Counting Objects with a Combination of Horizontal and Overhead Sensors

Erik Halvorson and Ronald Parr
Department of Computer Science
Duke University
Durham, NC 27708

Abstract

Counting the number of distinct objects within a region is a basic problem in the field of surveillance, with a wide array of possible uses. One approach to this problem involves scanning a wide area and recognizing objects of interest, an approach that can be both computationally intensive and error prone. A recent geometric approach, based upon change detection, can often provide an accurate count of the new objects in a scene without solving the recognition problem. This approach uses a visual hull, which is defined as a set of polygons at the intersection of silhouette cones. The visual hull provides upper and lower bounds on the number of objects in a scene. Often, however, the visual hull is quite ambiguous, meaning these bounds are not tight. Earlier work assumes that these ambiguities will reduce as objects move in the scene.

We consider using a two phase approach to counting objects. In the first phase, a set of horizontal sensors builds a visual hull. We then use a secondary network of overhead sensors to resolve ambiguous regions of the visual hull, tightening the bounds and permitting an exact count. One example of such a setup would be to use a network of ground-based cameras as the initial phase, then to use cameras on unmanned aerial vehicles to tighten the bounds.

We describe several results furthering the understanding of this problem: 1) A hardness result showing that computing a tight lower bound is intractable, 2) A greedy algorithm for maximizing the number of polygons viewed by a network of overhead sensors, 3) A hardness result showing that orienting the overhead sensors to view the maximum number of polygons is intractable, 4) Results showing that a greedy algorithm can be applied to resolve ambiguous portions of the visual hull, with provable bounds relative to an optimal algorithm, and 5) Extensions of these results to two abstracted sensor models for the case when polygons can be occupied by more than one object. Due to the generality of this framework, the algorithms and results apply to counting many different kinds of objects with a wide variety of different sensors.

1 Introduction

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.

Counting the objects in any environment first requires identifying the distinct individual objects and, typically, their locations. These problems are fairly simple for a human to solve, but are often extremely difficult and error-prone when attempted automatically. Moreover, they are often impossible from a single viewpoint, and, as such, require fusing the information from multiple sensors. One previous approach, by Yang, et al.[15], utilizes what is called a *visual hull* to count people with a network of fixed position cameras. The visual hull is a geometric construction for combining the information from multiple viewpoints to recover the geometry of objects in the plane. Since the technique is entirely geometric in nature and requires no recognition, it has a much lower computational burden. Yang, et al., demonstrated that their approach performs well in both simulated and real settings.

The visual hull is not limited to just cameras, however; any network of sensors, visual or not, can create a visual hull so long as they can create *silhouette cones* where they sense objects. For cameras, these cones are typically created by employing background subtraction techniques. An infrared motion detector, however, can also produce a silhouette cone when it detects motion. Since a wide variety of sensors can produce the same construction, we can approach the counting problem more abstractly; instead of worrying about the specific sensors involved, we will treat them as abstract units that sense occupancy. We will continue to refer to this construction as a visual hull, however, to ensure that this work retains its connection to previous results using the concept.

Though the visual hull allows the sensors to be abstracted away, algorithms using the visual hull still have to deal with occlusions. Yang et al. developed a novel way to use the visual hull to develop upper and lower bounds on the number of people in a region [15]. If the upper and lower bounds converge, they converge to the true count. In a static scene the bounds can be quite far apart. If, however, the objects in the scene are moving, then it is possible to prune away empty polygons from the visual hull, resulting in tighter bounds. Yang et al. demonstrate how to exploit the assumption of movement in an efficient way.

In some counting situations, such as counting slow-moving and/or stationary objects,[1] the assumption of constant movement is unrealistic – without a considerable amount of movement, the occlusions will never change and the count

[1] More generally, the difficult case is when the objects are moving slowly relative to the field of view of the sensors and the time scale of the problem. If, for example, a number of vehicles have moved into an area of interest over night, it may be desirable to achieve a count of the number of vehicles before they have a chance to move again.

will remain ambiguous. To solve these problems, we instead propose using a set of secondary, overhead sensors to tighten the bounds. We assume that the overhead sensors can be aimed at portions of the visual hull to sense occupancy. For example, one could use orbital imaging satellites or pan-tilt cameras mounted on unmanned aerial vehicles (UAVs). Again, these sensors are treated abstractly, so the results apply to any network of aimable overhead sensors that can be used to sense occupancy.

The goal is to answer the question of where to aim the overhead sensors to tighten the bounds resulting from the visual hull produced by the ground-based network. We present a simple, greedy strategy for aiming the overhead sensors, as well as complexity results indicating that finding the optimal strategy is intractable. We also describe how this strategy compares to an optimal strategy for aiming the sensors, but we do so in two steps. The first step bounds the suboptimality of the greedy strategy over a single set of aims for the overhead network, which we will refer to as a *phase*. We then extend this result to develop a bound on the number of phases required to recover the true count. Finally, we discuss how these results change when polygons can be occupied by multiple individuals.

2 Previous Work

Counting is not a new problem; this simple problem has been solved for millennia through the use of manual counting by having a person identify and count each individual object. One example of a traditional counting technique (for people) is "checkpoint counting", counting people as they pass through some pre-specified checkpoint, e.g. turnstiles. As stated before, however, there are a large number of situations where such a direct involvement is either impractical or impossible. Consequently, automated techniques have been attempted many times in the past. Many of these techniques have focused specifically on counting people, but much of the work extends to counting other objects as well.

One common approach to this problem involves tracking the objects of interest. Multi-target tracking algorithms generally either assume a fixed, known number of targets, or will attempt to generate a count of the number of targets while simultaneously tracking them. Many approaches to multi-target tracking [6, 14, 16, 9] attempt to model the arrival and departure of new objects, typically when an unrecognized object is detected. Solving this problem is non-trivial, however; to detect new objects accurately, an algorithm will either require constant human involvement or a solution to the data association problem. While there are many proposed solutions to the data association problem [2, 8], they are often not accurate enough to give strong guarantees on the count, nor even to develop upper and lower bounds on the number of individuals. Many tracking approaches also struggle with the occlusion problem, particularly when trying to determine the number of targets [16], making the count even more ambiguous if not impossible to determine. Observation planning approaches to the tracking problem also generally assume a known number of targets. He

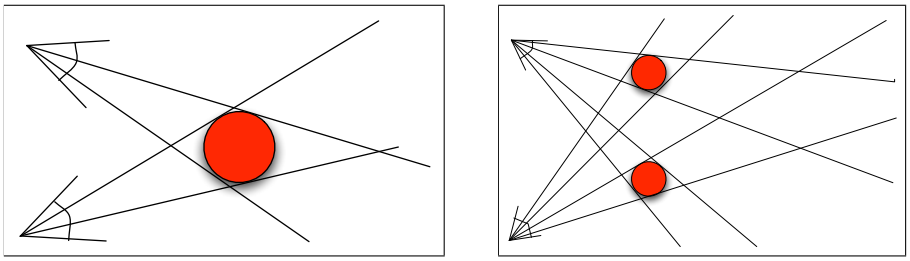

Figure 1: Two examples of a visual hull created with two sensors. Note that there are polygons in the second visual hull which are empty.

and Chong [7], for example, formulate the observation planning problem as a POMDP and use an approximate solution based on sampling. Berry et al. [1] uses a greedy, entropy-based measure to decide where to next aim a sensor to reduce the uncertainty of the position of a target.

