead sensors, like those found on aircraft, can be redirected in seconds, which makes it safe to assume that in many cases, several iterations of aiming and retargeting of the overhead sensors will be possible before the scene has changed significantly from the perspective of the ground based sensors.

We propose using a simple, greedy algorithm to aim the overhead sensors. Our analysis first bounds the suboptimality over a single set of aims, which we refer to as a *phase*. Since most scenes will require multiple phases, the next portion of our analysis extends these results to multiple phases, bounding the number required relative to an optimal algorithm. We also show that computing an optimal multiphase plan is intractable, and show that two closely related problems are intractable: the subproblem of orienting the overhead sensors to maximize the number of viewed potential objects, and computing the smallest number of objects consistent with a set of observations by the horizontal network.

2 Previous Work

One common approach to the counting problem involves tracking. Multi-target tracking algorithms generally either assume a known, fixed number of targets, or attempt to solve the counting problem while simultaneously tracking the targets. Many approaches to the latter problem attempt to model the arrival and departure of new targets, generally when an unrecognized object is detected (Särkkä, Vehtari, & Lampinen 2007; Zhao & Nevatia 2004). Several appearance-based tracking algorithms have been specifically applied to the problem of counting people (Liu *et al.* 2005; Masoud & Papanikolopoulos 2001). Accurately determining whether a target has been previously detected, however, is non-trivial and error-prone. Counting is itself an interesting problem because it could be used to initialize many multi-target tracking algorithms.

Observation planning approaches to tracking generally assume a known number of targets; He and Chong (2006), for example, formulate the tracking problem as a POMDP and use an approximate solution based on sampling. Guestrin, et al. (2005) develop a greedy approach for the sensor placement problem and bound its suboptimality. We are not aware of any work addressing an observation planning problem that attempts to improve an estimated count.

One very different approach to counting uses a geometric construction called a *visual hull*, which is defined as the intersection of all the silhouette cones seen from each sensor. A *silhouette cone* is a projection of a sensor detection into a conical region in front of the sensor. For example, in the case of camera that has detected a change in the scene that spans several pixels, the corresponding silhouette cone would be a cone extending from the lens into the world that covers all points in the world which project onto the effected pixels. It is possible to reconstruct the geometry of one or more objects by considering the geometry of the silhouette cones as seen from several sensors. Though originally developed for this purpose, visual hulls have been shown to be useful for counting the number of distinct objects detected by a sensor network (Yang, Gonzalez-Banos, & Guibas 2003). The original concept was developed by Laurentini (1994), who designed algorithms for constructing the visual hull in both two and three dimensions. In this paper, we compute a planar projection of the visual hull; this projection results in a number of polygons lying in the plane. Yang, et al, (2003) use these polygons to give lower and upper bounds on the number of objects which create a visual hull, and rely on the objects moving to reduce the gap in bounds. Even with considerable movement, however, the gap between the bounds can remain quite large.

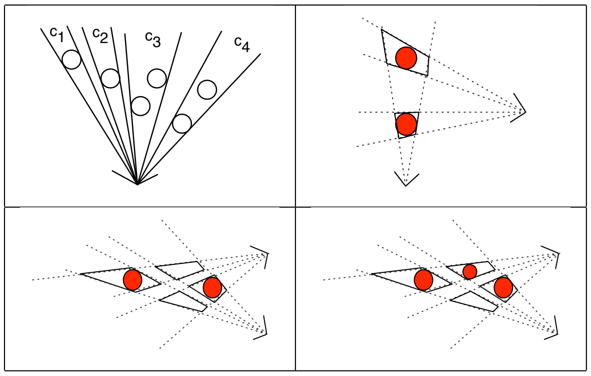


Figure 1: (top left) Example silhouette cones. c_1 and c_2 both contain exactly one object, c_3 contains two objects where the object farther from the sensor is viewed partially occluded, and the rear object is fully occluded in c_4. (top right) A visual hull with two objects. The dashed lines are the silhouette cones, while the solid lines represent the polygons in P. (bottom) Two identical visual hulls created by different numbers of objects. The empty portions of the visual hull are empty because at least one sensor does not detect any objects in that area.

3 Static Bound Calculation

We begin by formalizing the concept of a visual hull. We assume that the horizontal sensors in the network are capable of detecting objects and creating *silhouette cones* (see previous section) where these objects are detected. The sensors are not, however, capable of differentiating distinct objects, so objects lying in the same cone (meaning that they are seen as either fully or partially occluded) do not generate additional silhouette cones; rather, these additional detections appear to be a single, larger occupied region. See Figure 1 (top left) for an example. Given a horizontal network of such sensors, each viewing the same scene from different angles, the silhouette cones can be combined into a *visual hull*:

Definition A planar projection of a *visual hull* is a set of polygons P lying at the intersections of the silhouette cones from each sensor.

