Exact Algorithms for Single Frame Selection on Multiaxis Satellites

Dezhen Song, *Member, IEEE*, A. Frank van der Stappen, *Member, IEEE*, and Ken Goldberg, *Fellow, IEEE*

***Abstract*—New multi-axis satellites allow camera imaging parameters to be set during each time slot based on competing demand for images, specified as rectangular requested viewing zones over the camera's reachable field of view. The single frame selection (SFS) problem is to find the camera frame parameters that maximize reward during each time window. We formalize the SFS problem based on a new reward metric that takes into account area coverage and image resolution. For a set of n client requests and a satellite with m discrete resolution levels, we give an algorithm that solves the SFS problem in time $O(n^2m)$. For satellites with continuously variable resolution ($m = \infty$), we give an algorithm that runs in time $O(n^3)$. We have implemented all algorithms and verify performance using random inputs.**

***Note to Practitioners*—This paper is motivated by recent innovations in earth imaging by commercial satellites. In contrast to previous methods that required waits of up to 21 days for desired earth-satellite alignment, new satellites have onboard pan-tilt-zoom cameras that can be remotely directed to provide near real-time response to requests for images of specific areas on the earth's surface. We consider the problem of resolving competing requests for images: Given client demand as a set of rectangles on the earth surface, compute camera settings that optimize the tradeoff between pan, tilt, and zoom parameters to maximize camera revenue during each time slot. We define a new quality metric and algorithms for solving the problem for the cases of discrete and continuous zoom values. These results are a step toward multiple frame selection which will be addressed in future research. The metric and algorithms presented in this paper may also be applied to collaborative teleoperation of ground-based robot cameras for inspection and videoconferencing and for scheduling astronomic telescopes.**

***Index Terms*—Camera, ground imaging, multi-axis, satellite, teleoperation, telerobotics.**

I. Introduction

A TRADITIONAL imaging satellite, such as Spot 5 [2], can only passively scan a narrow land belt that is directly underneath its orbit like a one-dimensional (1-D) scanner. The time between consecutive visits over the same area of the earth varies from 16 to 44 days depending on the orbit parameters. This intervisit delay makes it difficult to respond to time critical tasks. However, a new generation of commercial satellites can perform pitch and roll operations, allowing their imaging sensors to access larger regions of the earth [14]. Such satellites, known as agile, or multi-axis satellites, selectively view regions of interest and increase response time. For example, the French PLEIADES satellite can visit the same area of the earth twice per day to produce near-real-time (NRT) images, suitable for time-critical applications, such as urban disaster response and search and rescue. The exact algorithm presented here is particularly useful for imaging specific zones such as buildings and streets.

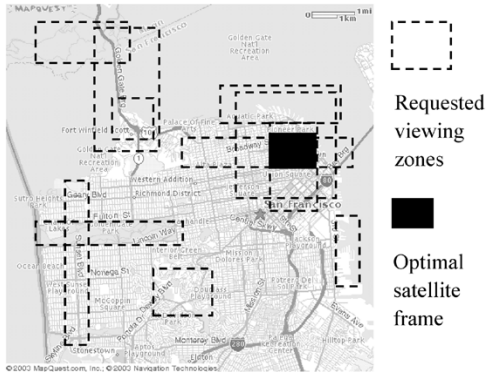


Fig. 1. SFS problem: Each time slot defines the camera's reachable field of view where client requests for images are shown with dashed rectangles. Given a set of requests, the problem is to compute a camera frame that optimizes reward. The solution in this case is illustrated with a solid rectangle.

In addition, advances in camera and imaging technology allow sensors to change imaging resolution. Whereas traditional satellites are similar to fixed 1-D scanners, new satellites are similar to robotic pan-tilt-zoom cameras. In this paper, we use the term "camera" to refer to onboard imaging sensors and use the terms "pan, tilt, and zoom" to describe satellite axis settings.

Multiaxis variable resolution satellites present new challenges for automation. In this paper, we study the frame selection problem for a single time slot, where the camera's field of view is restricted to a rectangular zone on the Earth's surface. During each time slot, a number of client requests for images are pending: Only one image can be captured. The commercial price charged for images depends on image resolution and coverage. We consider the problem of computing pan, tilt, and zoom parameters that will maximize reward.

The single frame selection (SFS) problem is illustrated in Fig. 1. Input is a set of n iso-oriented-requested rectangular regions. The camera image frame is a rectangle with a fixed aspect ratio. We propose a reward metric based on how closely a requested viewing zone compares with a candidate satellite

Manuscript received February 7, 2004; revised November 22, 2004. This paper was recommended for publication by Associate Editor S. Akella and Editor N. Viswanadham upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant IIS-0113147, in part by the Intel Corporation, in part by the University of California (UC) Berkeley's Center for Information Technology Research in the Interest of Society (CITRIS), and in part by TAMU Startup funds.

D. Song is with the Computer Science Department, Texas A&M University, College Station, TX 77843 USA (e-mail: dzsong@cs.tamu.edu).

A. F. van der Stappen is with the Department of Information and Computing Sciences, Utrecht University, Utrecht 3508 TB, The Netherlands (e-mail: frankst@cs.uu.nl).

K. Goldberg is with the IEOR and EECS Departments, University of California, Berkeley, CA 94720 USA (e-mail: goldberg@ieor.berkeley.edu).

Digital Object Identifier 10.1109/TASE.2005.860617

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. SFS is a nonlinear optimization problem; we exploit the structure of the problem to develop polynomial time algorithms that compute the exact solution for cameras with discrete and continuous resolution levels.

II. Related Work

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

In the satellite space mission problem [15], the goal is to optimally schedule a set of jobs on a satellite. Each candidate job has fixed duration, an available time window, and weight. The goal is to select a feasible sequence of jobs maximizing the sum of weights. This combinatorial optimization problem is known to be NP-hard. Recent research [2], [8], [18], [33] on the satellite space mission problem and its variations focuses on developing exact and approximate methods using numerical methods, such as column generation, Tabu search, and genetic algorithms (GAs).

Lemaitre *et al.* [21] studied 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 [9]. 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 an optimal zoom/resolution setting.

The structure of the multiple frame selection problem is related to the planar p-center problem, which Megiddo and Supowit [25] 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 [7] found an algorithm for 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 [35]. 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 [12], [13] 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, Liu, Kimber, Foote, and Liao combine a fixed panoramic camera with a robotic pan-tilt-zoom camera for collaborative videoconferencing [23]. They [22] address a frame selection problem by partitioning the solution space into small nonoverlapping regions. They estimate the probability that each small region will be viewed based on the frequency that this region intersects with user requests. Based on the probability distribution, they choose the optimum frame by minimizing discrepancy in probability-based estimation. Their approach is approximative and relies on how the nonoverlapping small regions are defined. This approach is promising but may be difficult to compute efficiently. One advantage of their approach is a Bayesian model for estimating user zone requests based on past data when explicit requests are unavailable. This is an interesting and important variant on the problem we address in our paper, where regions of interest are specified with exact rectangular frames, which is especially appropriate to imaging of urban areas where specific buildings or blocks are requested.

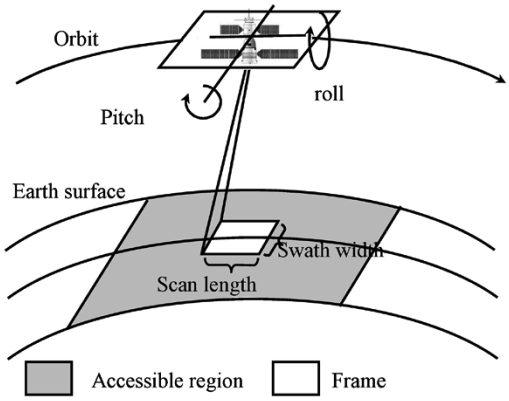


Fig. 2. Multiaxis satellite over an accessible viewing region.

The Alpha 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.* [6], these are multiple operator single robot (MOSR) systems. An Internet-based MOSR system is described by McDonald, Cannon, and colleagues [5], [24]. 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. Recent developments on MOSR systems can be found in [10] and [11].

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 [30], approximation solution with continuous zoom [28], [29], and approximate solution with fixed zoom [17]. This journal paper collects and expands our results on exact solutions [31]. It also introduces a new reward metric and extends the problem formulation to requested viewing rectangles of arbitrary aspect ratio.

III. Problem Definition

In this section, we review satellite terminology and formalize the SFS problem based on the new reward metric.

