An Exact Algorithm Optimizing Coverage-Resolution for Automated Satellite Frame Selection

Dezhen Song†, A. Frank van der Stappen‡, Ken Goldberg§
†: IEOR Department, University of California, Berkeley, CA 94720, USA
‡: Institute of Information and Computing Sciences, Utrecht University, PO Box 80089, 3508 TB Utrecht, The Netherlands
§: IEOR and EECS Departments, University of California, Berkeley, CA 94720, USA

***Abstract*— Near Real Time Satellite Imaging provides timely images of the earth for weather prediction, disaster response, search and rescue, surveillance, and defense applications. As the satellite passes over the earth, camera imaging parameters are changed during each time window based on demand for images, specified as user requested zones in the reachable field of view during that time window. The Satellite Frame Selection (SFS) problem is to find the camera frame parameters that maximize reward during each time window. To automate satellite management, we formalize the SFS problem based on a new reward metric that incorporates both image resolution and coverage. For a set of n client requests we give a series of algorithms, the fastest computes optimal results in $O(n^3)$ for satellites with continuously variable resolution. We have implemented the algorithms and compare computation speed for all algorithms.**

I. INTRODUCTION

The first commercially-available high-resolution optical satellite, IKONOS, was launched in 1999 [6]. Since then, satellite imaging has developed into a rapidly growing industry. According to the data from the Imaging and Geospatial Information Society [29], the market is $2.44 billion in 2001 and growing at a rate of fifteen percent annually. Clients include weather prediction, search and rescue, disaster recovery, journalism, and government. Commercial satellites are equipped with sophisticated cameras, which allow them to take high-resolution images as they fly over the Earth. Commercial cameras offer pan, tilt, and zoom (image resolution) control. Near Real Time (NRT) Imaging refers to freshly captured images that are delivered as quickly as possible, depending on the satellite's trajectory: at any given time, the camera's field of view is restricted to a zone on the Earth's surface. During each time window, a number of client requests for images are pending, and only one image can be captured. We consider the problem of automatically selecting pan, tilt, and zoom parameters to capture images that maximize reward.

The Satellite Frame Selection problem is illustrated in Figure 1. We assume the satellite image frame is a rectangle with a fixed aspect ratio. Input is the set of n iso-oriented rectangular regions from users. We propose a reward metric based on how closely a requested viewing zone compares with a candidate satellite image frame. The metric is proportional to the intersection of the candidate frame and the requested viewing zone and to the ratio of the resolution of the candidate and the request. The latter discourages excessively large frames with low resolution. Finding the frame that maximizes total reward is a non-linear optimization problem. Let n be the number of users. For a satellite with continuously variable resolution, we give a series of algorithms, the fastest runs in time $O(n^3)$.

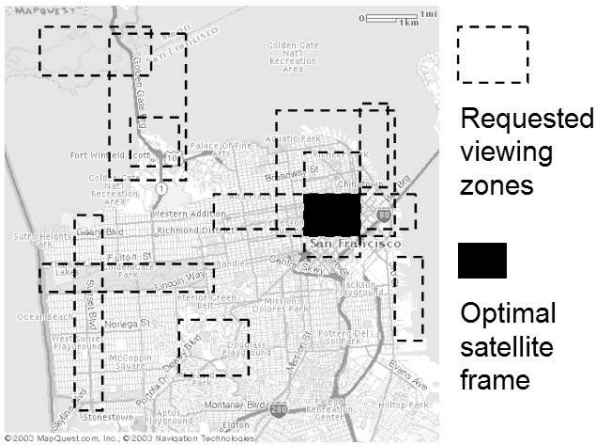

Fig. 1. *The Satellite Frame Selection (SFS) problem: each time window defines the camera's possible field of view. Client requests for images are shown as dashed rectangles. Given a set of requests, the objective is to compute the satellite frame that optimizes the coverage-resolution metric. The solution in this case is illustrated with a solid rectangle.*

II. RELATED WORK

Satellite Frame Selection is related to problems in job scheduling, facility location, spatial databases, videoconferencing and teleoperation.

The Satellite Space Mission problem (SM) [15] is to select and schedule a set of jobs on a satellite. Each candidate job has fixed duration, available time window, and weight. The goal is to select a feasible sequence of that jobs maximizes the sum of weights. This combinatorial optimization problem is known to be NP-hard. Recent research [7], [9], [18], [27] on the SM problem and its variations focuses on developing

This work was supported in part by the National Science Foundation under IIS-0113147, by Intel Corporation, and by UC Berkeley's Center for Information Technology Research in the Interest of Society (CITRIS). For more information please contact dzsong@ieor.berkeley.edu, frankst@cs.uu.nl, or goldberg@ieor.berkeley.edu.

exact and approximate methods using numerical methods such as column generation, Tabu search, and genetic algorithms.

Lemaitre et al. [20] study a related problem for the Earth Observing Satellite (EOS), which has a three-axis robotic camera that can be steered during each time window. Given a set of requested zones, they consider the problem of finding a trajectory for the camera that will maximally cover the requested zones (they do not consider variations in zoom/resolution). Their coverage problem is analogous to planning optimal routes for lawn mowers and vacuum cleaners [10]. Researchers have proposed greedy algorithms, dynamic programming algorithms, and methods based on constraint programming and Local Search. In our model, the time window is shorter and the objective is to servo the camera to a single optimal position with optimal zoom/resolution setting.

The structure of the SFS problem is related to the planar $p-$center problem, which Megiddo and Supowit [23] showed to be NP-complete. Given a set of point demand centers on the plane, the goal is to optimally locate p service centers that will minimize the worst case travel distance between client and server. Using a geometric approach, Eppstein [8] found an algorithm for the the planar 2-Center problem in $O(n \log^2 n)$. Halperin et al. [16] gave an algorithm for the 2-center problem with m obstacles that runs in randomized expected time $O(m \log^2(mn) + mn \log^2 n \log(mn))$.