The difficulty in using tracking to count objects lies in the problem of recognition. One markedly different approach to counting objects employs a geometric construct called the *visual hull* to obtain the count without solving the recognition problem. The visual hull is defined as the intersection of all cones swept out by the silhouettes of objects viewed from all the sensors, traditionally cameras. Computing the entire visual hull in three dimensions is computationally very expensive [10, 11], but a planar projection of the visual hull often suffices[15].

2.1 Counting using Visual Hulls

Previous work by Yang et al. [15] used a horizontal sensor network to build a planar projection of the visual hull, creating a set of polygonal objects in the plane. These polygons are equal to the intersection of all of the object silhouette cones as seen from each of the fixed location sensors. Each cone corresponds to a detection in a conical, sensitive area in front of the sensor. In the case of a camera, background subtraction is used to identify pixels that have changed from a reference image. Each cone then corresponds to the projection of the identified pixels on the sensor through the lens and into the region of interest. Once these cones have been created, simple techniques from computational geometry can be applied to create the visual hull in polynomial time. More specifically, one can view the visual hull as a subset of an arrangement of lines; using algorithms for computing this arrangement can create the visual hull in $O(n^2)$ time, where $O(n^2)$ is the maximum number of polygons that can result from n cones.

The visual hull only determines where it is possible for there to be an object. Determining the exact count from the visual hull is impossible in many situations, as shown in Figure 1. In the first diagram, there is only a single polygon; since each silhouette cone is created where the sensors view an object, the polygon contains (at least) a single object. In the second diagram, however,

the visual hull is too ambiguous to conclude an exact count; since there are two cones from each viewpoint, there must be at least two objects, but there could be many more. It is often impossible to conclude the exact count from the geometry of the visual hull alone, but the geometry of the cones and polygons can be used to compute upper and lower bounds.

- The **Upper Bound** (UB) is the total area of all the polygons in the visual hull divided by some minimum object size, fixed *a priori*. This loose bound assumes that objects fill polygons as a fluid would. It is possible that a better bound could be obtained by taking advantage of the geometry of the observed objects.
- The **Lower Bound** (LB) is the number of polygons seen by only one visual hull cone. Counting the number of such polygons gives a weak lower bound on the number of objects in the scene.

Yang, et al. [15] use these bounds to count people moving in a room through the use of stationary cameras. The movement of the people allows them to prune polygons, reducing the upper bound and, sometimes, increasing the lower bound. Maintaining these counts as the visual hull changes is a fairly simple process and the bounds give the true count when they converge to a single number. Yang, et al. show that the algorithm converges well for synthetic data and also does reasonably well for real data. The assumptions of Yang, et al. differ from this paper in that this paper attempts to improve the bounds with additional sensors, rather than relying upon the assumption of movement.

3 Static Bound Calculation

3.1 Formal Definition of the Lower Bound

This section defines what it means to conclude (or infer) that a polygon must be occupied. This is a more formal version of the lower bound described by Yang, et al. [15].

Definition The *cone upper bound* of a cone c is the number of polygons the cone contains.

Definition A polygon, composed of a set of cones C, is *provably occupied* if $\exists c \in C$ such that the cone upper bound of c is equal to 1.

See the example in Figure 2 (left). The dark polygons are the only polygons that are provably occupied, while it is impossible to conclude anything about the lighter polygon. The cone upper bounds are also given. Similarly, the right figure 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. In fact, this is just a more rigorous definition of the lower bound presented in the previous section. This new definition

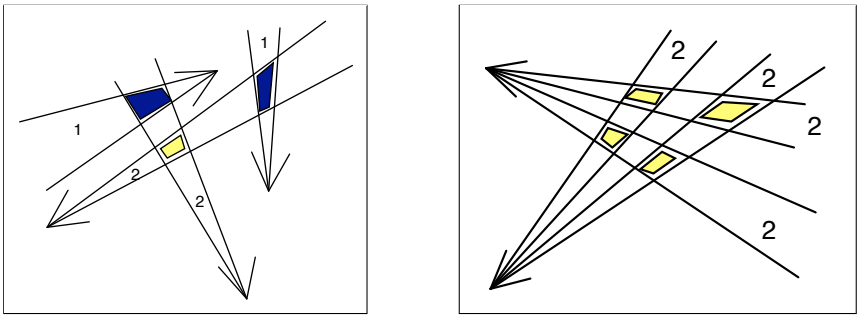

Figure 2: Examples of polygons which are provably occupied (dark) and which are not provably occupied (light). The right image has no provably occupied polygons. The cone upper bounds are also included.

helps to explain why the lower bound is weak; visual hulls with a large amount of occlusion will have many cones with high cone counts. Though this is a very weak lower bound, it does have a useful property. This bound is *informative*, in that all objects contributing to the bound have a known location in the visual hull. Not all lower bounds are informative. See, for example, Figure 2 (right); since there are two cones in the figure, there are at least two objects in the visual hull. Knowing that there are two, however, does not give any information about their positions.

3.2 Hardness Result for Lower Bound

As shown in the previous section, the informative lower bound, LB, may be considerably smaller the minimum number of objects possible in the scene. 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, this optimal lower bound could equivalently be defined 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 a number k, *LowerBound* decides whether it is possible to produce V with k objects (or fewer).