A. Inputs and Assumptions

1) Multiaxis Satellites: As illustrated in Fig. 2, the camera on a typical satellite orbits the Earth at a speed of more than

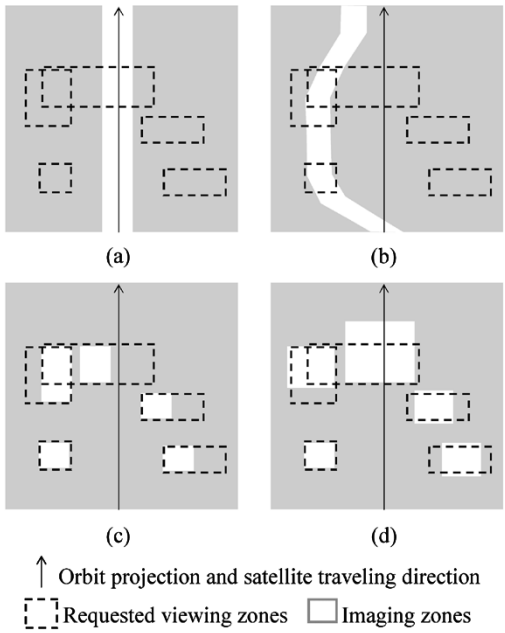


Fig. 3. Reachable field of view for different satellite models. (a) A zero-axis satellite can only scan a narrow land belt underneath its orbit. (b) A single axis satellite with roll motion can scan along a predefined trajectory. (c) A two-axis satellite with pitch and roll capability can view regions of interest. (d) A satellite with pitch, roll, and variable resolution can view regions of interest at different resolutions.

7 kilometers per second. A multi-axis satellite platform usually has a reflection mirror that moves under pitch and roll control, which directs a rectangular region of the earth into the camera's field of view. The roll motion acts like the pan motion of a robotic pan-tilt-zoom camera. The pitch motion acts similar to the tilt motion of the robotic pan-tilt-zoom camera.

The platform and sensor technologies determine the variables in selecting zones for satellite imaging. Fig. 3 illustrates the different capabilities for four different types of satellites, further illustrated in Table I. Our frame selection algorithm applies to the multi-axis satellites in Fig. 3(c) and (d).

2) Definition of a Satellite Frame: Since current satellites cannot perform yaw rotations, we assume the satellite image frame is aligned with its motion direction and consider only iso-oriented satellite image frames. We define a satellite image frame to be a rectangle: $c = [x, y, z, w, l]$ where $[x, y] \in R_a$ specifies the center point of the frame with respect to an accessible region R_a and z specifies the resolution of the frame. The pair $[x, y]$ determines the pitch and roll angles. Setting $z = 10$ m means a pixel in the image is equivalent to area of 10×10 m^2. A bigger z-value means lower image resolution but larger coverage. The attainable resolution set is Z, so $z \in Z$.

3) Imaging Sensor Types: As shown in Fig. 2, the swath width w and the scan length l of the frame determine the size of the frame. They are functions of resolution z for a given time window $T : w = w(z)$ and $l = l(z)$, which depends on the type of imaging sensors.

- *Push broom sensor:* The 1-D camera scans parallel to the satellite's orbit. In this case, swath width $w(z) = \alpha z + \beta$, where α and β are constant factors. For a 1-D camera with fixed swath width, $\alpha = 0$. The scanning length $l(z)$ satisfies $l(z) = \eta z T$, where ηz is the scanning speed, which is also a linear function of z; this means that we can scan faster when the required image resolution is low. If the scanning speed cannot be adjusted, then $l(z) = \eta T$ is a constant for the fixed T.
- *Whisk broom sensor:* The 1-D camera scans orthogonal to orbit. In this case, we can simply switch the $w(z)$ and $l(z)$ functions defined for push broom sensors.
- *Frame camera:* This is a two-dimensional (2-D) nonscanning camera; for example, the TK-350 camera on Russian KOSMOS. Let $\alpha_1, \beta_1, \alpha_2$ and β_2 be constants. In this case, $w(z) = \alpha_1 z$ for variable resolution or $w(z) = \beta_1$ for fixed resolution, which is the same as the $w(z)$ in the push broom sensor. The $l(z) = \alpha_2 z$ or $l(z) = \beta_2$ shares the same format but with different constants.

The definitions of $l(z)$ and $w(z)$ show how we have to balance the area of coverage and resolution because the satellite can only process a limited amount of information in the given fixed time window. We can see that the $w(z) = \alpha_1 z$ and $l(z) = \alpha_2 z$ of the frame camera with variable resolution is a generic formulation that includes all other settings as special cases. If we can solve the problem for the frame camera with variable resolution, other cases can also be solved with appropriate simplifications. In the rest of this paper, we will focus on this generic formulation. The camera frame $c = [x, y, z, w, l]$ is reduced to a triple $c = [x, y, z]$. For example, if a frame has $\alpha_2 = 1333$ and $\alpha_1 = 1000$, then the area of the frame is $1000 \times 1333 \times z^2$.

4) Requested Viewing Zones: For a given time window, we consider n requested viewing zones. The ith 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 the 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 (not necessarily with a 4:3 aspect ratio) have a pair of edges parallel to orbit. Therefore, they are iso-oriented with the satellite frame.

5) Solution Space and Image Distortion: Given a set of n requested viewing zones, the objective is to compute a single frame c^* that yields maximum total reward. The solution space is

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

We consider two cases: 1) Z is a finite discrete set; 2) Z is a continuous set. However, set Z may not be uniform across R_a due to image distortion. As shown in Fig. 2, the distance between a satellite and ground is the shortest when the imaging zone is directly underneath the satellite, known as the nadir, which yields the best resolution. When the satellite is off-nadir, the resolution decreases. For example, for the Quickbird 2 satellite, its nadir resolution is 2.5 m and its 30° off-nadir resolution is 2.9 m [19].

B. Reward Metric

We want to measure how well a candidate satellite frame satisfies a user's request. In this section, we define the "coverage-resolution ratio" (CRR) as follows. For a given candidate

TABLE I
SCHEDULING METHODS FOR DIFFERENT SATELLITES

Type	Roll	Pitch	Resolution	Scheduling and Planning	Representative Satellites
Figure 3(a)	No	No	Fixed	No	SPOT5 [2]
Figure 3(b)	Yes	No	Fixed	Trajectory Planning	RADARSAT-2 [27]
Figure 3(c)	Yes	Yes	Fixed	Frame Selection	AEOS [14], PLEIADES [26], IKONOS [21]
Figure 3(d)	Yes	Yes	Variable	Frame Selection	TerraSAR-X [32]

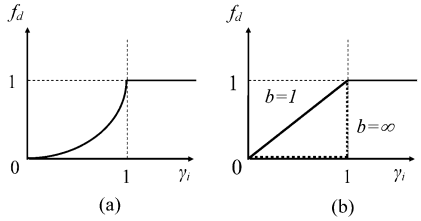


Fig. 4. Resolution discount function. (a) A general f_d function. (b) An f_d function takes the form of (3).

TABLE II
SAMPLE RESOLUTION DISCOUNT FUNCTION f_d FOR DISCRETE z. DATA ARE BASED ON PRICING DATA FROM RADARSAT 1 SATELLITE ON DECEMBER 18, 2003 [20] AND $z_i = 8m$

z	Price per km^2	f_d
8m	\$1.200	1.000
25m	\$0.275	0.229
30m	\$0.133	0.111
50m	\$0.033	0.028
100m	\$0.012	0.010

satellite frame, its reward is related to how much it overlaps with a user request and how good its resolution is. We can think of the former as coverage discount and the latter as resolution discount:

1) Coverage Discount: Recall that r_i is the ith requested viewing zone. The corresponding client has a utility u_i for this region. Define s_i as the reward from the ith request. Let $c = [x, y, z]$ be a candidate camera frame. If 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 coverage ratio

$$s_i = u_i \frac{\text{Area}(r_i \cap c)}{\text{Area}(r_i)}.$$

2) Resolution Discount: Define resolution ratio $\gamma_i = (z_i/z)$. If $\gamma_i < 1$ (the resolution requirement is not satisfied), then the reward should be discounted by a resolution discount factor $f_d(\gamma_i)$. Hence

$$s_i = u_i \frac{\text{Area}(r_i \cap c)}{\text{Area}(r_i)} f_d(\gamma_i). \tag{1}$$

As illustrated in Fig. 4(a), the resolution discount function $f_d(\gamma_i)$ has to satisfy the following conditions:

$$\begin{aligned} &0 \leq f_d(\gamma_i) < 1, \text{ when } \gamma_i < 1 \\ &f_d(\gamma_i) = 1, \text{ when } \gamma_i \geq 1 \\ &f_d(\gamma_i) \leq f_d(\gamma_i') \Leftrightarrow \gamma_i \leq \gamma_i'. \end{aligned} \tag{2}$$

It is an increasing function of γ_i because an image has more value as resolution increases. This parameterized definition of the CRR includes a large family of metrics.