The SFS problem is also related to "box aggregation" querying in spatial database research [30]. The spatial objects could be points, intervals, or rectangles. Aggregation over points is a special case of the orthogonal range search queries from computational geometry. Agarwal and Erickson [1] provide a review of geometric range searching and related topics. Grossi and Italiano [13], [14] proposed the cross-tree data structure, a generalized version of a balanced tree, to speed up range search queries in high-dimensional space. The continuity of the solution space of our problem makes it impossible to simply evaluate a fixed set of candidate frames through queries.

In the multimedia literature, Kimber and Liu et al. describe a multi-user robot camera for videoconferencing [19], [21]. They formulate frame selection for multiple simultaneous requests as an optimization problem based on position and area of overlap. To solve it, they propose an approximation based on comparing the bounding box of all combinations of user frames. The main concern of their algorithm is speed rather than accuracy. Although they did not provide bounds on their approximation, their approach is sufficient for videoconferencing applications.

Our lab at Berkeley is studying collaborative teleoperation systems where many users share control of a single physical resource. Inputs from each user must be combined to generate a single control stream for the robot. In the taxonomy proposed by Chong et al. [5], these are Multiple Operator Single Robot (MOSR) systems. An Internet-based MOSR system is described by McDonald, Cannon, and colleagues [4], [22]. In their work, several users assist in waste cleanup using Point-and-Direct (PAD) commands. Users point to cleanup locations in a shared image and a robot excavates each location in turn. More recent developments on MOSR systems can be found in [11], [12].

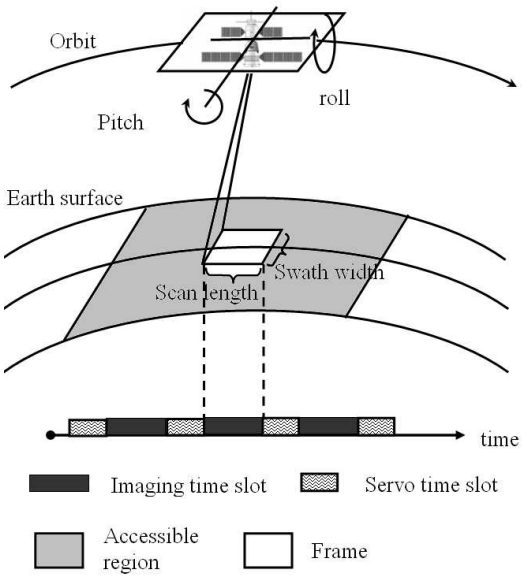

Fig. 2. *Satellite with accessible region, and frame definition.*

The SFS problem is closely related to controlling a shared robotic webcam. We introduced the frame selection problem for robotic webcams in a series of conference papers: exact solution with discrete zoom [26], approximation solution with continuous zoom [24], [25], approximate solution with fixed zoom [17]. This paper presents exact solution with continuous zoom, which is also extending to image requests of any aspect ratio and introducing new reward metric.

III. Problem definition

In this section we formalize the Satellite Frame Selection problem based on a new metric for reward.

A. Input and assumptions

The camera on a typical satellite orbits the Earth at a speed of more than 7km per second. As illustrated in Figure 2, a satellite with two axes allows its reflection mirrors to perform pitch and roll motions, which allow the satellite to view a rectangular region. By rolling and pitching, the satellite can access a square region on the ground. As illustrated in the figure, the imaging time for such satellite is discretized into disjoint time slots. In each time slot, it outputs a rectangular image, which we refer to as a frame. Since most satellites cannot perform yaw rotation, the satellite frame has two of its edges parallel to its orbit.

We assume that the frame is a rectangle with fixed aspect ratio (4:3) and its width is proportional to the resolution. A triple $c = [x, y, z]$ describes such a rectangle: $[x, y] \in R_a$ specifies the center point of the frame with respect to a accessible region R_a, and z specifies the resolution of the frame. The pair x, y determines the pitch and roll angles of the satellite. A $z = 10$ meter means a pixel in the image is equivalent to area of 10×10 square meters. A higher z-value means lower image resolution. The attainable resolution set is Z, so $z \in Z$. For example, a frame has a width that is 1000 times the resolution z and a length that is 1333 times

the resolution z, then the area of the frame is $1000*1333\times z^2$. The width and the length of the frame are linear functions of the resolution, which are defined as $w(z)$ and $l(z)$ respectively.

For a given time slot, we receive n requested view zones from clients. The i^{th} request, $0 \leq i \leq n$, is a rectangle $r_i = [x_i, y_i, w_i, l_i, z_i, u_i]$, where $[x_i, y_i] \in R_a$ specifies center point with respect to the accessible region, w_i, l_i are the width and the length of the requested rectangle, z_i is the desired resolution, and u_i is the utility for the request, which describes how much the client is willing to pay for the requested view zone. This is also the maximum reward associated with this request. We assume that all requested viewing zones are iso-oriented rectangles with a pair of edges parallel to satellite orbit.

Given a set of n requested viewing zones, we must compute a single frame c^* that will yield maximum total reward for the company. The solution space is

$$\Phi = R_a \times Z = \{[x, y, z] | [x, y] \in R_a, z \in Z\}.$$

Set $Z = [\underline{z}, \bar{z}]$ is a continuous set.