Theorem 3.2.1 *LowerBound is NP-Complete.*

Proof The reduction is from Planar Vertex Cover. Given a planar graph consisting of only straight edges, fill in the empty regions of the graph with walls. Then 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, i.e. towards v. These sensors should be thought of as having a very small field of view. From each sensor,

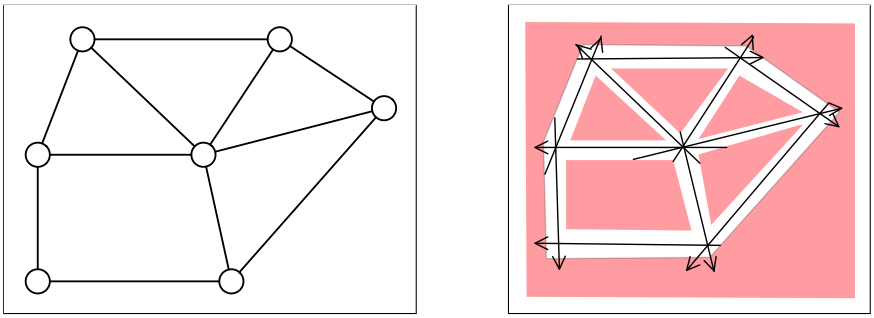

Figure 3: A planar graph before (left) and after (right) the reduction. The cones in this case have a very small angle, making them lines. Note that the polygons occur at the intersections.

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 3 for an example.

The original graph has a vertex cover of size k if and only if this visual hull could have been created by k objects. Since edges in the graph become cones in the visual hull, placing an object in a polygon is the same as placing a vertex in the cover. Thus LowerBound can solve Planar Vertex Cover. Note that LowerBound is trivially a member of NP; given a set of polygons, verifying that each cone contains at least one is simple. Therefore, LowerBound is NP-Complete. □

This argument contains some implicit assumptions. The first assumption is that vertices of the planar graph can be drawn in general position with all edges represented as straight lines. A known fact about planar graphs permits this assumption [3]. Even without this general fact, it is clear from the standard reduction from Vertex Cover to Planar Vertex Cover [5] that there is no loss of generality in this assumption.

A second assumption, or specialization, is the introduction of walls into the problem. Many geometric problems (e.g. Art Gallery [13]) become more difficult when there are walls inside the area of interest. These walls were introduced to maintain the simplicity of the reduction; it is possible to remove the walls, though doing so requires introducing additional sensors to the network. It is, in fact, possible to extend the reduction to include visual hulls contained within a convex region, even when all the sensors are on the border. The procedure works as follows, reducing again from Planar Vertex Cover.

- Compute the convex hull of the planar graph. Create walls outside this convex hull.
- Place the sensors as directed above, i.e, for each edge (u, v), place a sensor at the point u looking towards v. Assume that the field of view of each

sensor is sufficiently narrow that it can be treated as a line.

- For each sensor at a u endpoint, move the sensor along the $u-v$ line away from v. Keep track of any line segments that are crossed on the path to the convex hull. Call these crossings *unintentional intersections*.

- For each unintentional intersection w, create a new sensor that looks *only at* w and sees no other intersections. While the original sensors all provide *positive* information, meaning that they have detected objects in their field of view, the new sensors provide only *negative* information, i.e., they see no objects in their field of view. The negative information prevents unintentional intersections from changing the bounds. One might wonder if it is always possible to place a new sensor that sees only w and nothing else. The assumption of general position guarantees that there is always such a point on the convex hull that sees w and no other intersection points. Thus, one can remove the unintentional intersection points and still have all the sensors on the room boundary.

- The resulting visual hull will contain only polygons corresponding to vertices from the original graph and all sensors are located on the convex hull. This conversion produces an equivalent visual hull to that of the simpler reduction but does not require internal walls. See Figure 4 for before and after pictures.

The visual hull created here remains polynomial in size, as there are only $O(n^2)$ possible intersections on a graph with n edges. Thus the problem of computing the smallest number of objects which could create any given visual hull is intractable, even when the room is convex and all the sensors are on the perimeter.

4 Aim Planning

The general *aim planning* problem involves querying the status of a set of polygons (by aiming an auxiliary sensor) to reduce the gap between the upper and lower bounds. Since it may not be possible to cover all polygons given a fixed number of sensors, 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.

Rather than analyzing the multi-phase aim planning problem monolithically, our analysis is divided into parts. In Section 4.1 we analyze a single camera aim and the possible suboptimality resulting from a simple aiming strategy. In Section 4.2 we consider the subproblem of choosing a set of camera aims to maximize the number of polygons viewed, and the suboptimality resulting from a greedy strategy. Finally, in Section 4.3 we combine these results to address the full, multi-phase aiming problem.

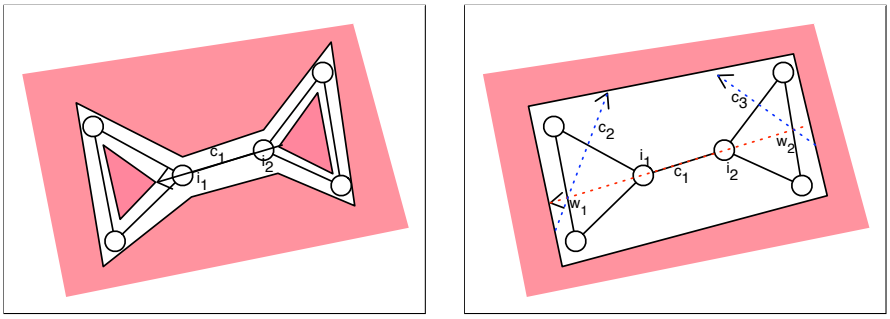

Figure 4: A planar graph after the basic reduction and then after converting to a convex room. To ensure readability, only the cameras pertaining to the central line have been drawn. In the left figure, only the correct polygons (at i_1 and i_2) exist, but removing the central walls and moving c_1 away 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.

4.1 Bound Tightening

In this section, we prove that no reasonable algorithm can do too poorly at tightening the gap between the bounds. Since computing an optimal lower bound is intractable, this section will use the informative lower bound. This section also makes the simplifying assumption that polygons contain at most one object. Section 5.2 extends these results to more general polygons. To recap:

- The Upper Bound (UB) is the number of polygons in the visual hull. This is based on the assumption that polygons are simply occupied or unoccupied – i.e., there are no polygons containing two or more objects.
- The Lower Bound (LB) is the number of provably occupied polygons. This is the same LB discussed in Section 3.1.

The goal is to aim a set of overhead sensors to reduce the difference between the two bounds, UB - LB, to zero (or the smallest number possible) in the smallest number of steps. Note that the number of ambiguous polygons (polygons of unknown occupancy) is exactly UB - LB.