3) Sample CRR Metrics: In Section III-A-V, we state that we solve the SFS problem for two cases: 1) Z is a finite discrete set and 2) Z is a continuous set. For case 1), the resolution function f_d does not need to be a continuous function. It could be a table corresponding to a small set of resolution levels. Table II gives a sample f_d function.

For case 2), we assume f_d is a continuous function. In the rest of the paper, we use the f_d in (3) to walk through the algorithm development. Readers can expand the algorithm to fit different f_d functions as long as f_d is composed of elementary functions

$$f_d(\gamma_i) = \min\{(\gamma_i)^b, 1\}. \tag{3}$$

Now the CRR becomes

$$s_i(c) = u_i \frac{\text{Area}(r_i \cap c)}{\text{Area}(r_i)} \min\left(\left(\frac{z_i}{z}\right)^b, 1\right). \tag{4}$$

The exponential discount factor b determines how resolution affects reward. Fig. 4(b) shows two cases $b = 1$ and $b = \infty$. The case is $b = \infty$ and corresponds to a scenario in which users will not accept any image with a resolution that is lower than requested. We use the case $b = 1$ as a default setting for numerical examples in the rest of the paper.

Given n requests, the total reward is the sum of individual rewards

$$s(c) = \sum_{i=1}^{n} s_i(c). \tag{5}$$

Our objective is to find $c^* = \arg\max_c s(c)$, the frame that maximizes total reward.

C. Properties of the CRR Reward Metric

In this section, we explore the properties of the CRR metric, which can help us to develop algorithms later. The CRR metric s is nonsmooth and piecewise linear in both x and y. For convenience, we use $s(x, y, z)$ instead of $s(c)$ with $c = [x, y, z]$.

1) Separability: According to (1), we know that the choice of $[x, y]$ does not affect value of resolution discount function f_d. This property allows us to reduce the problem from a single high-dimensional problem to multiple 1-D problems.

2) Nonsmoothness: Recall that $\text{Area}(r_i) = w_i l_i$. To study the properties of the reward function, we first treat z as a constant: $p_i(x, y) = \text{Area}(c \cap r_i)$ is a function of (x, y). The objective function defined by (1) becomes a function of the center point of the candidate frame

$$s(x, y) = \sum_{i=1}^{n} \omega_i p_i(x, y) \tag{6}$$

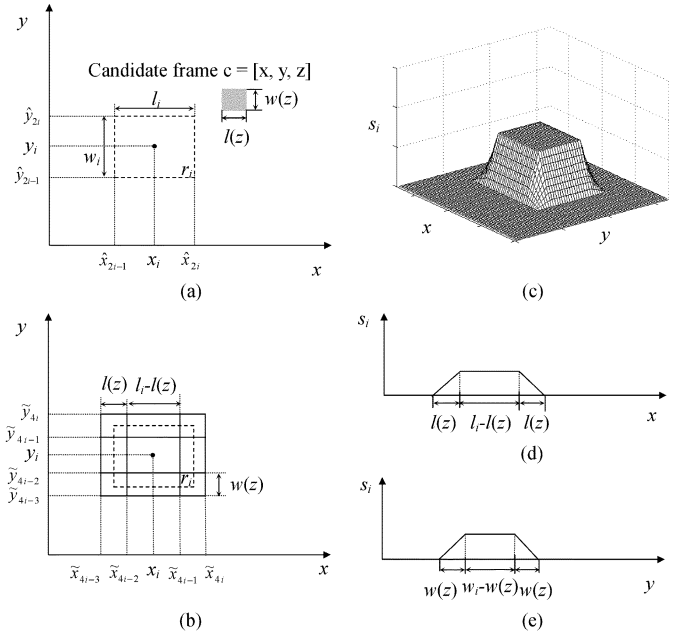


Fig. 5. CRR reward function for a single requested viewing zone. Consider the requested frame centered at x_i, y_i as shown in (a). Consider a small candidate frame (shown in gray). As we move the candidate frame, (c) illustrates the resulting CCR function. As illustrated in (b), (c), (d), and (e), the function is plateau-like with a maximum height of $u_i \mathrm{Area}(c \cap r_i)/\mathrm{Area}(r_i)$. The function consists of five planar and four quadratic surfaces at the corners. (a) The ith requested viewing zone r_i, (b) top view of $s_i(x, y)$, and (c) 3-D shape of $s_i(x, y)$. (d) Front view of $s_i(x, y)$. (e) Side view of $s_i(x, y)$.

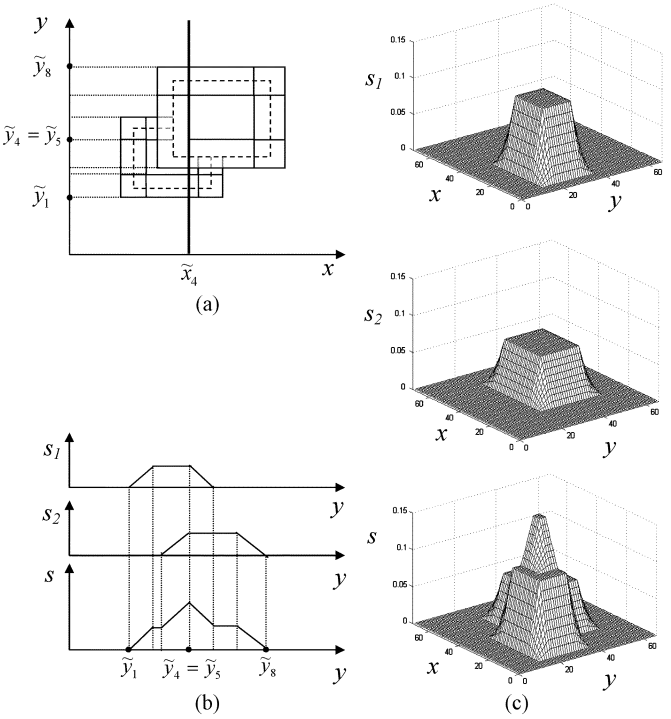


Fig. 6. Combined CCR function for two users. Ordered sets $\{\bar{y}_k\}$ and $\{\bar{x}_k\}, k = 1, \ldots, 8$ correspond to horizontal and vertical edges of plateaus. Note that $\bar{y}_4$ and $\bar{y}_5$ overlap in this case and $s = s_1 + s_2$. (a) Top view of S_1 and S_2. (b) Piecewise linear objective function at $x = \bar{X}_4$ and (c) 3-D view of objective functions.

where

$$\omega_i = \frac{u_i}{w_i l_i} f_d(\gamma_i) \tag{7}$$

is a constant for each user. We know that $p_i(x, y)$ is the area of the intersection of the ith requested viewing zone and the candidate frame (x, y, z). Therefore, the maximum value of $p_i(x, y)$ is $\min(\mathrm{Area}(c), \mathrm{Area}(r_i))$. This property determines that the shape of user i's satisfaction function is plateau-like. Since z is fixed, the objective function $s_i(c)$ becomes a function of the center point of the candidate frame $s_i(x, y)$. Fig. 5 shows the shape of $s_i(x, y)$ given $z < z_i$ and candidate frame c smaller than r_i. Note that s_i is nondifferentiable with respect to x and y so we cannot use derivative-based approaches to find optima.

3) Piecewise Linearity in x and y: Since all requested viewing zones and the candidate frame are iso-oriented rectangles, the shape of any intersection between them is also a rectangle with its edges parallel to either x-axis or y-axis. Thus, the term $p_i(x, y)$ in (6) is either 0 or the area of the rectangle formed by an intersection between r_i and $c = [x, y, z]$. This yields a nice property: the $p_i(x, y)$ is piecewise linear with respect to x if we fix y, and piecewise linear with respect to y if we fix x. Since the total reward function $s(x, y)$ is a linear combination of $p_i(x, y), i = 1, \ldots, n$, it has the same property. Fig. 6 shows an example for a case with two requested viewing zones.

D. Other Reward Metrics

Symmetric difference (SD) and intersection over union (IOU) are standard "similarity metrics" used in pattern recognition as a measure of how similar two shapes are [4], [3], [34], [36]. In our case, for a requested viewing zone r_i and a candidate frame c, the SD metric would be

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

The IOU metric would be

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

Compared with IOU, our 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 $c \cap r_i$;
- both depend—albeit differently—on the sizes of c and r_i.

The key differences between CRR and these metrics are the following:

- SD and IOU metrics are not piecewise linear in x or y;
- SD and IOU metrics do not take resolution into account.

IV. Algorithms

In this section, we start with the definition of two geometric classes of critical points: "base vertices" (BVs) and "plateau vertices." These critical points allow us to specify efficient algorithms for the SFS problem. Section IV-B describes an algorithm when the resolution is restricted to a set of discrete values. Section IV-C describes the extension to the case where resolution varies continuously.