Assuming that the entire region of interest is covered by at least two horizontal sensors, all objects in the scene must be located within polygons, although not all polygons necessarily contain objects, as shown in figure 1 (bottom). Note that much of the plane is not in P because at least one of the sensors failed to detect an object in these locations and the location is thus outside the intersection of the cones.

The visual hull can be created by any sensor capable of creating silhouette cones where objects are detected. For example, the cones could be created by applying background subtraction techniques with a camera. The set of polygons, along with the silhouette cones, can be used to develop bounds on the number of objects seen by the network (Yang, Gonzalez-Banos, & Guibas 2003). We first formalize a simple lower bound.

Definition The *cone upper bound* of a cone c, $cub(c)$, is the number of polygons contained in c.

Definition A polygon, generated by intersecting set of cones C, is *provably occupied* if $\exists c \in C$ with $cub(c) = 1$

Consider the examples in Figure 2. In Figure 2 (a), there are two polygons provably occupied (circled). The cone counts are also given. The middle polygon is not provably occupied, however, as both the cones containing it have a cub of 2. Figure 2 (b) has no provably occupied polygons, as all cones have a cone upper bound of two.

The number of provably occupied polygons is a *lower bound* on the number of objects contained by the visual hull. This is a more rigorous definition of the lower bound presented by Yang, et al. (2003). This definition of the lower bound is weak in the sense that the minimum number of objects consistent with the visual hull could be significantly larger, as in Figure 2 (b), where the number of provably occupied polygons is 0, but the number of objects is at least 2. Though weak, this lower bound is *informative*, in that all objects contributing to the bound have a known location in the visual hull.

A simple upper bound can also be derived by assuming a minimum size for the detected objects. We will refer to this minimum size as MINSIZE. Intuitively, such an upper bound would be:

$$UB(P) = \sum_{p \in P} \left\lceil \frac{\text{area}(p)}{\text{MINSIZE}} \right\rceil$$

Additionally, if all objects must be larger than MINSIZE, then polygons smaller than this size can be discarded from the visual hull. Thus every polygon in the visual hull contributes at least one to the upper bound. This is the same upper bound used by Yang et al. (2003). Note that this bound is also weak in the sense that it assumes objects can fill the polygons completely, which could lead to over-estimating the true number of objects inside a single polygon. It may be possible to tighten this bound by making additional assumptions about the geometry of the objects (e.g., circles of at least some radius).

3.1 Hardness Result for Lower Bound

The *optimal lower bound* is a true count of the smallest number of objects that could produce a given visual hull. Based on the definition of the visual hull, one could equivalently define this number as the size of the smallest set of polygons such that each cone contains at least one polygon in the set. This formulation leads to the following decision problem and hardness result.

Definition Given a Visual Hull V and integer k, *LowerBound* decides whether it is possible to produce V with k or fewer objects.

Theorem 1 *LowerBound is NP-Complete.*

Proof The reduction follows from Planar Vertex Cover. Given a planar graph consisting of only straight edges[1] and the vertices in general position[2], fill in the empty regions of the graph with walls. Place a single sensor for each edge in this manner: For the edge (u, v), select either u or v - we will use u for the purposes of this proof. Position a sensor at the chosen vertex looking down the edge towards v. These sensors should be thought of as having a very small field of view. From each sensor, place a cone down the edge, terminating at the wall beyond v. With proper placement of the cones, the only created polygons will be located at the vertices of the graph. See Figure 2 (c, d).

The original graph has a size k vertex cover if and only if this visual hull could have been created by k objects. Since edges in the graph became cones in the visual hull, placing objects in polygons is the same as placing vertices in the cover. Thus, LowerBound can solve Vertex Cover and LowerBound is NP-Hard. Note that LowerBound is also trivially in NP, and thus is NP-Complete. $\square$

This reduction creates a visual hull with many long, narrow passages and intersections at the vertices. Many geometric problems (e.g. Art Gallery (O'Rourke & Supowit 1983)) become more difficult when there are walls inside the area of interest, implying that this result may not generalize to simpler visual hulls. With some additional steps, however, we can transform the visual hull into one with no additional walls and all the sensors on the border of a convex region, though this new visual hull requires potentially several (though still polynomially) more sensors.

Theorem 2 *LowerBound remains NP-Hard, even when the region of interest is a convex polygon and the sensors are all on the boundary.*

Proof Given a planar graph, again with the vertices in general position and drawn with only straight edges, compute the convex hull of the vertices. Create walls outside this convex hull. Next, place the sensors as described in the proof of Theorem 1.

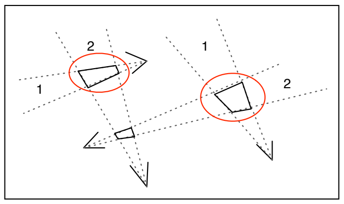
Now, for each sensor placed at vertex u looking towards vertex v (the sensor corresponding to the edge (u, v) in the original graph), move the sensor along the $u - v$ line away from v until it reaches the convex hull. Similarly, extend the cone beyond v until it reaches the other side of the convex hull. While doing this transformation, keep track of any additional intersections introduced; call these additional crossings *unintentional intersections*. See Figure 3 for examples.