B. Reward Metric

Recall that r_i is the i^{th} requested viewing zone. Its corresponding client has a utility u_i for this region. Define s_i as the reward from the i^{th} request. Let $c = [x, y, z]$ be a candidate camera frame. If the r_i is fully covered by c, i.e., $r_i \subseteq c$, and the desired resolution is obtained, i.e., $z_i \geq z$, then $s_i = u_i$. If the resolution requirement is satisfied but the coverage is partial, then the reward is discounted by a coverage ratio: $s_i = u_i \frac{Area(r_i \cap c)}{Area(r_i)}$. If $z_i < z$ (the resolution requirement is not satisfied) then the reward should be discounted by a resolution discount factor $d(z, z_i)$. Hence, $s_i = u_i \frac{Area(r_i \cap c)}{Area(r_i)} d(z, z_i)$. As illustrated in Figure 3(a), the resolution discount function $d(z, z_i)$ is a truncated function: $0 \leq d(z, z_i) \leq 1$. It is an increasing function of z_i/z because an image has more value as resolution increases. The resolution discount function we propose is

$$d(z, z_i) = min\{(z_i/z)^b, 1\}.$$

Let $Resolution(r_i) = z_i$ and $Resolution(c) = z$, then our reward function is a Coverage-Resolution Ratio (CRR),

$$s_i(c) = u_i \frac{Area(r_i \cap c)}{Area(r_i)} \min\Big(\big(\frac{Resolution(r_i)}{Resolution(c)}\big)^b, 1 \Big) \quad (1)$$

The exponential discount factor b determines how fast the frame image devalues as its resolution decreases. Figure 3(b) shows two cases: $b = 1$ and $b = \infty$. The case is $b = \infty$ corresponds to a scenario in which the user does not accept any image with a resolution that is lower than requested. We use the case $b = 1$ as default setting for numerical examples in the rest of the paper.

For n requests, the total reward is,

$$s(c) = \sum_{i=1}^{n} s_i(r_i, c). \quad (2)$$

We want to find $c^* = \arg\max_c s(c)$, the frame that maximizes total reward. We will often write $s(x, y, z)$ instead of $s(c)$ with $c = [x, y, z]$.

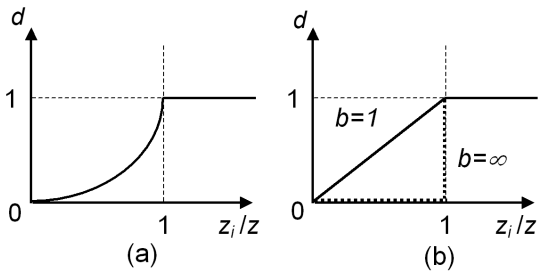

Fig. 3. Resolution discount function.

C. Comparison with similarity metrics.

In pattern recognition and computational geometry standard similarity metrics are Symmetric Difference (SD) and Intersection Over Union (IOU) [3], [2], [28]. For a requested viewing zone r_i and a candidate frame c, the SD metric is

$$SD = \frac{Area(r_i \cup c) - Area(r_i \cap c)}{Area(r_i \cup c)}.$$

The intersection-over-union metric is

$$IOU = \frac{Area(r_i \cap c)}{Area(r_i \cup c)} = 1 - SD.$$

Compared with IOU, our Coverage-Resolution Ratio (CRR) metric has similar properties:

- IOU and CRR attain their minimum value of 0 if and only if $c \cap r_i = \emptyset$,
- both attain their maximum value if and only if $c = r_i$,
- both are proportional to the area of $c \cap r_i$, and
- both depend—albeit differently—on the sizes of c and r_i.

The differences between CRR and SD are that

- the SD metrics is not piecewise linear in x or y,
- it is hard to extend SD to arbitrarily-shaped requested viewing zones because SD will become non-normalized for such cases,
- the SD metric only capture geometric similarity and do not take into account the resolution difference.

IV. Algorithms

In [26], we defined the notion of "virtual corners", which are intersections between extended edges of two requests. We have proved that an one of the corners of an optimal frame must coincide with one of virtual corner. Although [26] only addresses problems with fixed resolution, this result is also true when z is continuous. This virtual corner optimality condition can reduce the 3D optimization problem to $O(n^2)$ 1D optimization problems with respect to variable z. We then show that each 1D optimization problem can be dissected into $O(n)$ piecewise polynomial functions, each of which can be solved in $O(n)$. Using incremental computation and a diagonal sweep, we show how to improve the running time to $O(n^3)$.

A. Basic Virtual Corner Algorithm (BVC)

For n requested viewing zones, there are $O(n^2)$ virtual corners. The virtual corner optimality condition allows us to find

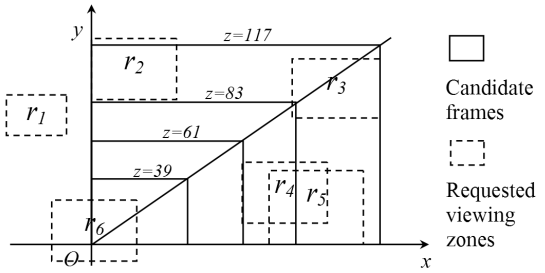

Fig. 4. *An example of the 1D optimization problem with respect to z. In this example, we assume $l(z) = 4z$, $w(z) = 3z$, $b = 1$, and $u_i = 1$ for $i = 1, ..., n$.*

the optimal frame by checking the candidate frames that have one of their corners overlapped with one of the virtual corners. This means that we can reduce the original 3D optimization problem in Equation (2) to $O(n^2)$ 1D optimization problems. Define $p_i(z) = Area(r_i \cap c)$, $a_i = Area(r_i) = w_i l_i$, the 1D optimization problem is to find,

$$\max_z s(z) = \sum_{i=1}^{n} u_i (p_i(z)/a_i) \min((z_i/z)^b, 1) \qquad (3)$$

subject to the constraint that a corner of the candidate frame $c = [x, y, z]$ coincides with a virtual corner.