A. Plateau and Base Vertices

1) Plateau Vertices: Plateau vertices depend on a particular candidate frame resolution level. Recall that the objective function for a given resolution z and one requested viewing zone $s_i(x, y)$ is a plateau-like function as shown in Fig. 5(c). The function consists of nine facets: one top plane, four side planes, and four quadratic surfaces at the corners. There are two vertical boundaries and two horizontal boundaries at the bottom (bounding the entire plateau), the same numbers of similar edges at the top (bounding the plateau's flat top), and eight boundaries separating side planes and corner quadratic surfaces [Fig. 7(a)]. The requested viewing zone r_i has four vertical plateau boundaries at $\{\tilde{x}_{4i-3}, \tilde{x}_{4i-2}, \tilde{x}_{4i-1}, \tilde{x}_{4i}\}$ and four horizontal plateau boundaries at $\{\tilde{y}_{4i-3}, \tilde{y}_{4i-2}, \tilde{y}_{4i-1}, \tilde{y}_{4i}\}$. Recall that $r_i = [x_i, y_i, w_i, l_i, z_i, u_i]$, $w(z)$ and $l(z)$ are the width and length of the candidate satellite frame, respectively, and $\hat{x}_{2i-1}, \hat{x}_{2i}, \hat{y}_{2i-1}$, and $\hat{y}_{2i}$ are left, right, bottom, and top extended edge of the requested zone, respectively. As shown in Fig. 5(b):

- the four vertical plateau boundaries satisfy

$$\tilde{x}_{4i-3} = \hat{x}_{2i-1} - .5l(z), \quad \tilde{x}_{4i-2} = \hat{x}_{2i-1} + .5l(z),$$
$$\tilde{x}_{4i-1} = \hat{x}_{2i} - .5l(z), \quad \tilde{x}_{4i} = \hat{x}_{2i} + .5l(z). \tag{8}$$

- four horizontal plateau boundaries satisfy

$$\tilde{y}_{4i-3} = \hat{y}_{2i-1} - .5w(z), \quad \tilde{y}_{4i-2} = \hat{y}_{2i-1} + .5w(z),$$
$$\tilde{y}_{4i-1} = \hat{y}_{2i} - .5w(z), \quad \tilde{y}_{4i} = \hat{y}_{2i} + .5w(z). \tag{9}$$

For a given frame resolution z and n requested viewing zones, there are n plateaus. Computing the plateau boundaries for all requested viewing zones, we obtain a set of vertical boundaries $\tilde{X} = \bigcup_{i=1}^{n}\{\tilde{x}_{4i-3}, \tilde{x}_{4i-2}, \tilde{x}_{4i-1}, \tilde{x}_{4i}\}$, and a set of horizontal boundaries $\tilde{Y} = \bigcup_{i=1}^{n}\{\tilde{y}_{4i-3}, \tilde{y}_{4i-2}, \tilde{y}_{4i-1}, \tilde{y}_{4i}\}$.

We define a plateau vertex as an intersection between any two boundaries, which includes both intersections of facet boundaries induced by a single plateau or by two distinct plateaus. Note that plateau vertices occur at the intersections between the actual boundaries of plateaus rather than between extended boundaries. Since all plateaus are iso-oriented, one of the intersecting boundaries is horizontal and the other is vertical. A plateau vertex can be represented by a three-dimensional (3-D) vector $(\tilde{x}, \tilde{y}, z), \tilde{x} \in \tilde{X}$, and $\tilde{y} \in \tilde{Y}$. For n requested viewing zones and m fixed resolutions, there are $O(mn^2)$ plateau vertices. Fig. 7(a) shows an example of plateau vertices for two requested viewing zones.

2) Base Vertices: We define BVs to be independent of image resolution. As illustrated in Fig. 7(b), a BV occurs at the intersection of the extended edges of requested viewing rectangles.

As shown in Fig. 5(a), a requested zone r_i has:

- two vertical extended edges at

$$\hat{x}_{2i-1} = x_i - .5l_i, \quad \hat{x}_{2i} = x_i + .5l_i \tag{10}$$

- and two horizontal extended edges at

$$\hat{y}_{2i-1} = y_i - .5w_i, \quad \hat{y}_{2i} = y_i + .5w_i. \tag{11}$$

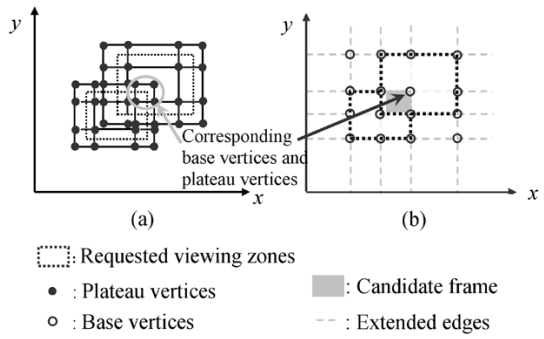


Fig. 7. Illustration of the relationship between plateau vertices and BVs for two requested viewing zones and a candidate frame with a given resolution. A plateau vertex corresponds to a candidate frame with one of its corners coincident with a BV. (a) Plateau vertices. (b) BVs.

So for n requested viewing zones, there are $O(n^2)$ BVs. Each BV is represented by a 2-D vector in the (x, y) plane.

3) Relationship Between Plateau and Base Vertices: Equations (8)–(11) and Fig. 7 show the relationship between plateau vertices and BVs, which is also described by Lemma 1.

Lemma 1 (Plateau/Base Vertex Invariant): Any candidate frame that corresponds to a plateau vertex must have one corner coincident with a BV.

Although we can find a corresponding BV for each plateau vertex, they are not equivalent. There are three reasons.

- The notion of plateau vertex is only applicable to cases where the resolution of the candidate frame is discrete and finite; the notion of BV applies to cases where the resolution is either discrete or continuous.
- Not all BVs have corresponding plateau vertices. For example, the top left BV in Fig. 7(b) does not have a corresponding plateau vertex. In fact, a BV that does not lay on a real (as opposed to extended) edge of a requested viewing zone does not have a corresponding plateau vertex.
- For m discrete resolutions, a BV has $O(m)$ corresponding plateau vertices. A BV has, at most, four corresponding plateau vertices for a fixed resolution. The position of the BV is invariant to resolution by definition.

4) Optimality Conditions: Plateau vertices and BVs can help us to find the optimal solution for the optimization problem defined in (5) for the discrete resolution case and the continuous resolution case, respectively.

Lemma 2 (Base Vertex Optimality Condition): At least one optimal frame has one corner coincident with a BV.

Proof: Let $c^* = [x^*, y^*, z^*]$ be an optimal solution. If we fix $z = z^*$, we get $s(x, y)$ as a summation of plateaus. As discussed earlier, for a fixed z and x, the objective function $s(y)$ is piecewise linear. So the optimum must be at a vertex $y = \tilde{y}$ such that $s(x^*, \tilde{y}, z^*) = s(x^*, y^*, z^*)$. We also know that the line $y = \tilde{y}$ in the (x, y) plane is one of the horizontal facet boundaries of the plateaus. Similarly, we can find another optimal frame $[\tilde{x}, \tilde{y}, z^*]$, where line $x = \tilde{x}$ is one of the vertical facet boundaries of the plateaus. Therefore, the optimal frame $[\tilde{x}, \tilde{y}, z^*]$ is centered at a plateau vertex $(\tilde{x}, \tilde{y})$ for a fixed resolution $z = z^*$. Applying Lemma 1, we know that the optimal frame $[\tilde{x}, \tilde{y}, z^*]$ must have one corner at one of the BVs. ■

Using the BV optimality condition (BVOC), we consider only frames where one corner is coincident with a BV, thereby reducing the dimensionality of the problem. The BVOC is true no matter whether the resolution variable is discrete or continuous. However, it is more convenient to use plateau vertices when the resolution variable is discrete. The BVOC can be transformed to the following plateau vertex optimality condition.

Lemma 3 (Plateau Vertex Optimality Condition): When the resolution variable is discrete, at least one optimal frame has a corner coincident with a plateau vertex.

Proof: From the proof the Lemma 2, we know that we can find an equivalent optimal solution $[\tilde{x}, \tilde{y}, z^*]$ from a given optimal solution $c^* = [x^*, y^*, z^*]$. We also know that $[\tilde{x}, \tilde{y}]$ is the intersection of two facet boundaries. For the discrete resolution case, z^* has to be one of the discrete resolutions in the solution space. Then, the point $[\tilde{x}, \tilde{y}, z^*]$ is one of the plateau vertices. ■

B. Algorithms for Discrete Resolution

For satellites with a discrete set of m resolution levels, we present algorithms to find the optimal image position and resolution parameters.