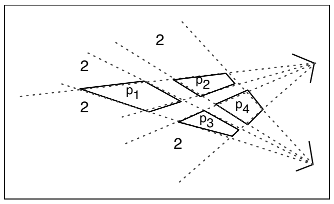
For each unintentional intersection w, create a new sensor on the boundary that looks *only at* w and sees no other intersections, intentional or otherwise, as in Figure 3. Unlike the sensors created in the previous reduction, these new sensors will contribute only *negative* information, meaning that they see no objects in their field of view. This additional information will remove the unintentional intersection w from the visual hull, preventing it from contributing to the bounds. Note that the assumption of general position ensures that such a sensor location can always be found.

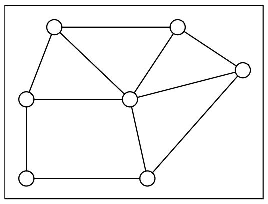
The resulting visual hull will be identical to that created in Theorem 1 and thus will have the same relationship with the original instance of Planar Vertex Cover, but will have no internal walls and all the sensors will be positioned on the boundary of a convex region. Thus, LowerBound remains NP-Hard even when the region is a convex polygon and the

[1]A known fact (Fáry 1948) about Planar Graphs shows that any planar graph can be drawn with only straight edges.

[2]General position in the plane means that no three vertices are on a line.


(a)


(b)


(c)

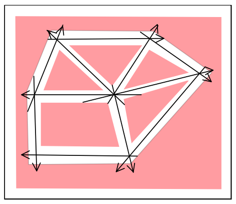

(d)

Figure 2: (left images) Example Visual hulls with (a) two provably occupied polygons (circled) and one ambiguous polygon and (b) no provably occupied polygons. (right images) A planar graph before (c) and after (d) the reduction. The cones in this case have a very small angle, making them basically lines. Note that the polygons occur at the intersections.

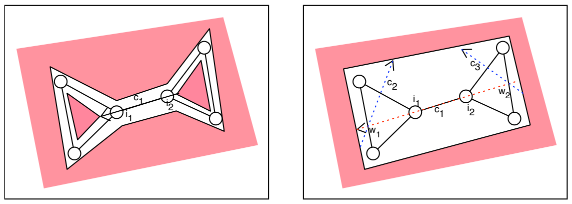


Figure 3: A planar graph after the basic reduction and then after converting to a convex room. To ensure readability, only the cameras pertaining to the central line have been drawn. In the left figure, only the correct polygons (at i_1 and i_2) exist, but removing the central walls and moving c_1 away from i_2 introduces unintentional intersections w_1 and w_2. Adding negative cones c_2 and c_3 removes these unintentional intersections, leaving only the original ones intact.

sensors are all placed on the boundary. Note that the resulting visual hull remains polynomial in size, since there can be at most $O(n^2)$ unintentional intersections, where n is the number of edges in the original graph. □

4 Aim Planning

The general *aim planning* problem involves aiming auxiliary sensors to query the status of portions of the visual hull, reducing the gap between the upper and lower bounds. Since it may not be possible to cover the entire visual hull at once, multiple phases of sensor aiming could be required before all possible information has been extracted from a scene, where a *phase* specifies a single aim for each overhead sensor. The goal of this section is to give an algorithm for planning the aims for the overhead sensors and bound the number of phases required to reduce the gap in bounds, UB - LB, to zero (or the smallest number possible).

Our analysis of the multi-phase aim planning problem is divided into parts. First, we analyze a single sensor aim and the possible suboptimality resulting from a simple aiming strategy. We then consider the subproblem of choosing a set of sensor aims to maximize the number of potential objects viewed, and the suboptimality resulting from a greedy strategy. Finally, we combine these results to address the full, multi-phase aiming problem.

4.1 Overhead Sensor Model

The overhead sensors are be aimed by directing the sensor towards a particular area in the plane. To abstract away from the specifics of the sensor platform, we describe sensor aims by the corresponding area in the plane which is sensed, rather the specific motions or joint angles required to position the sensor.

Definition An *aim* is the area in the plane that is within the field of view of an overhead sensor for some possible configuration of the sensor.

We generally assume that the area covered by an aim corresponds to a ball in some metric space, e.g. the L_∞ ball corresponding to the coverage area of a solid state image sensor that is high overhead.

The overhead sensors behave in a manner similar to the horizontal sensors; they detect occupancy in a conical region extending from the sensor and into the scene. What makes the overhead sensors special is their cones are orthogonal to the plane. Moreover, with the mild assumption that objects are not stacked on top of each other, the overhead sensors are immune to ambiguities from occlusions. each detection by an overhead sensor introduces a new polygon in the plane corresponding to the intersection of the sensor's detection cone with the plane. Figure 4 gives an example of this sensor model, both before viewing and after.