To study the 1D maximization problem in Equation (3), consider a virtual corner. For simplicity, we assume that the virtual corner is at the origin. Moreover, we assume that the virtual corner coincides with the lower left corner of the candidate frame. (The virtual corner in Figure 4 is the intersection of the extensions of the left edge of r_2 and the bottom edge of r_5.) Placements in which one of the other three corners of the candidate frame coincides with the virtual corner are handled in a similar fashion. We may be able to eliminate some of the placements beforehand, but it reduces the computation by only a constant factor. Now, we gradually increase z and observe the value of $s(z)$: Figure 5 shows the function for the example in Figure 4.

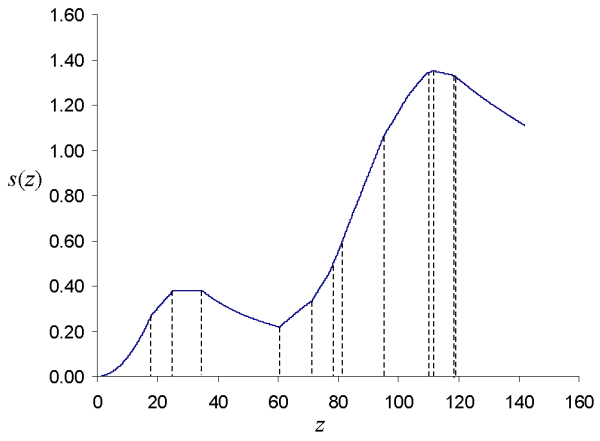

Fig. 5. *Reward function for the example in figure 4 as a function of image resolution .*

a. Critical z Values and Intersection Topologies. The function $s(z)$ is a piecewise smooth function (see Figure 5), so derivative-based approaches cannot be used directly. We refer to a maximal z-interval on which $s(z)$ is smooth as a segment. We consider four questions that form the basis for our algorithms.

1) Can we give a geometric characterization of the endpoints of the segments?
2) How many segments are there?
3) What is the closed-form description of $s(z)$ within a single segment, and how complex is the computation of the maximum of $s(z)$ on that segment?
4) How different are the closed-form descriptions of $s(z)$ on two adjacent segments?

The first three questions lead to an $O(n^4)$ algorithm; the fourth question results in an improvement to $O(n^3 \log n)$.

We start with question 1).

Definition 1: A critical z value is the z value such that $s(z)$ changes its closed-form representation.

Let $Z_c(x_v, y_v)$ be the set of critical z values for virtual corner (x_v, y_v). From Equation (4), we see that the non-smoothness comes from the non-smoothness of either $\min((z_i/z)^b, 1)$ or $p_i(z)$. The critical z values that come from the former type form a subset $Z'_c(x_v, y_v)$, those of the latter type a subset $Z''_c(x_v, y_v)$. The former type is easy to deal with because it occurs at $z = z_i$, $i = 1, ..., n$. Therefore, $Z'_c(x_v, y_v) = \{z_i|\text{i=1,...,n}\}$, so $|Z'_c(x_v, y_v)| = n$. Note that $Z'_c(x_v, y_v)$ is the same for all virtual corners (x_v, y_v), so $Z'_c(x_v, y_v) = Z'_c$.

Obtaining $Z''_c(x_v, y_v)$ is less straightforward. Depending upon the intersection topology, the intersection area $p_i(z)$ of a rectangle r_i with an expanding candidate frame c is one of the following 4 types: it is of type 0 if $p_i(z)$ equals zero, of type 1 if $p_i(z)$ equals a positive constant q_{i0}, of type 2 if $p_i(z)$ is described by a first-degree polynomial $q_{i1}z + q_{i0}$, and of type 3 if $p_i(z)$ is described by a second-degree polynomial $q_{i2}z^2 + q_{i1}z + q_{i0}$, where q_{i0}, q_{i1}, and q_{i2} are coefficients. We are interested in how the type changes as z gradually increases from 0^+ to $+\infty$.

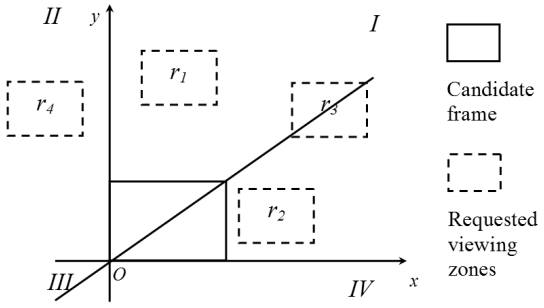

Fig. 6. *Examples for "fundamental rectangles". In this figure, r_1 and r_2 are type (a) rectangles, r_3 is a type (b) rectangle, and r_4 is a type (o) rectangle.*

To further simplify this problem, we consider "fundamental rectangles" from three classes.

- Class (o): A rectangle that does not intersect Quadrant I,
- Class (a): A rectangle that is fully contained in Quadrant I and does not intersect the extended diagonal of the candidate frame.
- Class (b): A rectangle that is fully contained in the Quadrant I and that has a diagonal that overlaps the

extended diagonal of the candidate frame.

Figure 6 gives examples for these three classes of fundamental rectangles.

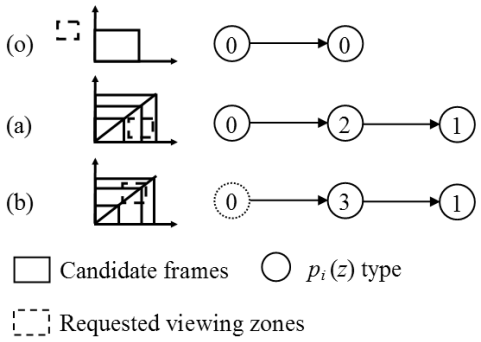

Fig. 7. *Change of $p_i(z)$ for the three classes of requested viewing zones when z gradually increases from 0^+ to $+\infty$.*

As shown in Figure 7, as z increases,

- the $p_i(z)$ for a class (o) rectangle always remains type 0,
- the $p_i(z)$ for class (a) rectangle starts from type 0, changes to type 2 when its intersection with the expanding candidate frame begins, then changes to type 1 when it becomes fully contained.
- the $p_i(z)$ for a class (b) rectangle can start either from type 3 or type 0 depending on whether the bottom left corner of the rectangle coincides with the origin or not. It also changes to type 1 once it becomes fully contained.