Before considering the full aiming problem, first consider a more abstract version. Suppose there is an oracle that indicates whether a polygon is occupied and that this oracle can be queried about any polygon in the visual hull.[2] In this framework, the bounds always change by at least one: A query either decreases

[2]This is a reasonable abstraction if the overhead sensor is very high above the plane and has a narrow field of view. The sensor could be aimed at any specified polygon without concerns about occlusion. This might not be true if the overhead sensors are not very high above the plane.

the UB by exactly one (on a negative, or unoccupied, answer) or increases the LB by at least one (on a positive answer). A negative answer for p will change the *cone upper bounds* (see Section 3.1) of all cones in which p lies, possibly reducing the cone upper bounds to 1 and making other polygons provably occupied. To bound the number of polygons which can become provably occupied from a negative answer, we assume that there are no more than $c_{\max}$ cones per polygon. This quantity, $c_{\max}$, is bounded above by the number of sensors in the ground-based network, since each sensor can see a given polygon only a single time. It is reasonable to assume that in the cases where overhead sensors are used, $c_{\max}$ will not be large. As $c_{\max}$ increases, and ambiguity in the visual hull decreases and the motivation for using overhead sensors also decreases.

Theorem 4.1.1 *Consider two algorithms, A and B, that can query this oracle. Both algorithms choose the same* number *of polygons to query, k. Assume that A follows an optimal querying strategy, whereas B can be any algorithm to choose k polygons. B yields a* $c_{\max} + 1$ *approximation algorithm to A.*

Proof First consider the change in bounds for B. In the worst case, B will gain information about no polygons aside from the ones it queries, i.e., it either never queries an unoccupied polygon or never reduces a cone upper bound to 1. On the other hand, the best A could do is to reduce $c_{\max}$ cone upper bounds to 1 for *each* polygon it queries, thus reducing the bounds by $c_{\max} + 1$ for each of the k polygons. The total change in bounds is thus k for algorithm B and $(c_{\max} + 1)k$ for algorithm A. B is therefore a $(c_{\max} + 1)$ approximation. □

This theorem demonstrates that *any* algorithm for choosing k polygons would be a $(c_{\max} + 1)$ approximation algorithm. Based on this result, it is safe to say that maximizing the number of polygons queried would be a reasonable approach to this problem. Note that if B employs this strategy, A cannot choose to query more polygons than B and the result holds.

4.2 Maximizing Polygons Viewed by Multiple Sensors

This section considers the problem of choosing a set of aims $\theta_1, \theta_2, \ldots, \theta_m$ to maximize the number of polygons viewed. The main result of this section is that a simple, greedy approach yields a constant factor approximation for the largest number of objects the entire 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 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 polygons 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

```
function polyselect(S) ; S is a 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 polygons sstar can see as viewed
    polyselect(S[1..size(S)]);
```

Figure 5: The Polyselect algorithm.

assigned, Polyselect chooses the sensor and aim that maximizes the number of previously unseen polygons viewed. The newly viewed polygons are removed from the set of viewable polygons and the procedure continues until aims are determined for all sensors.

4.2.1 Maximizing Polygons viewed by non-interchangeable sensors

Theorem 4.2.1 *Polyselect is a 2-approximation of the optimal polygon selection procedure.*

Proof Let $G_1, G_2, \ldots, G_m$ be the total number of previously unviewed polygons 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, $O_1, O_2, \ldots, O_m$, given in the same order (O_j is the optimal aim for sensor j in the greedy ordering). Both quantities are only the *new* polygons seen by each sensor, meaning that O_k does not count any polygons counted by $O_{1\ldots k-1}$.

Define the loss to be the difference between the number of polygons viewed by Polyselect and the number viewed by an optimal algorithm. Trivially:

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

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 polygons than what Polyselect chose, and there are at least $O_j - G_j$ more polygons at this optimal aim. Since Polyselect chose the aim giving G_j (instead of O_j), however, these additional polygons must have been seen by sensors Polyselect fixed earlier, and are accounted for in $G_1, G_2, \ldots, G_{j-1}$. Therefore,

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

Substituting into the original expression for the loss:

$$\sum_i O_i - \sum_i G_i \leq \sum_i G_i$$

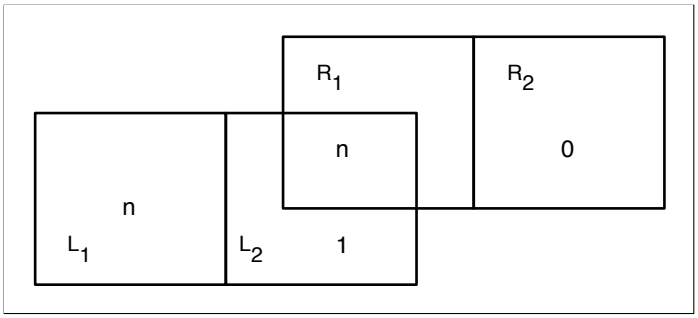

Figure 6: An example demonstrating the tightness of the approximation ratio for non-interchangeable sensors. There are two overhead sensors, each with two different aim points. Polyselect will choose the aims L_2 and R_2, whereas an optimal algorithm would choose L_1 and R_1.

$$\sum_i G_i \geq \frac{1}{2} \sum_i O_i$$

Thus, Polyselect yields a 2-approximation for the optimal set of aims. □

This approximation ratio is also tight. Consider the simplified scenario in Figure 6, in which there are two sensors, each with two possible aims. Polyselect will choose to orient both sensors in position 2, yielding a total of $n+1$ polygons. An optimal algorithm, however, will orient both sensors in position 1, with a total of $2n$ polygons.

4.2.2 Maximizing Polygons viewed by Interchangeable Sensors

If all the sensors can see the same region, and thus can choose from the same set of views, Polyselect achieves a better approximation ratio. This result draws upon earlier work on the maximization of submodular functions [12]. Nemhauser et al. established several equivalent criteria for a function z to be a *submodular non-decreasing function*. We use the following criterion:

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

Lemma 4.2.2 *Let A be the set of views available to the overhead sensors and $z : 2^A \rightarrow \mathbf{N}$ be the number of distinct polygons viewed by a subset of these views. z is a non-decreasing, submodular function.*

Proof Let S and T be subsets of A such that $S \subset T$. Now consider adding an additional aim i to both sets. Since z counts the number of *distinct* polygons viewed by a subset of the aims, the additional aim i cannot contribute fewer