1) Brute Force Approach: Based on the Lemma 3, we can solve the optimization problem by simply checking all combinations of resolution levels and corresponding plateau vertices. We evaluate the objective function for each of the $O(n^2)$ plateau vertices and repeat this for each of the m resolution levels. It takes $O(n)$ time to evaluate a candidate frame c. Therefore, the brute force algorithm runs in $O(n^3 m)$.

2) Efficient Traversal of Plateau Vertices: For n requested viewing zones, we have $4n$ horizontal plateau facet boundaries $\{\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{4n}\}$ and $4n$ vertical plateau facet boundaries $\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{4n}\}$. The plateau vertex traversal (PVT) algorithm is summarized below. It reduces the computation complexity from $O(n^3 m)$ to $O(n^2 m)$.

Plateau Vertex Traversal (PVT) Algorithm

$s^* = 0$,	$O(1)$
Sort $\{\hat{y}_{2i-1}\}$, $i = 1, ...n$	$O(n \log n)$
Sort $\{\hat{y}_{2i}\}$, $i = 1, ...n$	$O(n \log n)$
For each resolution level z	(m in total)
(i). Compute $\tilde{X}$	$O(n)$
(ii). Get $\{\tilde{y}_{4i-3}\}$, $\{\tilde{y}_{4i-2}\}$ from $\{\hat{y}_{2i-1}\}$	$O(n)$
Get $\{\tilde{y}_{4i-1}\}$, $\{\tilde{y}_{4i}\}$ from $\{\hat{y}_{2i}\}$	$O(n)$
Merge the 4 ordered sequences: $\{\tilde{y}_{4i-3}\}$, $\{\tilde{y}_{4i-2}\}$, $\{\tilde{y}_{4i-1}\}$, and $\{\tilde{y}_{4i}\}$, $i = 1, ..., n$ to get the sorted $\tilde{Y}$.	$O(n)$
(iii). /*Solve 1D problems */	$O(n^2)$
For $x = \tilde{x}_i$, $i = 1, ..., 4n$, $s = \max_y\ s(\tilde{x}_i, y, z)$, if $s > s^*$ then $s^* = s, x^* = \tilde{x}_i, y^* = y, z^* = z$.	
End For	
End For	
Output s^* as optimal objective function value and (x^*, y^*, z^*) as optimal frame.	

3) Base Vertex Algorithm (BV): For n requested viewing zones, there are $O(n^2)$ BVs. The BV optimality condition (Lemma 2) allows us to find the optimal frame by checking the candidate frames that have one of their corners coincident with one of the BVs. This means that we can reduce the original 3-D optimization problem in (5) to $O(n^2)$ 1-D optimization problems. Defining $p_i(z) = \text{Area}(r_i \cap c), a_i = \text{Area}(r_i) = w_i l_i$, the 1-D optimization problem is to find

$$\max_z s(z) = \sum_{i=1}^{n} u_i (p_i(z)/a_i) \min((z_i/z)^b, 1) \qquad (12)$$

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

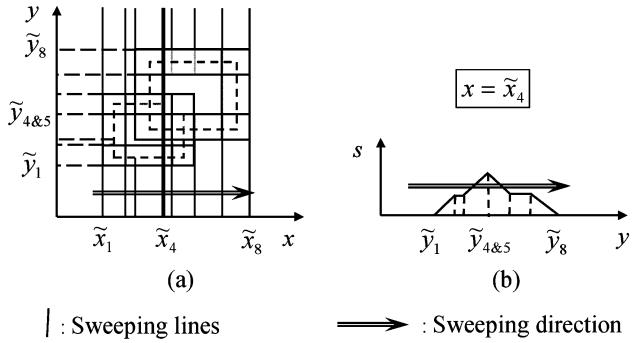


Fig. 8. Illustration of the PVT algorithm using the example in Fig. 6. (a) Shows how we sweep along the x-axis to dissect the 2-D optimization problem into $O(n)$ 1-D optimization problems. (b) Shows how we solve the 1-D optimization problem by traversal over the ordered vertices for $x = \tilde{x}_4$. (a) Sweeping along the x-axis. (b) Sweeping along the y-axis.

C. Algorithms for Continuous Resolution

If a satellite camera has a variable and continuous resolution, there are infinitely many "plateau vertices." Instead, we can use the BV optimality condition to reduce the 3-D optimization problem to $O(n^2)$ 1-D optimization problems with respect to variable z. We then show that each 1-D optimization problem can be dissected into $O(n)$ piecewise polynomial functions; for each, we can find an optimum in time $O(n)$. Using incremental computation and a diagonal sweep, we show how to improve the running time to $O(n^3)$.

In step iii) of the PVT algorithm, we traverse the vertical facet boundaries of the plateaus one by one. Fig. 8 illustrates how it works using the example of two requested viewing zones. For each vertical edge, we find the maximum. Using Lemma 2, we know that this procedure will find an optimal solution. It remains to be shown how much time is required to solve the resulting problem of finding

$$\max_y s(x, y, z)$$

for given x and z. This special optimization problem can be solved in $O(n)$ with a sorted sequence $\{\tilde{y}_1, \tilde{y}_2 \ldots, \tilde{y}_{4n}\}$. The objective function is a "summation" of n plateaus, which is shown in Fig. 6. For fixed x and z, this piecewise linear function only changes slope at $\{\tilde{y}_i\}, i = 1, \ldots, 4n$. For each vertex $\tilde{y}_i$, we know how much the slope will change after crossing the vertex. We can find the maximum objective value by walking over all ordered vertices $\{\tilde{y}_i\}$ from the one side to the other side on the line $x = \tilde{x}_i$. This process only takes $O(n)$. Therefore, step iii) of the algorithm will take $O(n^2)$, proving the following theorem.

Theorem 1: We can solve the SFS problem in time $O(n^2 m)$ for n users and m resolution levels.

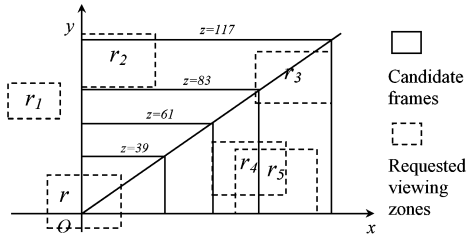


Fig. 9. Example of the 1-D optimization problem with respect to z. In this example, we assume $l(z) = 4z, w(z) = 3z, b = 1$, and $u_i = 1$ for $i = 1, \ldots, n$.

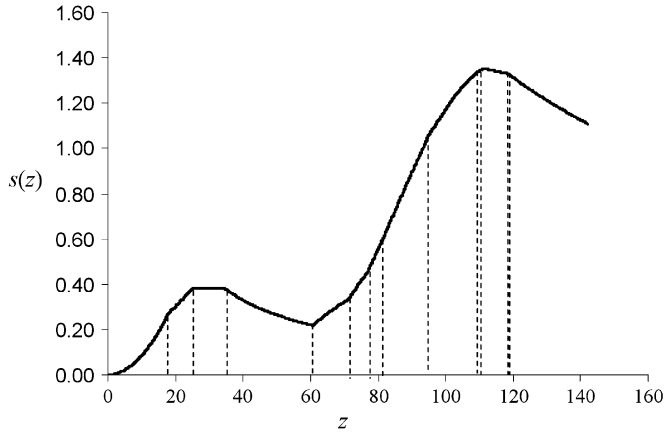


Fig. 10. Reward function for the example in Fig. 9 as a function of image resolution.

To study the 1-D maximization problem in (5), consider a BV. For simplicity, we assume that the BV is at the origin. Moreover, we assume that the BV is coincident with the lower left corner of the candidate frame. (The BV in Fig. 9 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 are coincident with the BV 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 $s(z)$. Fig. 10 shows the function for the example in Fig. 9.

a) Critical z Values and Intersection Topologies: The function $s(z)$ is a piecewise smooth function (Fig. 10), 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).

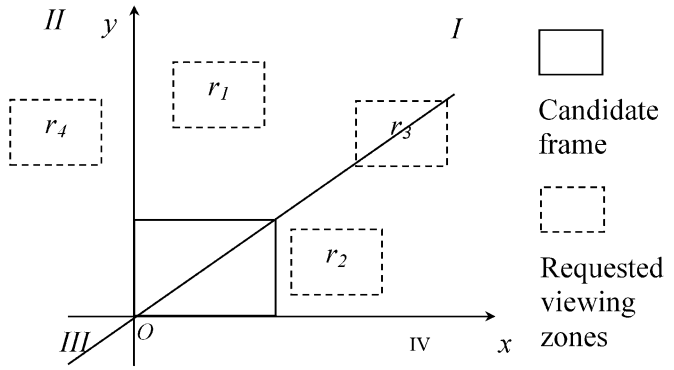


Fig. 11. Examples for "fundamental rectangles." In this figure, r_1 and r_2 are type (a) rectangles, r_3 is a type (b) rectangle, and r_4 is a type (o) rectangle.