4.2 Bound Tightening

This section proves that no reasonable algorithm can do too poorly at tightening the gap between the bounds. We will use the informative lower bound (LB) and the simple upper bound (UB) defined in Section 3. Before stating the major result of this section, we define a useful property of an aim:

Definition Given an aim v which covers unviewed polygons $p(v)$ in the visual hull, let $C(v)$ be the number of *potential objects* seen by v. More formally:

$$C(v) = \sum_{p \in p(v)} \left\lceil \frac{\text{area}(p \cap v)}{\text{MINSIZE}} \right\rceil$$

$C(v)$ also provides a lower bound on the greatest change in bounds caused by a single aim.

Lemma 3 *Assuming all objects are the same size, choosing aim v will reduce the gap in bounds by* at least $C(v)$*, regardless of the number of objects detected in the aim.*

Proof Suppose the overhead sensor detects a total of k objects. Viewing k objects creates k new polygons, each of size roughly equal to the size of the objects, as in Figure 4. All other area inside the aim will be removed from the visual hull. Since all objects are the same size, the UB decreases by $C(v) - k$ (the area removed from the visual hull) and the LB increases by k, giving a net change of $C(v)$. □

This does not, however, provide an upper bound on the maximum change in the gap between bounds. Consider viewing an empty polygon; after viewing, this polygon will be removed from the visual hull, changing the *cone upper bounds* (see Section 3) for all the cones that it occupied. It is possible that the cone upper bound is reduced to 1 for each of these cones, creating several new provably occupied polygons. We refer to this phenomenon as *inference*. For example, in Figure 2 (b), viewing p_1 could allow us to infer that both the polygons p_2 and p_3 are provably occupied.

To quantify the change in bounds as a result of inference, let $c_{\max}$ be the maximum number of cones per polygon; this quantity, $c_{\max}$, is less than or equal to the number of sensors in the horizontal network, since each polygon can be composed of at most one cone per sensor. Typically, however, the motivation for using overhead sensors will be that the horizontal sensors are sparse enough to create ambiguities. Therefore, it is reasonable to assume that $c_{\max}$ will not be large in practical applications. This definition, along with Lemma 3, is sufficient to derive an approximation ratio for a general class of algorithms.

Theorem 4 *Let A and B be two algorithms for aiming the overhead sensors. Both algorithms choose the same number of* potential objects *to view, meaning that they choose v_A and v_B such that $C(v_A) = C(v_B)$. Let A be an optimal algorithm with respect to the bounds gap, whereas B is any algorithm for choosing an aim of this type. B is a $c_{\max} + 1$ approximation to A.*

Proof Consider the change in bounds for algorithm B. In the worst case, the bounds will change by $C(v_B)$, as demonstrated by Lemma 3; this corresponds with the case where all the polygons are fully occupied. A can, however, potentially change the bounds by as much as $C(v_A) + |p(v_A)| \cdot (c_{\max})$ by seeing only empty polygons and, for each one, inferring that up to $c_{\max}$ other polygons are occupied. This maximum change in bounds for A can be at most $C(v_A) \cdot (c_{\max} + 1)$, since each polygon contributes at least one to $C(v_A)$. Thus, B is a $c_{\max} + 1$ approximation. □

Since this theorem makes no assumptions about B, *any* algorithm for choosing an aim with $C(v_A)$ potential objects would be a $c_{\max}+1$ approximation algorithm. Of course, the number of potential objects viewed by an optimal algorithm is not known *a priori*. If B maximizes the number of potential objects viewed, however, then A cannot view more, and B must be a $c_{\max} + 1$ approximation.

4.3 Maximizing Potential Objects Viewed by Multiple Sensors

This section considers the problem of choosing a set of aims to maximize the number of viewed potential objects. The main result of this section is that a simple, greedy approach yields a constant factor approximation for the largest number of potential objects the overhead network can see. If the overhead sensors have distinct sets of possible aims, then the greedy algorithm is a 2-approximation. If the overhead sensors are *interchangeable* in the sense that all aims are possible for all sensors, then the greedy algorithm is an $\frac{e}{e-1}$ approximation.