The transitions correspond to critical z values.

We can ignore class (o) fundamental rectangles because they do not contribute to our objective function. A requested viewing zone that is a fundamental rectangle from class (a) or (b) generates at most two critical z values. Many of the requested viewing zones though will not be fundamental rectangles. We resolve this by decomposing those requests.

b. Requested viewing zone decomposition. A requested viewing zone that is not a fundamental rectangle intersects at least one of following: the positive x-axis, the positive y-axis, and the extended diagonal of the expanding candidate frame. We treat the different intersection patterns and show that in each case the requested viewing zone can be decomposed into at most four fundamental rectangles (see also Figure 8).

- If the requested viewing zone intersects only the diagonal, then it can be decomposed into two class (a) rectangles and one class (b) rectangle.
- If the requested viewing zone intersects only one positive coordinate axis, then it can be decomposed into a class (a) rectangle and a class (o) rectangle.
- If the requested viewing zone intersects the diagonal and exactly one positive coordinate axis, then it can be decomposed into two class (a) rectangles, one class (b) rectangle, and one class (o) rectangle.
- If the requested viewing zone intersects the diagonal and both positive coordinate axes, then it can be decomposed into one class (a) rectangle, one class (b) rectangle, and two class (o) rectangles.

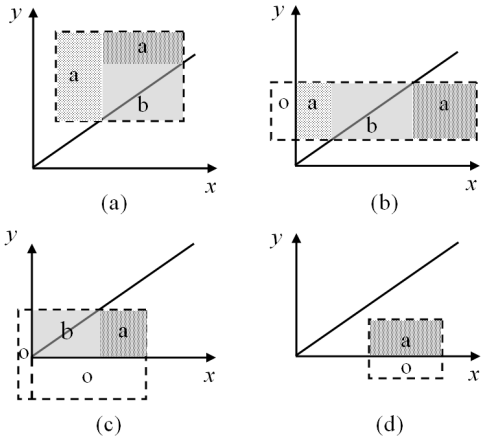

Fig. 8. *Examples of four requested viewing zone decomposition cases.*

As we can see from figure 8, a decomposed requested viewing zone can yield at most three fundamental rectangles that are either class (a) or class (b). Every fundamental rectangle inherits the z_i value of the original request.

In summary, we claim that the n requested viewing zones can be classified and/or decomposed into $O(n)$ fundamental rectangles that are either class (a) or class (b). Since each rectangle in class (a) or (b) generates (at most) two critical z values, we find that $|Z_c''(x_v, y_v)| = O(n)$. Combining this with the bound on the size of $Z_c'(x_v, y_v)$ yields that $|Z_c(x_v, y_v)| = O(n)$. Since the critical z values partition the z axis into $O(n)$ segments, on each of which $s(z)$ is a smooth function, the following lemma is true.

Lemma 1: For each virtual corner, the z-axis can be partitioned into $O(n)$ segments, on which $s(z)$ is smooth.

Lemma 1 answers our question 2) from the previous section.

c. Optimization Problem on a Segment. With the knowledge of question 1) and 2), we are ready to attack question 3): derive a closed-form representation of $s(z)$ on a segment and solve the constrained optimization problem. We have the following lemma. (The order of the resulting polynomial depends on the resolution discount factor b),

Lemma 2: For each segment, $s(z)$ is a polynomial function with 6 coefficients g_0, g_1, g_2, g_3, g_4, and g_5,

$$s(z) = g_0 z^{-b} + g_1 z^{-b+1} + g_2 z^{-b+2} + g_3 + g_4 z + g_5 z^2. \quad (4)$$

Proof: For a virtual corner (x_v, y_v), let us assume the segment is defined by $[z', z'')$, where $z', z'' \in Z_c(x_v, y_v)$ are two adjacent critical z values. The n requested viewing zones have been classified and decomposed into $k = O(n)$ class (a) or (b) rectangles. We denote those rectangles as $\tilde{r}_i$, $i = 1, ..., k$. Let us define set $S' = \{i | z_i \leq z'\}$ and set $S'' = \{i | z_i \geq z''\}$. From the definition of critical z value, we know that $z_i \notin (z', z'')$ for $i = 1, ...n$ so that $S' \cup S'' = \{1, ..., k\}$ and $S' \cap S'' = \emptyset$. Therefore, Equation (3) becomes,

$$s(z) = \sum_{i \in S''} u_i p_i(z)/a_i + \sum_{i \in S'} u_i (p_i(z)/a_i)(z_i/z)^b \quad (5)$$

We also define S_j be the set of rectangles with type j intersection areas when $z \in [z', z'')$, for $j = 1, 2, 3$ respectively.

Recall that $a_i = w_i l_i$ is a constant; we have

$$\begin{aligned}\sum_{i\in S''} u_i p_i(z)/a_i &= \sum_{i\in S''\cap S_1} u_i q_{i0}/a_i \\ &+ \sum_{i\in S''\cap S_2} u_i(q_{i1}z + q_{i0})/a_i \\ &+ \sum_{i\in S''\cap S_3} u_i(q_{i2}z^2 + q_{i1}z + q_{i0})/a_i\end{aligned}$$

We can perform a similar transform for the second term of Equation (5)

$$\begin{aligned}&\sum_{i\in S'}(u_i p_i(z)/a_i)(z_i/z)^b \\ &= z^{-b}\sum_{i\in S'\cap S_1} u_i z_i^b q_{i0}/a_i \\ &+ z^{-b}\sum_{i\in S'\cap S_2} u_i z_i^b(q_{i1}z + q_{i0})/a_i \\ &+ z^{-b}\sum_{i\in S'\cap S_3} u_i z_i^b(q_{i2}z^2 + q_{i1}z + q_{i0})/a_i.\end{aligned}$$

Combining them, we get Equation (4). ■

The proof of Lemma 2 shows that Equation (3) can be converted into Equation (4) in $O(n)$ time. The maximum of Equation (4) can be found in constant time. Combining Lemma 1 and Lemma 2 yields the Basic Virtual Corner Algorithm.