Definition 1: A critical z value is a z value at which $s(z)$ changes its closed-form representation, which is caused either by intersection topology changes or crossing one of the $z_i, i = 1, \ldots, n$ when the candidate frame changes its size.

Let $Z_c(x_v, y_v)$ be the set of critical z values for BV (x_v, y_v). From (4), we see that the nonsmoothness comes from the nonsmoothness 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, form a subset $Z''_c(x_v, y_v)$. The former type is easy to deal with because it occurs at $z = z_i, i = 1, \ldots, n$. Therefore, $Z'_c(x_v, y_v) = \{z_i \,|\, i = 1, \ldots, n\}$, so $|Z'_c(x_v, y_v)| = n$. Note that $Z'_c(x_v, y_v)$ is the same for all BVs (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 four 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 $+\infty$.

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.

Fig. 11 gives examples of these three classes of fundamental rectangles.

As shown in Fig. 12, as z increases:

- $p_i(z)$ for a class (o) rectangle always remains type 0;
- $p_i(z)$ for a 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;

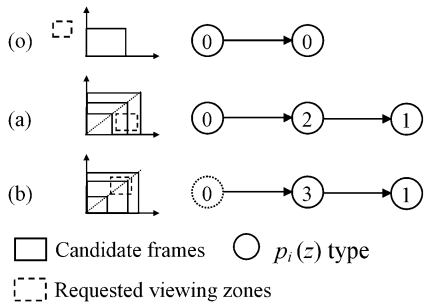


Fig. 12. Change of $p_i(z)$ for the three classes of requested viewing zones when z gradually increases from 0^+ to $+\infty$.

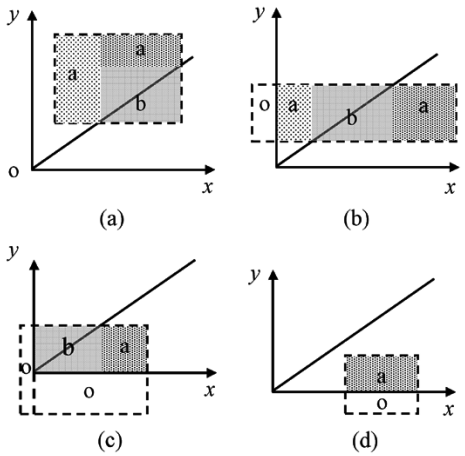


Fig. 13. Examples of four requested viewing zone decomposition cases.

- $p_i(z)$ for a class (b) rectangle can start either from type 3 or type 0 depending on whether the bottom left corner of the rectangle is coincident 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 (Fig. 13).

- If the requested viewing zone intersects only the diagonal—as in Fig. 13(a)—then it can be decomposed into two class (a) rectangles and one class (b) rectangle.
- If the requested viewing zone intersects the diagonal and exactly one positive coordinate axis—as in Fig. 13(b)—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—as in Fig. 13(c)—then it can be decomposed into one class (a) rectangle, one class (b) rectangle, and two class (o) rectangles.
- If the requested viewing zone intersects only one positive coordinate axis—as in Fig. 13(b)—then it can be decomposed into a class (a) rectangle and a class (o) rectangle.

As we can see from Fig. 13, 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 4: For each BV, the z-axis can be partitioned into $O(n)$ segments, on which $s(z)$ is smooth.

Lemma 4 answers our question 2) from the previous section.

c) Optimization Problem on a Segment: With the knowledge of questions 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 5: For each segment, $s(z)$ is a polynomial function with six coefficients g_0, g_1, g_2, g_3, g_4, and g_5

$$s(z) = g_0 z^{-b} + g_1 z^{-b+1} + g_2 z^{-b+2} + g_3 + g_4 z + g_5 z^2. \quad (13)$$

Proof: For a BV (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 $\tilde{r}_i, i = 1, \ldots, k$. Let us define set $S' = \{i | z_i \le z'\}$ and set $S'' = \{i | z_i \ge z''\}$. From the definition of critical z value, we know that $z_i \notin (z', z'')$ for $i = 1, \ldots n$ so that $S' \cup S'' = \{1, \ldots, k\}$ and $S' \cap S'' = \emptyset$. Therefore, (12) becomes

$$s(z) = \sum_{i \in S''} u_i p_i(z)/a_i + \sum_{i \in S'} u_i (p_i(z)/a_i)(z_i/z)^b. \quad (14)$$

We also define S_j to 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

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

We can perform a similar transform for the second term of (14)

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

Combining them, we get (13). ■

The Proof of Lemma 5 shows that (12) can be converted into (13) in $O(n)$ time. The maximum of (13) can be found in constant time because it does not depend on n. Combining Lemma 4 and Lemma 5 yields the BV algorithm.

Theorem 2: The BV solves the SFS problem in $O(n^4)$ time.

Lemma 5 is only applicable to the CRR metric defined in (4). For general CRR metrics that consist of continuous elementary functions, (13) remains a continuous elementary function so that the complexity of computing the optimal will not go higher.

2) BV With Incremental Computing (BV-IC): The inner loop in the BV 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 recompute the coefficients entirely if we solve the $O(n)$ subproblems 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 BV by the following subroutine.

The BV-IC algorithm improves the running time.

Theorem 3: The BV with incremental computing (BV-IC) algorithm solves the SFS problem in $O(n^3 \log n)$.

3) BV With Incremental Computing and Diagonal Sweeping (BV-IC-DS): In the outer loop of the BV-IC algorithm, the sorting of $Z_c(x_v, y_v)$ for each BV is the dominating factor. Is it necessary to sort critical z values repeatedly for each BV? Recall that $Z_c(x_v, y_v)$ 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 we call a critical point in Fig. 14(a). Each critical point corresponds to the point that the candidate frame begins to intersect a requested viewing zone or when the intersection between the candidate frame and some requested viewing zone ends. This is a geometric interpretation for critical z values. Fig. 14(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

Base Vertex (BV) Algorithm

For each base vertex (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^*.	

Base Vertex with Incremental Computing (BV-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	

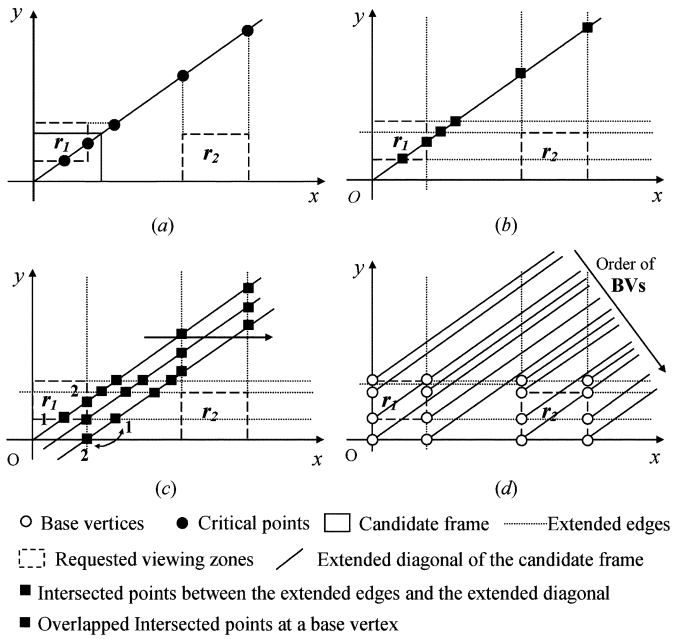


Fig. 14. (a) $Z_c''(x_v, y_v)$ for a case with two requested viewing zones. (b) $Z_e''(x_v, y_v)$ is a set of intersection points between the extended diagonal of the candidate frame and the extended edges. (c) Two intersection points switch order only at a BV formulated by the intersection of the two extended edges that generate the two intersection points. (d) Sorting BVs in this order can reduce the sorting cost of the algorithm.

edges, which is illustrated in Fig. 14(b). $Z_e''(x_v, y_v)$ also depends on BV (x_v, y_v). As shown in Fig. 14(a) and (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 $Z_e''(x_v, y_v)$.

Fig. 14(c) illustrates a nice property of the sorted sequence of points in $Z_e''(x_v, y_v)$. In this figure, we have an ordered sequence of intersected points at the extended diagonal that starts from the origin O. Let the point closest to the origin be point 1 and the next closest be 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 BV by definition. Let us define this BV to be the adjacent BV to the BV at the origin. Point 1 and point 2 switch their order at the adjacent BV [i.e., the gray rectangle in Fig. 14(c)]. This phenomenon shows that if we have a sorted sequence of the intersection points at a BV, we can get the sorted sequence at an "adjacent base vertex" in constant time.