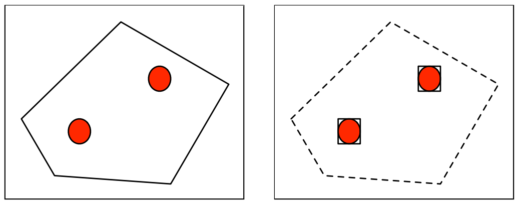


Figure 4: Result of overhead aims. On the left, assume that 5 horizontal cones have produced a polygon which is large enough to contain several objects, in this case 2. The right shows the result of an overhead aim that contains the entire original polygon. Two objects have been detected and polygons corresponding to the intersection of the detection cones with the plane are added, while the rest of the original polygon is removed. The new, square polygons would arise from the detection cones of square image sensor pixels.

```
function polyselect(S) ; S = list of sensors
  if S is empty, stop
  for i:1..size(S)
    mx[i] = maxaim(S[i])
  sstar = argmax(mx[i])
  swap(S[0], S[sstar]);
  mark the view sstar as viewed
  polyselect(S[1..size(S)]);
```

Figure 5: The Polyselect Algorithm

Figure 5 presents the pseudocode for a greedy aiming algorithm called *Polyselect*. Polyselect assumes the existence of a function called *maxaim* that exhaustively considers all possible aims for a sensor and returns the maximum number of new potential objects viewable given the set of aims possible for the sensor. Clearly, there are many opportunities for caching and incremental computation in the implementation of maxaim. Among all sensors for which an aim is not already assigned, Polyselect chooses the sensor and aim that maximizes the number of previously unviewed potential objects. The area chosen by this aim is marked so that subsequent aims do not consider the overlap and the procedure continues until aims are determined for all sensors.

4.4 Non-Interchangeable sensors

Theorem 5 *Polyselect is a 2-approximation of the optimal aim selection procedure.*

Proof Polyselect is a 2-approximation because if it chooses a suboptimal aim, then the potential objects contributing to this suboptimality were previously viewed by Polyselect.

More formally, let $G_1, G_2, \ldots, G_m$ be the total number of previously unviewed potential objects seen by the aims chosen by Polyselect, given in descending order, i.e., the ordering chosen by Polyselect. Now consider the output of an optimal algorithm, $O_1, O_2, \ldots, O_m$, where O_j is the optimal aim for sensor j in the Polyselect ordering. Both quantities are only the *new* potential objects seen by each sensor, meaning that O_k does not count any potential objects counted by $O_{1\ldots k-1}$.

Define the *loss* to be the difference between the number of potential objects viewed by the greedy algorithm and the number viewed by an optimal algorithm. Trivially:

$$\text{loss} = \sum_i (O_i - G_i) \le \sum_i \max\{0, O_i - G_i\}$$

Now consider some $O_j > G_j$, i.e. one of the sensors that contributes to the final summation. For this sensor j, there is an aim viewing a larger number of potential objects than what Polyselect chose, and there are at least $O_j - G_j$ more potential objects at this aim. Since the greedy algorithm chose the aim giving G_j (instead of O_j), however, these additional potential objects must have been covered by sensors Polyselect fixed earlier, and are accounted for in $G_1, G_2, \ldots, G_{j-1}$. Thus, the suboptimality must be bounded by the total number of potential objects seen by Polyselect. More formally,

$$\text{loss} \le \sum_i \max\{0, O_i - G_i\} \le \sum_i G_i$$

Substituting into the original expression for the loss:

$$\sum_i O_i - \sum_i G_i \le \sum_i G_i \Rightarrow \sum_i G_i \ge \frac{1}{2} \sum_i O_i$$

yielding a 2-approximation for the optimal set of aims. □

This approximation ratio is also tight. Consider the scenario in Figure 6 (top). Polyselect will choose to aim the sensors at L_2 and R_2, yielding a total of $n + 1$ potential objects. An optimal algorithm, however, will aim the sensors at L_1 and R_1, with a total of $2n$ potential objects.

4.5 Interchangeable Sensors If the sensors are *interchangeable*, meaning that all aims are possible for all sensors, the greedy algorithm achieves a better approximation ratio. This result draws upon earlier work on maximizing submodular functions. Nemhauser et al. (1978) established several equivalent criteria for a set function z, defined over the subsets of the set A, to be a *submodular non-decreasing function*. We use the following criterion:

$$z(S \cup \{i\}) - z(S) \ge z(T \cup \{i\}) - z(T) \ge 0 \, , \forall S \subset T \subset A, \forall i \in A$$

Lemma 6 *Let A be the set of available aims and $z_C : 2^A \rightarrow$ N be the number of potential objects viewed by a subset of these aims. z_C is a non-decreasing, submodular function.*

Proof Let $S \subset T$ be subsets of A. Now consider adding an additional aim i to both sets. Since z_C counts the number of *distinct* potential objects viewed by a subset of the aims, the additional aim i cannot contribute fewer new potential objects to S than it would to T. z_C is also non-decreasing because adding an aim cannot reduce the number of potential objects. □