Basic Virtual Corner (BVC) Algorithm

For each virtual corner (x_v, y_v)	$O(n^2)$
Compute members of $Z_c(x_v, y_v)$	$O(n)$
For each segment	$O(n)$
Compute polynomial coefficients	$O(n)$
Find maximum for the polynomial	$O(1)$
End For	
End For	
Report the maximum $s(c)$ and the corresponding c^*.	

Theorem 1: The Basic Virtual Corner algorithm (BVC) solves the problem in $O(n^4)$ time.

1) Virtual Corner with Incremental Computing (VC-IC): The inner loop in the BVC algorithm takes $O(n^2)$, which is the product of two factors: $O(n)$ segments and $O(n)$ time to compute polynomial coefficients. One observation is that we do not need to re-compute the coefficients entirely if we solve the $O(n)$ sub-problems in an ordered manner. Comparing the polynomial coefficients of two adjacent segments, we find that the difference is caused by the critical z that separates the two segments. The critical z value belongs to some rectangle. Therefore, we only need to do a coefficient update on one polynomial to get another one. This update only takes constant time. To exploit this coherence we must sort the elements of $Z_c(x_v, y_v)$ in the inner loop to be able to consider the segments in order; this takes $O(n \log n)$ time. We replace the inner loop in BVC by the following subroutine.

Virtual Corner with Incremental Computing (VC-IC)

Sort members of $Z_c(x_v, y_v)$	$O(n \log n)$
Compute first polynomial coefficients	$O(n)$
For each subsequent segment	$O(n)$
Update polynomial coefficients	$O(1)$
Find maximum for the polynomial	$O(1)$
End For	

The VC-IC algorithm improves the running time:

Theorem 2: The Virtual Corner with Incremental Computing (VC-IC) algorithm solves the problem in $O(n^3 \log n)$.

2) Virtual Corner with Incremental Computing and Diagonal Sweeping (VC-IC-DS): In the outer loop of the VC-IC algorithm, sorting of $Z_c(x_v, y_v)$ for each virtual corner is the dominating factor. The question is: is it necessary to sort critical z values repeatedly for each virtual corner? Recall $Z_c(x_v, y_v)$ is the union of a set Z'_c and a set $Z''_c(x_v, y_v)$.

Each critical z value in $Z''_c(x_v, y_v)$ uniquely defines the position of the upper right corner of the candidate frame on its extended diagonal, which is called critical point in the figure 9(a). Each critical point corresponds to the point that the candidate frame start intersecting some requested viewing zone or the point that the intersection between the candidate frame and some requested viewing zone ends. This gives a geometric interpretation for those critical z values. Figure 9(a) shows a case with two requested viewing zones and five critical z values.

Let $Z''_e(x_v, y_v)$ be the set of the corresponding z values of the intersections between the extended diagonal and the extended edges, which is illustrated in Figure 9(b). $Z_e{}''(x_v, y_v)$ also depends on virtual corner (x_v, y_v). As shown in Figure 9(a) and Figure 9(b),

$$Z_c{}''(x_v, y_v) \subseteq Z_e{}''(x_v, y_v).$$

If we have a sorted sequence $Z_e{}''(x_v, y_v)$, we can get a sorted sequence $Z_c{}''(x_v, y_v)$ by checking whether a point in $Z_e{}''(x_v, y_v)$ belongs to $Z_c{}''(x_v, y_v)$. This takes $O(n)$ time because there are $O(n)$ points in $Z_e{}''(x_v, y_v)$.

Figure 9(c) illustrates a nice property of the sorted sequence of points in $Z_e{}''(x_v, y_v)$. In the figure, we have an ordered sequence of intersected points at the extended diagonal that starts from the origin O. we number the point closest to the origin as point 1 and the second closest as point 2. As we gradually move the extended diagonal downward and observe what happens to the sorted sequence, we find that the order of the sorted sequence does not change until the diagonal line hits an intersection between two extended edges, which is a virtual corner by definition. Let us define this virtual corner be the adjacent virtual corner to the virtual corner at the origin. Point 1 and point 2 switch their order at the adjacent virtual corner (i.e. the gray rectangle in the figure 9(c)). This phenomenon shows that if we have a sorted sequence of the intersection points at a virtual corner, we can get the sorted sequence at an "adjacent virtual corner" in constant time.

This result can reduce the sorting cost from $O(n \log n)$ to $O(n)$ if we handle the virtual corners in a diagonal order: imagine there is a sweep line that has same slope as the extended diagonal and an intercept at $+\infty$, we decrease the