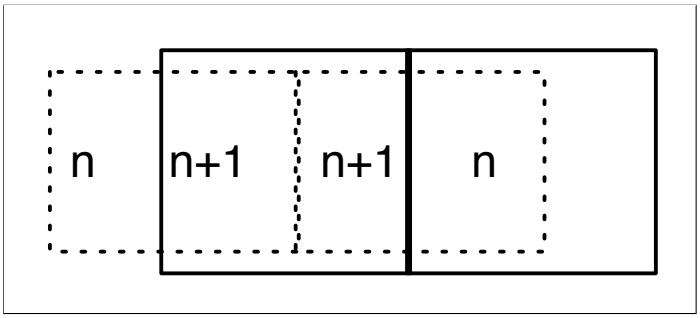

Figure 7: An example where Polyselect will give a 4/3-approximation to the optimal polygon selection. The optimal aims are given by the two dashed rectangles, while the greedy solution is drawn with solid lines. The size of the optimal aims have been shortened to ensure readability.

new polygons to S than it would to T. Also note that adding an aim can only increase the number of distinct polygons, implying that z is non-decreasing. □

Note that interchangeability is necessary for submodularity. If the sensor views are not chosen from the same set, z is not a set function, as there are some views which are not available to all the sensors. Consequently, the results in this section do not apply to the case of non-interchangeable sensors.

Nemhauser et al. describe a *greedy heuristic* [12], for maximizing a submodular set function. Starting with a null set S, the greedy heuristic adds a new item $i \notin S$ that maximally increases $z(S \cup \{i\})$. After k greedy choices, the resulting $z(S)$ is within

$$1 - \left(\frac{k-1}{k}\right)^k$$

of the best possible choice of k elements. In the limit of large k, this becomes $(e-1)/e$.

Theorem 4.2.3 *When all the sensors have the same viewable region, Polyselect gives an $e/(e-1)$-approximation to the optimal procedure for selecting polygons.*

Proof By construction, Polyselect is a greedy heuristic of the form described by Nemhauser et al. Therefore, the number of polygons chosen by Polyselect is within $(e-1)/e$ of optimal, making Polyselect an $e/(e-1)$ approximation. □

The tightness of this bound remains an open question. Figure 7 depicts an example of a situation where Polyselect with interchangeable sensors will give a 4/3-approximation to the optimal solution for two overhead sensors, corresponding to the Nemhauser et al. result for $k = 2$. This pattern can be replicated and Polyselect remains a 4/3 approximation for an arbitrary number of copies. This is better than the limiting case of the Nemhauser et al. result, which would be

$e/(e-1)$. It is not known if more difficult, large instances exist to fill the gap between $e/(e-1)$ and $4/3$.

4.2.3 The Hardness of Maximizing Polygons viewed by Multiple Sensors

One could reasonably ask whether it is necessary to resort to a greedy algorithm (or other approximation) to maximize the number of polygons viewed. 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. MaxPoly is the decision version of this polygon maximization problem. More formally:

Definition Given a collection S of overhead sensors (with $|S| = c$) and a set P of polygons, MaxPoly decides whether there exists a set of aims Θ such that aiming the sensors in these aims will allow the network to view at least k polygons.

Theorem 4.2.4 *MaxPoly is NP-Complete.*

Proof This problem is NP-Hard so long as the number of overhead sensors is not fixed ahead of time. The reduction follows from the *c-center* problem: *Given a set P of n points 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_∞. [4] Note that "balls" in L_∞ are axis parallel squares of size $2r$. The reduction works as follows:

- Given a set of points P, create a polygon for each point. Let each polygon be very small, so none of the polygons overlap.
- Create c overhead sensors. These overhead sensors have a square field of view and are positioned in so that they can view any portion of the plane occupied by the points in P. The field of view of these sensors is also just the size needed to view a radius r square in L_∞, that is, a square of size $2r$.

An algorithm to decide this instance of the MaxPoly decision problem will also decide the original instance of c-center. Suppose there exist aims for the sensors which view at least n polygons. Looking at the squares created by the fields of view of the c sensors would create a set of c squares that contain all n of the input points.

MaxPoly is also trivially a member of the class NP. Given a set of aims for the sensors, it is easy to decide whether the overhead sensors can actually view k polygons. Thus, MaxPoly is NP-Complete. □

[3] This problem is typically known as the *p-center* problem. We refer to it as the *c-center* problem to emphasize the connection between the covering "balls" and the overhead sensors.

This theorem demonstrates that finding the aims maximizing the number of viewed polygons is intractable if the number of overhead sensors is part of the problem input. If the number of sensors is a constant, however, then finding the aims that maximize the number of viewed polygons can be solved in polynomial time via exhaustive search since there are $O(n^c)$ possible choices. For moderate values of c, however, the runtime of this procedure can be quite high.

4.3 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 and set of polygons. In each phase, the network gathers more information about the count in the region. The objective of this section is to determine how many greedy phases are required to minimize UB - LB, relative to an optimal algorithm.

We consider two versions of this problem, one with *inference* and the other without. In this section, inference means using the *cone upper bounds* to prove occupancy, as described in Section 3.1. Bound Resolution without inference requires viewing *all* of the polygons in the visual hull, whereas resolving with inference can conclude that a polygon is occupied without necessarily viewing it. Recall that a polygon can become provably occupied if it lies on a cone for which all other polygons were shown to be empty.

4.3.1 Bound Resolution Without Inference

Suppose there are n polygons to view and that an optimal algorithm is able to view them in k phases, which will henceforth be referred to as a *round*.

Lemma 4.3.1 *If an optimal algorithm requires k phases (one round) to view n polygons, then Polyselect will view at least $n/2$ polygons in one round in general, and at least $\frac{n(e-1)}{e}$ when the cameras are interchangeable.*

Proof Consider an aim planning algorithm that plans aims for a single round by planning aims for k copies of the c overhead sensors. The algorithm could produce a physically impossible sequence of aims, e.g., by tasking a sensor multiple times within a window of c aims. However, since no camera is used more than c times, the aims can be reordered to correspond to a physically possible sequence. Note that the total number of polygons viewed for any strategy does not depend upon the order.

Consider the greedy algorithm from Section 4.2 applied to the problem of aiming k copies of the c sensors. By Theorem 4.2.1 this algorithm is a 2 approximation in general. If the sensors are interchangeable, it is a $\frac{e}{e-1}$ approximation by Theorem 4.2.3. □

In general, assuming a greedy d approximation algorithm is used for aim planning, the number of extra rounds needed is logarithmic in the number of polygons.