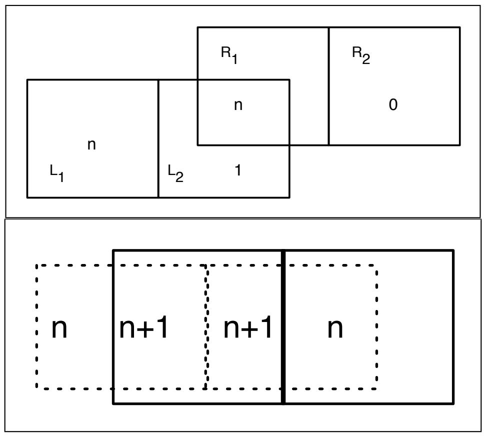


Figure 6: (top) An example demonstrating the tightness of the approximation ratio in theorem 5. There are two aims available to each sensor; for one sensor, the aims available are n and $n + 1$. For the other, n and 0, with the n overlapping with the first sensor. The optimal aims are clearly L_1 and R_1, while Polyselect picks L_2 and R_2. (bottom) An example where Polyselect will give a 4/3-approximation. A sensor can choose to cover any two adjacent sets of potential objects. The optimal aims are the two dashed rectangles, while Polyselect chooses the solid rectangles.

Note that interchangeability is necessary for submodularity. Without interchangeability, z_C is not a set function, as there are some aims which are not available to all the sensors.

Theorem 7 *If the overhead sensors are interchangeable, Polyselect is an $e/(e-1)$-approximation of the optimal aim selection procedure.*

Proof Nemhauser et al. describe a greedy, $e/(e-1)$ approximation algorithm that starts with an empty set and iteratively builds a solution by adding the item i which maximizes $z(S \cup \{i\}) - z(S)$. Polyselect follows the same procedure and is therefore an instance of this algorithm with the objective function z_C. Since z_C is submodular, the $e/(e-1)$ approximation follows as an immediate consequence of the Nemhauser et al. results. □

Figure 6 (top) shows a case where Polyselect with interchangeable sensors yields a 4/3-approximation to the optimal solution. This is the worst case we have devised thus far, suggesting that the bound in Theorem 7 may not be tight.

4.6 Hardness of Maximizing Number of Viewed Potential Objects Could a polynomial time algorithm choose a maximizing set of aims? This section shows that, in general, some form of approximation will be necessary because the basic problem is intractable.

Definition Given a collection S of overhead sensors (with $|S| = c$) and a set P of polygons, MaxObject decides whether there exists a set of aims which allow the sensors in the network to aim at least k potential objects.

Theorem 8 *MaxObject is NP-Complete.*

Proof This problem is NP-Hard so long as the number of overhead sensors is considered part of the input. The reduction follows from the *c-center* problem: *Given a set of points P (with $|P| = n$) in the plane, does there exist a set of c "balls" of radius r which can completely cover all the points in P?* [3]

The *c-center* problem is NP-Complete so long as c is part of the input, even when the metric is L_∞ (Fowler, Paterson, & Tanimoto 1981). Note that "balls" of radius r in L_∞ are axis parallel squares of size $2r$. An instance of the c-center problem can be converted to an instance of MaxObject by creating very small (MINSIZE) polygons for each point, and then creating c sensors which can each view a square of size $2r$. Clearly, an algorithm which can solve this instance of MaxObject can also be used to solve the original instance of c-center. Since MaxObject is trivially a member of NP, it is NP-Complete. $\square$

This theorem demonstrates that finding the aims maximizing the number of viewed potential objects is intractable if the number of overhead sensors is part of the problem input. If the number of sensors is a constant, then finding the aims that maximize this quantity can be solved in polynomial time via exhaustive search since there are $O(n^c)$ possible choices, where n is the number of possible aim points available to each sensor (assuming the set of available aims is discrete). If the set of aims is continuous, however, then solving this problem will require techniques from the field of computational geometry. In either case, the runtime of these procedures can be quite high, even for moderate values of c, making approximation algorithms more practical.

4.7 Multi-phase Bound Resolution

This section considers how Polyselect performs when applied over multiple phases of sensor aims. A *phase* assigns an aim to each sensor and processes the results of the aims, updating the visual hull. In each phase, the network gathers more information about the count in the region. It is assumed that the objects do not move in between phases, a reasonable assumption if the objects are either stationary or moving slowly relative to the speed to the sensors. Many overhead sensors can make multiple aims very quickly, such as pan-tilt cameras mounted on unmanned aerial vehicles. When the overhead network consists of sensors like these, assuming static objects can be reasonable even in the case of moving objects.

The goal of this section is to determine how many greedy phases are required to minimize UB - LB, relative to an optimal algorithm. This problem is particularly interesting because the optimal strategy could be conditional: The selection of a certain aim could depend upon the outcome of earlier aims. This section will use the word *resolve* to mean determining the status of a potential object, either through inference or viewing.

The analysis in this section proceeds in two steps. The first step uses the results from Sections 4.2 and 4.3 to bound the performance of the greedy algorithm over the course of one round, where a round is the length of time an optimal algorithm takes to resolve all the potential objects. This bound leads to a simple recurrence which can then be solved to give an upper bound on the total number of greedy rounds required to minimize the gap between the bounds. This section considers both interchangeable and general sensors.