This reduces the sorting cost from $O(n \log n)$ to $O(n)$ if we handle the BVs in a diagonal order: imagine there is a sweep line that has the same slope as the extended diagonal and an intercept at $+\infty$; we decrease the intercept and stop at each BV. As shown in Fig. 14(d), we solve the subproblem for the BV when the sweeping line stops. This yields the following BV-IC-DS algorithm.

Theorem 4: The BV with an incremental computing and diagonal sweeping (BV-IC-DS) approach solves the SFS problem in $O(n^3)$ time.

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

Sort Z'_c	$O(n \log n)$
Sort base vertices in sweeping order	$O(n^2 \log n)$
Sort $Z''_e(x_v, y_v)$ for the first base vertex	$O(n \log n)$
For each base vertex (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-C.2.	$O(n)$
End For	
Report the maximum $s(c)$ and the corresponding c^*.	

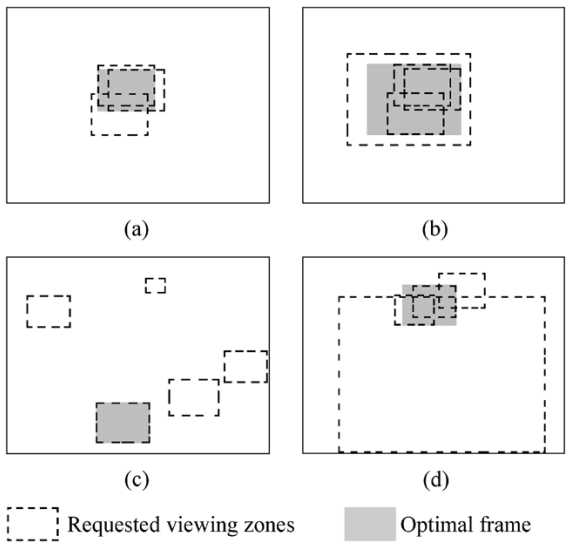


Fig. 15. Examples of computed optimal frames (shown in gray). We set $b = 1$ and $u_i = 1$ for all requests and use the PVT algorithm from Section IV-B. We have ten different resolution levels and set $l(z) = 4z$ and $w(z) = 3z$. We put the optimal reward S in the caption of subfigures.

V. Simulation Results

We have implemented all algorithms. The discrete resolution algorithms were implemented on a PC with a 950-MHz AMD Athlon central processing unit (CPU) and 1-GB random-access memory (RAM). The machine runs under Redhat Linux 7.1 and the algorithms are programmed in Java. The algorithms for variable and continuous resolutions were implemented using Microsoft Visual C++ on a PC laptop with 1.6-GHz Pentium-M and 512-MB RAM.

Fig. 15 shows the results for four different sets of inputs. As we can see from Fig. 15(a) and (b), the optimal frame does not necessarily have one corner coincident with a corner of a requested viewing zone. However, one corner of the optimal frame does coincide with one of the BVs. Fig. 15(b) has three requested viewing zones that are exactly the same as those in (a) and one more big requested viewing zone. It is interesting to see how the optimal frame changes after the big requested viewing zone joined in the system. Fig. 15(c) shows that if all input rectangles fall far away from each other, the algorithm functions as a chooser and selects one input rectangle with the highest utility value as the output. Fig. 15(d) shows that a large requested viewing zone does not necessarily yield a large optimal frame. Note that the results can depend on the utility u_i, or weight assigned to each requested viewing zone. Our examples illustrate cases where utility is constant across requested viewing zones.

We use random inputs for testing. 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 interest, 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 create six random numbers. One of them is used to determine which seed the request will be associated with. Two of them are 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 of the request. Each data point in Fig. 16 is the average of ten runs.

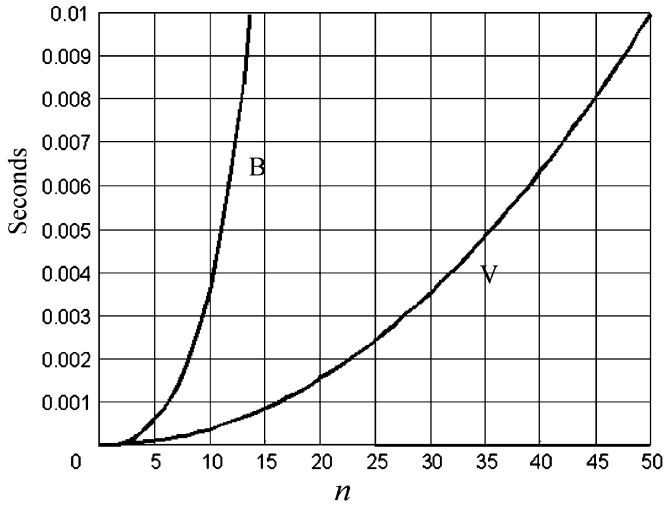


Fig. 16. Speed comparison for the two algorithms from Section IV-B. Curve B refers to the brute-force algorithm, and curve V refers to the PVT algorithm. Each data point is based on the average of ten runs with random inputs.

Fig. 16 compares the computation speed of the two algorithms presented in Subsection IV-B for a fixed resolution level $(m = 1)$. The results clearly confirm the theoretical analysis.

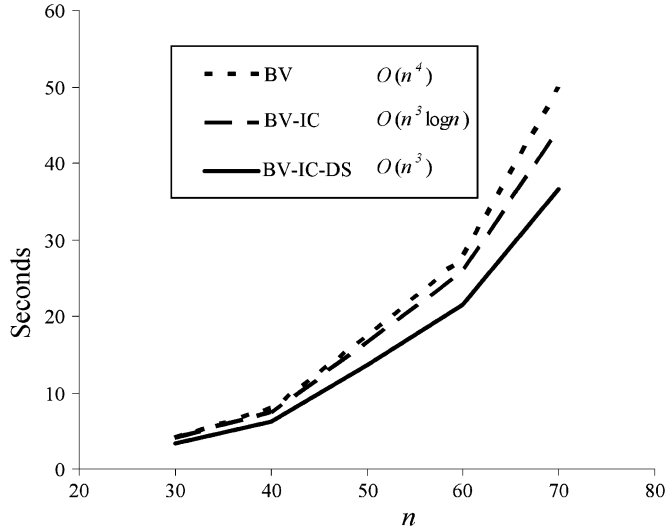


Fig. 17. Computation speed comparison for random inputs. Each data point in Fig. 17 is an average of ten trials with different random inputs, where the same random inputs are used to test all three algorithms.

Fig. 17 illustrates the speed difference between BV, BV-IC, and BV-IC-DS algorithms. These timing results are also consistent with the theoretical analysis.

VI. Conclusions and Future Work

To automate satellite camera control, this paper introduces the SFS problem: find the satellite camera frame parameters that maximize reward during each time window. 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 and a satellite with m discrete resolution levels, we give an SFS algorithm that computes optimal frame parameters in $O(n^2m)$. For satellites with continuously variable resolution $(m = \infty)$, we give an SFS algorithm that computes optimal frame parameters in $O(n^3)$. We have implemented all algorithms and compared computation speeds on randomized input sets.

In future work, we will consider approximation algorithms for the SFS problem and generalizations 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 interested in generalizing the SFS problem to cases where requested viewing zones are nonrectangular; for example, convex or concave polygons. There are also a number of multiple-frame extensions to the problem: in one case, p cameras are available to servo simultaneously, which is related to the p-center facility location problem. As another extension, we are interested in finding, for a given set or sequence of frame requests, the optimal sequence of p camera frames. This is a variant of the Traveling Salesperson's problem, even when reward is related to latency, and may be amenable to approximation techniques.

Acknowledgment

The authors would like to thank the reviewers for insightful suggestions. They thank A. Levandowski for bringing the near real-time satellite imaging industry to our attention. Also, thanks to M. Overmars, V. Koltun, S. Har-Peled, D. Volz, 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. Finally, thanks also to W. Zheng, K. "Gopal" Gopalakrishnan, R. Alterovitz, I. Y. Song, L. Xiao, J. Voung, and J. Wang, and for their contributions to the Alpha Lab. at the University of California, Berkeley.

References

[1] P. Agarwal and J. Erickson, "Geometric range searching and its relatives," in *Contemporary Mathematics*, B. Chazelle, J. E. Goodman, and R. Pollack, Eds. Providence, RI: Amer. Math. Soc., 1999, vol. 23, Advances in Discrete and Computational Geometry, pp. 1–56.

[2] E. Bensana and M. Lemaître, "Earth observation satellite management," *Constraints: Int. J.*, vol. 4, no. 3, pp. 293–299, 1999.

[3] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *J. Comput. Syst. Sci., Special Issue on STOC*, vol. 60, no. 3, pp. 630–659, 2000.