Theorem 4.3.2 *If an optimal aim planning algorithm requires k phases (one round) to view n polygons, then a greedy d-approximation algorithm requires at most* $\log_{d/(d-1)} n$ *rounds to view all the polygons and is therefore a* $\log_{d/(d-1)} n$ *approximation.*

Proof Suppose that after some round i of the greedy algorithm, n_{left} polygons remain unviewed. The same set of aims used by the optimal algorithm will suffice to view these n_{left} polygons. Therefore, by Lemma 4.3.1, the greedy algorithm will be able to view at least n_{left}/d in the next round. Each round, in the worst case, the greedy algorithm reduces the number of remaining polygons by a constant factor. This leads to a simple recurrence for the number of rounds required by the greedy algorithm: $T(n) = T(n \cdot (1 - 1/d)) + 1$. Solving the recurrence gives: $T(n) = \log_{d/(d-1)} n$. □

Corollary 4.3.3 *Polyselect requires at most* $\log_2 n$ *more rounds than an optimal algorithm to view all polygons.*

Corollary 4.3.4 *When the sensors are interchangeable, Polyselect requires at most* $\ln n$ *more rounds than an optimal algorithm to view all polygons.*

4.3.2 Bound Resolution With Inference

Consider the general case where it is possible to infer that polygons are occupied in some cases. The problem is now to view, or infer, the status of n polygons in the minimum number of phases. This problem is more interesting than the case without inference because the optimal strategy could be conditional: The decision to view a particular polygon could depend upon the status of earlier views. In this section, we will use the word *resolve* to mean either viewing or inferring the status of a polygon.

Lemma 4.3.5 *If an optimal algorithm requires k phases (one round) to resolve n polygons, then Polyselect will view at least* $\frac{n}{2(c_{\max}+1)}$ *polygons in one round in general, and at least* $\frac{n(e-1)}{e(c_{\max}+1)}$ *when the cameras are interchangeable.*

Proof By Theorem 4.1.1, an algorithm that exploits inference can resolve at most a factor of $c_{\max} + 1$ more polygons than an algorithm that doesn't plan on inference. To resolve n polygons, the optimal algorithm must view at least $n/(c_{\max} + 1)$ polygons. If it is possible to view $n/(c_{\max} + 1)$ polygons, then by Theorem 4.2.1, a greedy algorithm will view at least $n/2(c_{\max} + 1)$ polygons in general and at least $\frac{n(e-1)}{ec_{\max}}$ when the cameras are interchangeable. □

Theorem 4.3.6 *Using a greedy d-approximation to plan the sensor aims in each phase requires no more than* $d(c_{\max}+1)\log_2 n$ *more rounds than an optimal algorithm that plans to use inference.*

Proof Suppose that after some round i of the greedy algorithm, n_{left} polygons remain. The same set of aims used by the optimal algorithm will suffice to

resolve these n_{left} polygons. Therefore, by Lemma 4.3.5, 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, the greedy algorithm cuts the number of remaining polygons by a constant factor. This leads to the same recurrence as Theorem 4.3.2, with $a = d(c_{\max}+1)$:

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

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, $1/a$, implying:

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

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

Corollary 4.3.7 *When the optimal algorithm can plan to use inference, Polyselect (see Section 4.2) requires at most* $2(c_{\max}+1)\log n$ *more rounds than the optimal algorithm to resolve all polygons.*

Corollary 4.3.8 *When the optimal algorithm can plan to use inference and the overhead sensors are interchangeable, Polyselect requires at most* $\frac{e-1}{e}(c_{\max}+1)\log n$ *more rounds than the optimal algorithm to resolve all polygons.*

4.3.3 Hardness of Multi-phase Planning

Theorem 4.3.9 *The multi-phase MaxPoly problem is NP-Hard if the number of sensors is not fixed* a priori.

Proof Section 4.2.3, proves that maximizing the number of viewed polygons for c sensors is NP-Hard so long as c is part of the input. If the number of phases is not determined *a priori*, then any algorithm that solves the multiphase MaxPoly program must also solve the single phase problem to determine if more than a single phase is required. □

It is also possible to extend this result to the case where the number of sensors is fixed *a priori*.

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 4.3.10 *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 4.2.4. 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 4.2.3 as the problem remains NP-Hard even when the number of sensors a constant.

4.4 Complete Approximation Algorithm

The complete algorithm works as follows:

1. Given an initial visual hull with n ambiguous polygons, run an algorithm that maximizes the number of ambiguous polygons seen by all m sensors. This problem is NP-Hard in general (see Theorem 4.2.4), but Section 4.2 provides a 2-approximation in general and an $\frac{e}{e-1}$ approximation when the sensors are interchangeable.

2. Orient the sensors to the positions given by step (1) and update the cone counts. Remove the resolved polygons from the visual hull.

3. Repeat from step one until all polygons are resolved. As proved in Section 4.3.2, this algorithm will not require more than than $2(c_{\max}+1)\log n$ more rounds than an optimal algorithm would require in general, and no more than $\frac{e}{e-1}(c_{\max}+1)\log n$ more if the sensors are interchangeable.

5 Large Polygons

So far, the analysis has assumed that only a single object can be in each polygon, meaning that polygons are simply occupied or unoccupied. Depending upon the range and density of the sensors, however, a visual hull could contain larger polygons which are large enough to contain multiple objects. This section considers how performance changes with the relaxation of the single object per polygon assumption. This assumption is relaxed in two different ways, resulting in slightly different versions of the multi-object detection problem. The first problem is quite similar to the single-object case discussed in Section 4.1; we call this problem *Multiple Occupancy Single Detection*. This problem corresponds with the case where sensors can detect occupied polygons, but cannot distinguish distinct objects within the polygons. The other relaxation leads to

a problem with more powerful sensors; this problem is called *Multiple Occupancy Multiple Detection*, and it corresponds with the case where sensors can distinguish distinct objects within polygons. The same strategies as discussed in Section 4 apply to these new problems, leading to bounds on the number of phases required to fully resolve the number of objects.

5.1 Upper and Lower Bounds

The number of polygons previously served as an upper bound on the number of objects that could create a visual hull, an assumption that no longer holds in the case of large polygons. Following from Yang, et al. [15], it is possible to offer some better bounds for this case:

- The **Lower Bound** remains the number of provably occupied polygons. This provides a loose lower bound on the smallest number of objects in the visual hull.
- The **Upper Bound** is the same as the one used by Yang, et al.: It is the maximum number of objects that could be placed within the visual hull. This could be computed by dividing the total area of all the polygons in the visual hull by some pre-specified minimum object size. The upper bound is also a loose estimate of the maximum number of objects, as it assumes that the objects occupying each polygon can fill it like a liquid [15]. Making further assumptions about the geometry of the objects could yield a tighter bound.