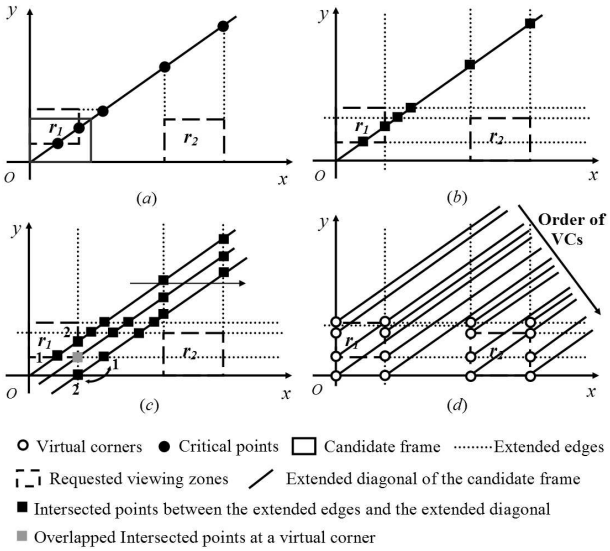

Fig. 9. *(a) $Z_c''(x_v, y_v)$ for a two requested viewing zone case, (b) $Z_e''(x_v, y_v)$ are set of intersection points between the extended diagonal of the candidate frame and the extended edges, (c) The two intersection points switch order only at a virtual corner formulated by the intersection of the two extended edges that generate the two intersection points, and (d) Sorting virtual corners in this order can reduce the sorting cost in the algorithm.*

intercept and stop at each virtual corner. As shown in figure 9(d), we solve the sub problem for the virtual corner when the sweeping line stops. This yields the following VC-IC-DS algorithm.

VC-IC with Diagonal Sweeping (VC-IC-DS) Algorithm

Sort Z_c'	$O(n \log n)$
Sort virtual corners in sweeping order	$O(n^2 \log n)$
Sort $Z_e''(x_v, y_v)$ for the first virtual corner	$O(n \log n)$
For each virtual corner (x_v, y_v)	$O(n^2)$
Update ordered set $Z_e''(x_v, y_v)$	$O(1)$
Get members of $Z_c''(x_v, y_v)$	$O(n)$
Merge Z_c' and $Z_c''(x_v, y_v)$	$O(n)$
Run the sub routine in section IV-A.1.	$O(n)$
End For	
Report the maximum $s(c)$ and the corresponding c^*.	

Theorem 3: The Virtual Corner with Incremental Computing and Diagonal Sweeping (VC-IC-DS) approach solves the problem in $O(n^3)$ time.

V. Results

We have implemented the algorithms using Microsoft Visual C++ on a PC laptop with 1.6Ghz Pentium-M and 512MB RAM. Figure 10 illustrates an sample output with 14 requested frames.

Random inputs are used to test speed of algorithms. The random inputs are generated in two steps. First, we generate four random points, which are uniformly distributed in R_a. The four points represent locations of interests, which are referred as seeds. For each seed, we use a random number to generate a radius of interest. Then we generate requested viewing zones. To generate a requested viewing zone, we need six random numbers. One of them is used to determine which

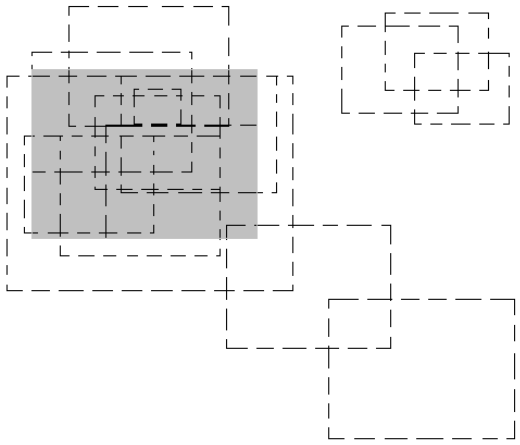

Fig. 10. *An example of computed optimal frame. (shown in grey). We set $b = 1$ and $u_i = 1$ for all requests and use VC-IC-DS Algorithm.*

seed the request will be associated with. Two of them will be used to generate the location of the center point of the request, which is located within the corresponding radius of the associated seed. The remaining three random numbers are used to generate width, length, and resolution for the request.

Figure 11 illustrates the speed difference between BVC, VC-IC, and VC-IC-DS algorithms. Each data point in Figure 11 is an average of 5 trials with different random inputs, where the same random inputs are used to test all three algorithms. The timing results are consistent with the theoretical analysis.

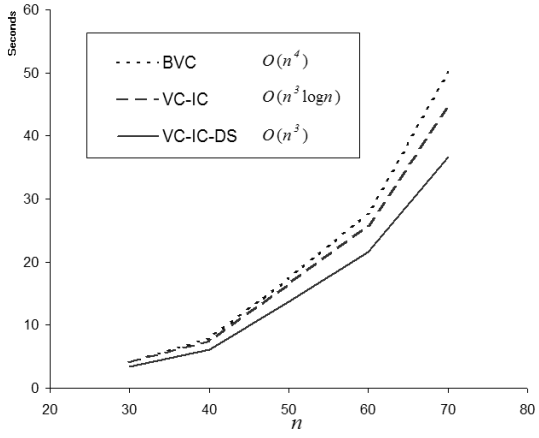

Fig. 11. *Computation speed comparison between three algorithms.*

VI. Conclusions and Future Work

This paper introduces the Satellite Frame Selection problem: automatically finding the optimal satellite frame for a group of competing request regions to maximize reward. Each requested viewing zone is an iso-oriented rectangle with a pair of edges parallel to satellite orbit. The problem is to find a satellite frame that maximizes the total reward. We define a new metric for reward and provide a series of algorithms for solving the nonlinear optimization problem.

In future work, we will consider versions of the problem where the satellite has a third axis to permit yaw motion. In this case the optimal frame is not necessarily aligned with the requested viewing zones. We are also interested in more general cases where the requested viewing zones are non-rectangulary, for example convex or concave polygons. We will also consider extensions to cases where the solution includes more than one frame: allowing p sequential views produces a path planning problem, and allowing p different cameras produces a variant of the p-center "facility location" problem.

ACKNOWLEDGMENTS

We thank A. Levandowski for bringing the Near Real Time Satellite Imaging industry to our attention. Our thanks to M. Overmars, V. Koltun, S. Har-Peled, A. Lim, S. Rao, D. Zhang, D. Halperin, J. Luntz, P. Wright, R. Bajcsy, D. Plautz, C. Cox, D. Kimber, Q. Liu, J. Foote, L. Wilcox, and Y. Rui for insightful discussions and feedback. Thanks also to W. Zheng, K. "Gopal" Gopalakrishnan, R. Alterovitz, I. Y. Song, L. Xiao, J. Voung, J. Wang and for their contributions to the Alpha Lab at UC Berkeley.