[3]This problem is generally known as the *p-center* problem.

Lemma 9 *If an optimal algorithm requires k phases (one round) to resolve n potential objects, then Polyselect will view at least $n/(2(c_{\max}+1))$ potential objects in one round with non-interchangeable sensors, and at least $(n(e-1))/(e(c_{\max}+1))$ with interchangeable sensors.*

Proof By Theorem 4, an algorithm that exploits inference can resolve at most a factor of $c_{\max}+1$ more potential objects than an algorithm that doesn't plan to exploit inference. To resolve n potential objects, the optimal algorithm must view at least $n/(c_{\max}+1)$ potential objects. If it is possible to view $n/(c_{\max}+1)$ potential objects, then by Theorem 5, Polyselect will view at least $n/2(c_{\max}+1)$ when they are not interchangeable and at least $(n(e-1))/(e(c_{\max}+1))$ when the sensors are interchangeable. $\square$

Each greedy round reduces the number of unresolved potential objects by a constant factor, leading to a simple recurrence relating the original number of potential objects and the number of phases required to resolve them.

Theorem 10 *Using a greedy d-approximation to plan the sensor aims in each phase requires no more than $d(c_{\max}+1)\log_2 n$ times as many rounds as an optimal algorithm that plans to exploit inference.*

Proof Suppose that after some round i of the greedy algorithm, n_{left} potential objects remain. The same set of aims used by the optimal algorithm will suffice to resolve these n_{left} potential objects. Therefore, by Lemma 9, the greedy algorithm will be able to view at least $n_{left}/d(c_{\max}+1)$ in the next round. Each round, in the worst case, Polyselect cuts the number of remaining potential objects by a constant factor. Letting $a = d(c_{\max}+1)$ be this constant fraction, this reasoning leads to a simple recurrence:

$$T(n) = T\left(\left(1-\frac{1}{a}\right)n\right)+1$$

Solving the recurrence yields:

$$T(n) = \log_{\frac{a}{a-1}} n = \frac{\log_2 n}{\log_2 a - \log_2 (a-1)}$$

The denominator, $\log_2 a - \log_2 (a-1)$, is a finite difference approximation of the derivative of $\log_2$ at a. Since log is concave, this must be *larger* than the true derivative of $\log_2$ at a, $1/a$, implying:

$$T(n) = \frac{\log_2 n}{\log_2 a - \log_2 (a-1)} \leq \frac{\log_2 n}{\frac{1}{a}} = a\log_2 n$$

For $a = d(c_{\max}+1)$, $T(n) \leq d(c_{\max}+1)\log_2 n$. $\square$

Corollary 11 *Polyselect requires at most $2(c_{\max}+1)\log n$ more rounds than an optimal algorithm when using general sensors.*

Corollary 12 *Polyselect requires at most $\frac{e}{e-1}(c_{\max}+1)\log n$ more rounds than an optimal algorithm when using interchangeable sensors.*

4.8 Hardness of Multi-phase Planning

In the previous sections, we demonstrated that applying an approximation algorithm to aim a set of sensors at each phase has bounded suboptimality relative to an optimal planning algorithm. One question remains, however: Could a polynomial time algorithm compute this optimal plan? This section shows that computing such an optimal plan is intractable, even when the number of sensors is fixed.

Definition Given a collection S of overhead sensors (with $|S| = c$) and a set P of polygons, NumPhases is the problem of determining whether it is possible to view P with m phases.

Theorem 13 *NumPhases is NP-Hard, even when the number of sensors is fixed* a priori.

Proof The reduction is from the c-center problem, and follows a similar line of reasoning as used in Theorem 8. Given a set of points P, create a very small polygon for each point; these polygons should be small enough that none overlap. Next, create a single overhead sensor with a square field of view of radius r and position it such that it can aim at any location within the region of interest.

An algorithm to decide this instance of NumPhases will also decide the original instance of c-center. Consider the set of aims chosen by the algorithm deciding NumPhases. These k aims would correspond with k squares (of size $2r$) covering all the points in P, thus also deciding the original decision problem. Therefore, NumPhases is NP-Hard. □

This result is much stronger than the result proved in Section 4.6 as the problem remains NP-Hard even when the number of sensors a constant.

5 Empirical Results

We evaluated our greedy approach using a simulated version of our counting problem with nine horizontal sensors by running Polyselect to completion and measuring the change in bounds over time. To implement `maxaim`, we developed a sweepline approach which finds local maxima in the number of viewed potential objects as the overhead sensor's aim is swept in the y-direction. This sweepline algorithm was then run for a discrete set of x positions (each separated by a constant amount), generating a set of local optima. This set of detected local maxima is then used as a basis for choosing the aims for Polyselect. We compared the performance of Polyselect to a procedure that truly maximizes the area of viewed polygons (*TrueGreedy*) using a brute-force search over all combinations of aims. Both algorithms chose from the same set of aims. The optimal, non-myopic strategy is too expensive to compute because the non-myopic strategy is conditional and could require computing the change in bounds for all possible sequences of aims, as opposed to all possible sequences of just the local maxima. All of the tested configurations had interchangeable sensors.