5.2 Single Detection

Recall from Theorem 4.1.1 that, for single occupancy polygons, any reasonable aiming algorithm resolves at least $1/(c_{\max}+1)$ times as many polygons as an optimal algorithm. The proof of this approximation ratio, however, depended crucially on the assumption that each polygon is occupied by at most a single object. This section extends these results to multiply occupied polygons and shows that the previous proof is a special case of this more general framework. We first consider a simple modification of this previous problem.

Definition The *Multiple Occupancy Single Detection (MOSD)* problem is an instance of the aim planning problem in which polygons can be occupied by multiple objects, but overhead sensors return only a single bit of information; they can distinguish between occupied or unoccupied polygons but cannot determine how many objects are within an occupied polygon.

As with the single occupancy case, the objective of the MOSD problem is to reduce the difference between the bounds, $UB - LB$, to the minimum. An optimal algorithm would reduce this gap in the smallest possible number of "steps", where the definition of a single step depends on the method of querying polygons.

First we define a few properties of specific polygons given a minimum object size, MINSIZE.

- The **Lower Score**, $LS(p)$, of a polygon is one for all polygons.
- The **Upper Score**, $US(p)$, of a polygon p is $\lfloor area(p)/\text{MINSIZE} \rfloor$.

Note that the Upper Bound becomes the sum of $US(p)$ for all polygons. Let the quantity $u_{\max}$ be the maximum $US(p)$, i.e., $u_{\max} = \max_p US(p)$.

Mirroring the theoretical development in Section 4.1, assume that there is an oracle which gives information about polygon occupancy. This oracle plays the role of an abstract sensor, meaning that it gives a single bit of information when queried about a polygon.

With this type of oracle, a single 'unoccupied' result reduces the upper bound by $US(p)$ and possibly increases the lower bound by $LS(p)$ for at most $c_{\max}$ other polygons. An 'occupied' result, on the other hand, increases the lower bound by only $LS(p)$ for the queried polygon. This leads to a more general version of Theorem 4.1.1 with these new bounds.

Theorem 5.2.1 *Let A and B be algorithms that query an MOSD oracle about polygon occupancy, with both algorithms querying k polygons. Let A be an optimal algorithm, whereas B can be any other algorithm to choose k polygons. B yields a $(c_{\max} + u_{\max})$ approximation to A.*

Proof Algorithm B will gain information about *at least* k polygons, but possibly no more. On the other hand, Algorithm A gains information about *at most* $\sum_p (US(p) + c_{max})$ polygons. Since $US(p) \leq u_{\max}$, then A gains information about at most $k(u_{\max} + c_{\max})$ polygons. Thus, in this framework, B is a $(u_{\max} + c_{\max})$ approximation to A. □

Recall Theorem 4.1.1 proved that B is a $(c_{\max} + 1)$ approximation. This followed by assuming that each polygon contains at most one object, i.e. $US(p) = 1\ \forall p$ and $u_{\max} = 1$. The MOSD problem is therefore a more general case of the problem defined in Section 4.1 and the results from Theorem 4.3.6 extend to these new bounds.

Corollary 5.2.2 *Using Polyselect to aim sensors for the MOSD problem requires no more than $2(c_{\max} + u_{\max}) \log n$ more rounds than an optimal algorithm.*

Corollary 5.2.3 *Using Polyselect for the MOSD problem with interchangeable sensors requires no more than $\frac{e}{e-1}(c_{\max} + u_{\max}) \log n$ more rounds than an optimal algorithm.*

As stated before, the MOSD problem makes a pessimistic assumption about the inability of the overhead sensors to resolve the individual occupants of the polygons, i.e., if a polygon has more than one object in it, the sensors cannot detect exactly how many. The next subsection addresses a different problem where the sensors are able to detect multiple objects within a single polygon.

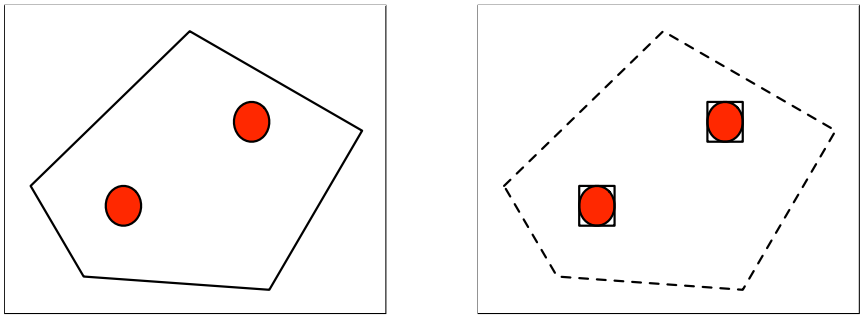

Figure 8: Before and after pictures for sensors in the MOMD problem. The polygon in the left image contains two objects, represented by filled circles. The result of viewing this polygon with a multiple detection sensor is shown on the right. There are two new polygons, each containing exactly one object.

5.3 Multiple Detection

Suppose that the overhead sensors are also capable of creating cones and adding them to the projected visual hull. These cones are perpendicular to the projection plane and, as such, will appear to be provably occupied polygons in the plane. This description leads to a new version of the counting problem:

Definition The *Multiple Occupancy Multiple Detection (MOMD)* problem is an instance of the aim planning problem in which polygons can be occupied by multiple objects and overhead sensors can resolve individual objects within polygons.

For an example of multiple detection sensors, see Figure 8. This section analyzes the worst-case reduction in bounds for an algorithm directing sensors which satisfy the MOMD assumptions. The bounds given here are crucially dependent on the *minimum object size*, as defined in the previous sections.

Definition Given an aim v viewing polygons $p(v)$ in the visual hull, we define $C(v)$ to be the number of *potential objects* seen by v. More formally:

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

This definition gives a lower bound on how much a view can change the bounds.

Lemma 5.3.1 *In the MOMD problem, if all objects are the same size, then choosing aim v and adding the information to the visual hull will reduce the gap in the bounds by* at least $C(v)$*, regardless of the number of objects seen in the view.*

Proof First, suppose that there were no objects detected by aim v. If this is the case, then the upper bound (defined in Section 5.1) will decrease by $C(v)$, because all the polygons that were previously in the field of view of v will be removed from the visual hull. Now suppose the overhead sensor detects a total of k objects. Viewing the k objects creates k new polygons, each of size roughly equal to the size of the objects, as in Figure 8. Since the objects are all of the same size, the upper bound will be decreasing by $C(v) - k$ and the lower bound will increase by k, resulting in a net change of $C(v)$. □

This lemma leads directly to a bound on the worst-case reduction in bounds as compared to an optimal aim selection algorithm.