REFERENCES

[1] P. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry, volume 23 of Contemporary Mathematics*, pages 1–56, Providence, RI, 1999. American Mathematical Society Press.

[2] A. Z. Broder, M. Charikar, A.M. Frieze, and M.Mitzenmacher. Min-wise independent permutations. *Special Issue for STOC, Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[3] A.Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of SEQUENCES 1997, Positano, Italy, IEEE Comput. Soc.*, pages 21–29, March 1998.

[4] D. J. Cannon. *Point-And-Direct Telerobotics: Object Level Strategic Supervisory Control in Unstructured Interactive Human-Machine System Environments*. PhD thesis, Stanford Mechanical Engineering, June 1992.

[5] N. Chong, T. Kotoku, K. Ohba, K. Komoriya, N. Matsuhira, and K. Tanie. Remote coordinated controls in multiple telerobot cooperation. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3138–3343, April 2000.

[6] G. Dial and J. Grodecki. Applications of IKONOS imagery. In *ASPRS 2003 Annual Conference Proceedings, May 2003, Anchorage, Alaska, USA*, 2003.

[7] M. Lemaitre E. Bensana and G. Verfaillie. Benchmark problems : Earth observation satellite management. *CONSTRAINTS: An International Journal*, (3):293–299, 1999.

[8] D. Eppstein. Fast construciton of planar two-centers. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 131–138, January 1997.

[9] V. Gabrel. Improved linear programming bounds via column generation for daily scheduling of earth observation satellite. Technical report, LIPN, University 13 Paris Nord, January 1999.

[10] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry: Theory & Applications*, 24(3):197–224, April 2003.

[11] K. Goldberg and B. Chen. Collaborative control of robot motion: Robustness to error. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 655–660, October 2001.

[12] K. Goldberg, D. Song, and A. Levandowski. Collaborative teleoperation using networked spatial dynamic voting. *The Proceedings of The IEEE*, 91(3):430–439, March 2003.

[13] R. Grossi and G. F. Italiano. Efficient cross-trees for external memory. In James Abello and Jeffrey Scott Vitter, editors, *External Memory Algorithms and Visualization*, pages 87–106. American Mathematical Society Press, Providence, RI, 1999.

[14] R. Grossi and G. F. Italiano. Revised version of "efficient cross-trees for external memory". Technical Report TR-00-16, Oct. 2000.

[15] N.G. Hall and M. J. Magazine. Maximizing the value of a space mission. *European Journal of Operation Research*, (78):224–241, 1994.

[16] D. Halperin, M. Sharir, and K. Goldberg. The 2-center problem with obstacles. *Journal of Algorithms*, 32:109–134, January 2002.

[17] Sariel Har-Peled, Vladlen Koltun, Dezhen Song, and Ken Goldberg. Efficient algorithms for shared camera control. In *19th ACM Symposium on Computational Geometry, San Diego, CA*, June 2003.

[18] S.A. Harrison and M.E. Price. Task scheduling for satellite based imagery. In *The Eighteenth Workshop of the UK Planning and Scheduling, Special Interest Group, University of Salford, UK*, pages 64–78, 1999.

[19] D. Kimber, Q. Liu, J. Foote, and L. Wilcox. Capturing and presenting shared multi-resolution video. In *SPIE ITCOM 2002. Proceeding of SPIE, Boston*, volume 4862, pages 261–271, Jul. 2002.

[20] M. Lemaitre, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6:367–381, July 2002.

[21] Q. Liu, D. Kimber, L. Wilcox, M. Cooper, J. Foote, and J. Boreczky. Managing a camera system to serve different video requests. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME), Lausanne, Switzerland*, volume 2, pages 13–16, Aug. 2002.

[22] M. McDonald, D. Small, C. Graves, and D. Cannon. Virtual collaborative control to improve intelligent robotic system efficiency and quality. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 418–424, April 1997.

[23] N. Megiddo and K.J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13:182–196, February 1984.

[24] D. Song and K. Goldberg. Sharecam part I: Interface, system architecture, and implementation of a collaboratively controlled robotic webcam. In *IEEE/RSJ International Conference on Intelligent Robots (IROS)*, Nov. 2003.

[25] D. Song, A. Pashkevich, and K. Goldberg. Sharecam part II: Approximate and distributed algorithms for a collaboratively controlled robotic webcam. In *IEEE/RSJ International Conference on Intelligent Robots (IROS)*, Nov. 2003.

[26] D. Song, A. F. van der Stappen, and K. Goldberg. Exact and distributed algorithms for collaborative camera control. In *The Workshop on Algorithmic Foundations of Robotics*, Dec. 2002.

[27] M. Vasquez and J.-K. Hao. A logic-constraint knapsack formulation of a tabu algorithm for the daily photograph scheduling of an earth observation satellite. *J. Comput. Optim. Appl.*, 20(2):137–157, Appl. 2001.

[28] R. C. Veltkamp and M. Hagedoorn. Shape similarity measures, properties, and constructions. In *Advances in Visual Information Systems, 4th International Conference, VISUAL 2000, Lyon, France, November 2-4, 2000, Proceedings VISUAL*, volume 1929, pages 467–476. Springer, 2000.

[29] www.eomonline.com/Common/industrynews/industrynews32702.html.

[30] D. Zhang, V. J. Tsotras, and D. Gunopulos. Efficient aggregation over objects with extent. In *Proc. of 21th ACM International SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Madison, Wisconsin, USA*, pages 121–132, June 2002.