[4] A. Z. Broder, "On the resemblance and containment of documents," in *Proc. Compress. Complex. Seq. 1997, IEEE Comput. Soc.*, Positano, Italy, Mar. 1998, pp. 21–29.

[5] D. J. Cannon, "Point-and-direct telerobotics: Object level strategic supervisory control in unstructured interactive human-machine system environments," Ph.D. dissertation, Dept. Mech. Eng., Stanford Univ., Stanford, CA, Jun. 1992.

[6] N. Chong, T. Kotoku, K. Ohba, K. Komoriya, N. Matsuhira, and K. Tanie, "Remote coordinated controls in multiple telerobot cooperation," in *Proc. IEEE Int. Conf. Robotics Automation*, vol. 4, Apr. 2000, pp. 3138–3343.

[7] D. Eppstein, "Fast construciton of planar two-centers," in *Proc. 8th ACM-SIAM Symp. Discrete Algorithms*, Jan. 1997, pp. 131–138.

[8] V. Gabrel, "Improved Linear Programming Bounds Via Column Generation for Daily Scheduling of Earth Observation Satellite," LIPN, Univ. 13 Paris, Nord, France, Tech. Rep., Jan. 1999.

[9] Y. Gabriely and E. Rimon, "Competitive on-line coverage of grid environments by a mobile robot," *Comput. Geom.: Theory Appl.*, vol. 24, no. 3, pp. 197–224, Apr. 2003.

[10] K. Goldberg and B. Chen, "Collaborative control of robot motion: Robustness to error," in *Proc. Int. Conf. Intelligent Robots Systems (IROS)*, vol. 2, Oct. 2001, pp. 655–660.

[11] K. Goldberg, D. Song, and A. Levandowski, "Collaborative teleoperation using networked spatial dynamic voting," *Proc. IEEE*, vol. 91, no. 3, pp. 430–439, Mar. 2003.

[12] R. Grossi and G. F. Italiano, "Efficient cross-trees for external memory," *External Mem. Alg. Visual.*, pp. 87–106, 1999.

[13] R. Grossi and G. F. Italiano, "Revised version of Efficient Cross-Trees for External Memory," Univ. Pisa, Pisa, Italy, Tech. Rep. TR-00-16, Oct. 2000.

[14] D. Habet and M. Vasquez, "Saturated and consistent neighborhood for selecting and scheduling photographs of agile earth observing satellite," in *Proc. 5th Metaheuristics Int. Conf.*, Kyoto, Japan, Aug. 2003.

[15] N. G. Hall and M. J. Magazine, "Maximizing the value of a space mission," *Eur. J. Oper. Res.*, no. 78, pp. 224–241, 1994.

[16] D. Halperin, M. Sharir, and K. Goldberg, "The 2-center problem with obstacles," *J. Algorithms*, vol. 32, pp. 109–134, Jan. 2002.

[17] S. Har-Peled, V. Koltun, D. Song, and K. Goldberg, "Efficient algorithms for shared camera control," presented at the Proc. 19th ACM Symp. Computational Geometry, San Diego, CA, Jun. 2003.

[18] S. A. Harrison and M. E. Price, "Task scheduling for satellite based imagery," in *Proc. 18th Workshop U.K. Planning Scheduling*. Manchester, U.K., Dec. 1999, pp. 64–78.

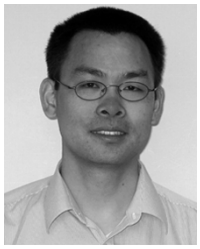
[19] Quickbird 2 is successfully launched on 18 Oct. 2001, Quickbird 2. [Online]. Available: http://www.crisp.nus.edu.sg/research/tutorial/quickbird.htm

[20] RADARSAT-1: Price List $US, MDA. [Online]. Available: http://www.rsi.ca/products/sensor/radarsat/rs1_price_us.asp

[21] M. Lemaitre, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerosp. Sci. Technol.*, vol. 6, pp. 367–381, Jul. 2002.

[22] Q. Liu, D. Kimber, J. Foote, and C. Liao, "Multichannel video/audio acquisition for immersive conferencing," presented at the Proc. IEEE Int. Conf. Multimedia Expo (ICME), Baltimore, MD, Jul. 2003.

[23] Q. Liu, D. Kimber, L. Wilcox, M. Cooper, J. Foote, and J. Boreczky, "Managing a camera system to serve different video requests," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, vol. 2, Lausanne, Switzerland, Aug. 2002, pp. 13–16.
[24] M. McDonald, D. Small, C. Graves, and D. Cannon, "Virtual collaborative control to improve intelligent robotic system efficiency and quality," in *Proc. IEEE Int. Conf. Robotics Automation*, vol. 1, Apr. 1997, pp. 418–424.
[25] N. Megiddo and K. J. Supowit, "On the complexity of some common geometric location problems," *SIAM J. Comput.*, vol. 13, pp. 182–196, Feb. 1984.
[26] PLEIADES [Online]. Available: http://smsc.cnes.fr/pleiades/gp_satellite.htm
[27] RADARSAT-2 [Online]. Available: http://www.radarsat2.info/index.asp
[28] D. Song and K. Goldberg, "Sharecam part I: Interface, system architecture, and implementation of a collaboratively controlled robotic webcam," presented at the Proc. IEEE/RSJ Int. Conf. Intelligent Robots, Las Vegas, NV, Nov. 2003.
[29] D. Song, A. Pashkevich, and K. Goldberg, "Sharecam part II: Approximate and distributed algorithms for a collaboratively controlled robotic webcam," presented at the Proc. IEEE/RSJ Int. Conf. Intelligent Robots, Las Vegas, NV, Nov. 2003.
[30] D. Song, A. F. van der Stappen, and K. Goldberg, "Exact and distributed algorithms for collaborative camera control," in *Algorithmic Foundations of Robotics V, Springer Tracts in Advanced Robotics 7*, J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, Eds. Berlin, Germany: Springer-Verlag, Dec. 2002, pp. 167–183.
[31] D. Song, A. F. van der Stappen, and K. Goldberg, "An exact algorithm optimizing coverage-resolution for automated satellite frame selection," presented at the Proc. IEEE Int. Conf. Robotics Automation, New Orleans, LA, May 2004.
[32] TerraSAR-X [Online]. Available: http://www.spotimage.fr/html/_167_240_570_684_.php
[33] 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.*, vol. 20, no. 2, pp. 137–157, 2001. Appl..
[34] R. C. Veltkamp and M. Hagedoorn, "Shape similarity measures, properties, and constructions," presented at the Proc. 4th Int. Advances in Visual Information Systems Conf., Lyon, France, Nov. 2–4, 2000.
[35] D. Zhang, V. J. Tsotras, and D. Gunopulos, "Efficient aggregation over objects with extent," in *Proc. 21st ACM Int. SIGACT-SIGMOD-SIGART Symp. Principles Database Systems*, Madison, WI, Jun. 2002, pp. 121–132.
[36] R. C. Veltkamp and M. Hagedoorn, "Shape similarity measures, properties, and constructions," in *Advances in Visual Information Systems, Proc. 4th Int. Conf.*, vol. 1929, Lyon, France, Nov. 2–4, 2000, Springer, pp. 467–476.



Dezhen Song (S'02–M'04) received the Ph.D. degree from the University of California, Berkeley, in 2004.

Currently, he is an Assistant Professor with Texas A&M University, College Station. His research area is networked robotics, computer vision, optimization, and stochastic modeling.

Dr. Song received the Kayamori Best Paper Award of the 2005 IEEE International Conference on Robotics and Automation (with J. Yi and S. Ding).

A. Frank van der Stappen (M'03) received the Ph.D. degree from Utrecht University, Utrecht, The Netherlands, in 1994 and the M.Sc. degree from Eindhoven University of Technology, Eindhoven, The Netherlands, in 1988.

Currently, he is an Associate Professor with the Department of Information and Computing Sciences, Utrecht University. His research focuses on geometric algorithms for manipulation of parts in automated assembly processes, manipulation of deformable material, and exact and approximate motion planning.

Ken Goldberg (S'84–M'90–SM'98–F'05) received the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, in 1990.

Currently, he is Professor of Industrial Engineering and Operations Research with the University of California, Berkeley, with an appointment in Electrical Engineering and Computer Science. From 1991 to 1995 he taught at the University of Southern California, Los Angeles, and was Visiting Faculty at the Massachusetts Institute of Technology Media Lab, Cambridge, in 2000. in 2000. He and his students work in two areas: geometric algorithms for automation and networked robots.

Dr. Goldberg is Founding Chair of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING Advisory Board. He received the National Science Foundation Young Investigator Award in 1994, the National Science Foundation's Presidential Faculty Fellowship in 1995, the Joseph Engelberger Award for Robotics Education in 2000, and the IEEE Major Educational Innovation Award in 2001.