Theorem 5.3.2 *Consider two algorithms for aiming sensors: A and B. Both algorithms choose the same number of* potential objects *to view, meaning that they chose v_A and v_B such that $C(v_A) = C(v_B)$. Let A be an optimal algorithm, whereas B is any algorithm that can choose a view of this type. For multiple occupancy polygons and the MOMD problem, B is a $c_{\max} + 1$ approximation.*

Proof Consider the change in bounds for algorithm B. In the worst case, the bounds will change by $C(v_B)$; this corresponds with the case where all the polygons are occupied. On the other hand, A can potentially change the bounds by much more. The best that A can do is to change the bounds by $C(v_A) + |p(v_B)| \cdot (c_{\max} + 1)$, which we can upper bound by $C(v_B) \cdot (c_{\max} + 1)$. Thus, B is a $c_{\max} + 1$ approximation. □

As in Section 4.1, this theorem is deliberately vague about the algorithm used for B; based on this theorem, *any* algorithm that chooses a view v seeing at least $C(v_A)$ would be a $c_{\max} + 1$ approximation to the optimal algorithm. Consequently, an algorithm which maximizes this quantity would work well for bounds resolution with this sensor model. Polyselect (see Section 4.2) can be modified to maximize this quantity instead, and will give the same approximation guarantees. Likewise, this problem still remains NP-Hard, as the reduction in Section 4.2.3 corresponds to the case where none of the polygons are larger than a single object.

As in the previous section, the previous multi-phase analysis (see Section 4.3.2) extends to this new sensor model.

Corollary 5.3.3 *For the MOMD problem, Polyselect (approximately maximizing the number of potential objects viewed) will require no more than $2(c_{max} + 1) \log n$ more rounds than an optimal algorithm, where n is the number of potential objects in the original visual hull.*

Corollary 5.3.4 *For the MOMD problem with interchangeable sensors, Polyselect will require no more than $\frac{e}{e-1}(c_{\max} + 1) \log n$ more rounds than an optimal algorithm.*

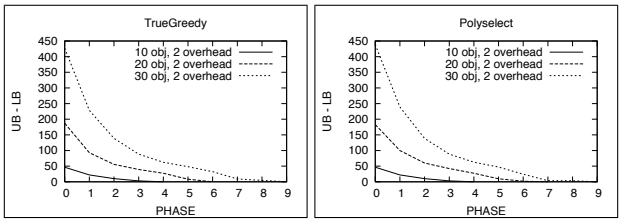

Figure 9: (left) A plot of the gap in bounds for TrueGreedy vs Phase. Data were averaged over 15, 14 and 10 experiments. (right) A plot of the gap in bounds versus phase for Polyselect. Data were averaged over 15 experiments for all lines. Note that both plots are essentially the same.

6 Empirical Results

We evaluated our greedy approach using a simulated version of the counting problem. We generated random configurations of objects with nine horizontal and two overhead sensors. Since many real visual hulls contain large polygons, we did not make the assumption that polygons are only occupied by single objects and used the MOMD sensor model. We then ran Polyselect to completion and measured change in bounds over time. To implement `maxaim`, we developed a sweepline approach which finds local maxima in the viewed polygon area as the overhead sensor's aim is swept in the y-direction. These local maxima were then searched to find the global maximum. We compared the performance of Polyselect to a procedure which actually maximizes the area of viewed polygons, which we call *TrueGreedy*. Both algorithms chose from the same set of aims. The optimal, non-myopic strategy is too expensive to compute, so we do not compare against the truly optimal set of aims. All of the tested configurations had interchangeable sensors.

Polyselect, in general, runs approximately 10-40 times faster than TrueGreedy. Figure 9 shows two plots of the gap in bounds (UB - LB), one running TrueGreedy and the other running Polyselect. As the graphs demonstrate, the suboptimality of using Polyselect is reasonable. Note also that no more than ten phases were required for any of the experiments.

7 Conclusion

This paper described a fairly simple, greedy method for using a combination of horizontal and overhead sensors to count the number of objects in a region. The results are very general in that they apply to any ground-based sensors that can sense occupancy and produce silhouette cones; some examples include cameras, infrared motion detectors, and possibly others. The algorithm presented here assumes only that the overhead sensors can be aimed at various points in the region and can similarly detect occupancy. Moreover, this technique has

provable bounds relative to an optimal algorithm for aiming the same set of sensors. These bounds extend to several different sensor models and special occupancy cases (e.g., multiply occupied polygons). The paper also presented hardness results on the difficulty of finding a tight lower bound and on orienting the overhead sensors to view the largest number of polygons.

8 Acknowledgments

This work was partially supported by the Sloan Foundation, and by the NSF IIS award 0209088. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] P. Berry, C. Pontecorvo, and D. Fogg. Optimal employment of space surveillance resources for maritime target tracking and re-acquisition. In *International Conference on Information Fusion*, pages 719–725, 2003.

[2] T. Darrell, G. Gordon, J. Woodfill, H. Baker, and M. Harville. Robust real-time people tracking in open environments using integrated stereo, color and face detection. *IEEE Workshop on Visual Surveillance*, page 26, 1998.

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

[4] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12:133–137, 1981.

[5] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *ACM Symposium on Theory of Computing*, pages 47–63, 1974.

[6] I. Haritaoglu, D. Harwood, and L. Davis. W^4S: A real-time system for detecting and tracking people in 2 1/2 D. *European Conference on Computer Vision*, 1998.

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

[8] M. Isard and J. MacCormick. BraMBLe: A Bayesian multiple-blob tracker. *IEEE International Conference on Computer Vision*, 2:34–41, 2001.

[9] J. Larocque, J. Reilly, and W. Ng. Particle filters for tracking an unknown number of sources. *IEEE Transactions on Signal Processing*, 50:2926–2937, Dec 2002.

[10] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:150–162, February 1994.

[11] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. *Eurographics Workshop on Rendering*, 2001.

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

[13] J. O'Rourke and K. J. Supowit. Some np-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29:181–190, 1983.

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

[15] D.B. Yang, H.H. Gonzalez-Banos, and L.J. Guibas. Counting people in crowds with a real-time network of simple image sensors. In *IEEE International Conference on Computer Vision*, 2003.

[16] T. Zhao and R. Nevatia. Tracking multiple humans in crowded environment. *IEEE Computer Vision and Pattern Recognition (CVPR '04)*, 2:406–413, 2004.