With two overhead sensors TrueGreedy runs up to $40\times$ slower than PolySelect, and TrueGreedy can be hundreds of times slower with three or more sensors. Figure 7 (top) shows two plots of the bound gap (UB - LB), for TrueGreedy and Polyselect, with various numbers of objects. Figure 7

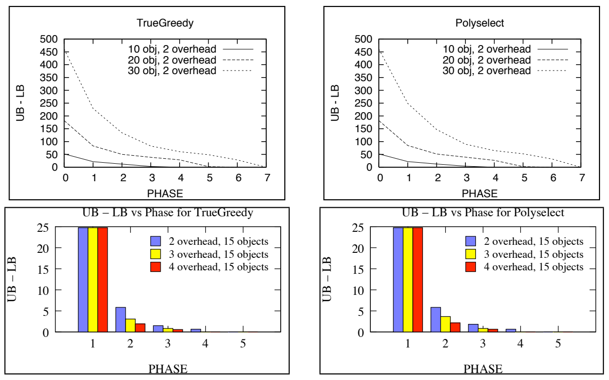


Figure 7: (upper left plot) A plot of the gap in bounds for TrueGreedy vs Phase for two sensors and several different numbers of objects. (upper right plot) A corresponding plot for Polyselect vs Phase on the same runs. (lower left plot) A plot of the gap in bounds for TrueGreedy vs Phase for 15 objects and various numbers of overhead sensors. (lower right plot) A corresponding plot for Polyselect vs. Phase on the same runs. Data were averaged over 15 experiments for the top experiments, and over 12 for the bottom experiments. Note that in both cases, the plots are essentially the same.

(bottom) shows the gap in bounds for TrueGreedy and Polyselect for various numbers of overhead sensors. Note that no more than ten phases were required for any of the experiments. We have noticed empirically that Polyselect is often an excellent approximation algorithm, in many cases choosing equivalent aims to TrueGreedy. Consequently, the suboptimality for both sets of plots in Figure 7 is less than a fraction of an object, even for many sensors. As the graphs demonstrate, the suboptimality of using Polyselect is reasonable.

6 Conclusion

We described a simple, greedy method for planning aims for a set of overhead sensors to resolve an ambiguous count of the number of objects seen by a network of horizontal sensors. We proved that the suboptimality of this approach is both bounded and reasonable. We also demonstrated that solving the sensor aiming problem optimally is intractable.

Acknowledgment

This work was partially supported by the Sloan Foundation, and by NSF IIS award 0209088, NSF CAREER award 0546709, and by the DARPA CSSG program. Any opinions, findings, conclusions or recommendations are those of the authors only. The authors also wish to thank Pankaj Agarwal for many useful suggestions.

References

Fáry, I. 1948. On straight-line representing of planar graphs. *Acta Sci. Math* 11:229–233.

Fowler, R. J.; Paterson, M. S.; and Tanimoto, S. L.

1981. Optimal packing and covering in the plane are NP-Complete. *Information Processing Letters* 12:133–137.

Guestrin, C.; Krause, A.; and Singh, A. 2005. Near-optimal sensor placements in gaussian processes. In *ICML*.

He, Y., and Chong, E. 2006. Sensor scheduling for target tracking: A Monte Carlo sampling approach. *Digital Signal Processing* 16:533–545.

Laurentini, A. 1994. The visual hull concept for silhouette-based image understanding. *IEEE PAMI* 16:150–162.

Liu, X.; Tu, P.; Rittscher, J.; Perera, A.; and Krahnstoever, N. 2005. Detecting and counting people in surveillance applications. In *Advanced Video and Signal Based Surveillance*.

Masoud, O., and Papanikolopoulos, N. 2001. A novel method for tracking and counting pedestrians in real-time using a single camera. *IEEE Transactions on Vehicular Technology* 50(5):1267–1278.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming* 14:265–294.

O'Rourke, J., and Supowit, K. J. 1983. Some NP-Hard polygon decomposition problems. *IEEE Transactions on Information Theory* 29:181–190.

Särkkä, S.; Vehtari, A.; and Lampinen, J. 2007. Rao-blackwellized particle filter for multiple target tracking. *Information Fusion Journal* 8:2–15.

Yang, D.; Gonzalez-Banos, H.; and Guibas, L. 2003. Counting people in crowds with a real-time network of simple image sensors. In *IEEE ICCV*.

Zhao, T., and Nevatia, R. 2004. Tracking multiple humans in crowded environment. *IEEE CVPR* 2:406–413.