Approximate Algorithms for a Collaboratively Controlled Robotic Camera

Dezhen Song, *Member, IEEE*, and Kenneth Y. Goldberg, *Fellow, IEEE*

***Abstract*—Deployed as a natural environment observatory or a surveillance device, a remote networked robotic pan-tilt-zoom camera needs to be controlled by simultaneous frame requests from both online users and *in situ* sensors such as motion detectors. This paper presents algorithms that are capable of finding a camera frame that optimizes a measure of total satisfaction over all requests, which is a generalized version of the single frame-selection problem proposed by Song *et al.* in 2006.We present a lattice-based approximation algorithm; given n requests and approximation bound ϵ, we analyze the tradeoff between solution quality and the corresponding computation time, and prove that the algorithm runs in $O(n/\epsilon^3)$ time. We also develop a branch-and-bound-like implementation that reduces the constant factor of the algorithm by more than 70%. We have implemented the algorithms, and numerical experiment results conform to our analysis. Field experiments of the proposed algorithms have been conducted in the past three years. The proposed algorithms have been deployed successfully in a variety of real world applications including natural environment observation, building construction monitoring, and the surveillance of public space.**

***Index Terms*—Natural environment observation, pan-tilt-zoom camera, teleoperation, telerobotics.**

NOMENCLATURE

Z	$= [\underline{z}, \overline{z}]$ is a set of feasible values of image resolution/camera zoom range.
(x, y)	Center position of a camera frame.
z	Camera frame size, $z \in Z$.
c	Camera frame $c = [x, y, z]$.
c^*	Optimal camera frame $c^* = [x^*, y^*, z^*]$.
n	Number of request frames.
z_i	Image resolution of the ith request, $z_i \in Z, i = 1, \ldots, n$.
r_i	Request $i, r_i = [T_i, z_i]$, where T_i is an arbitrary closed region, and it takes constant time to compute its area, $i = 1, \ldots, n$.
s_i	Satisfaction function of request i.
s	Overall satisfaction.
ϵ	Approximation bound.
$\mathrm{Area}(\cdot)$	Function that computes the area of the closed polygon.
w, h, g	Camera pan, tilt, and zoom range.

Manuscript received May 26, 2006; revised February 24, 2007. This paper was recommended for publication by Associate Editor L. S. Victor and Editor K. Lynch upon evaluation of the reviewer's comments. This work was supported in part by the National Science Foundation under Grant IIS-0534848/0535218 and Grant IIS-0643298, in part by Panasonic Research, in part by Intel Corporation, and in part by UC Berkeley's Center for Information Technology Research in the Interest of Society (CITRIS).

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

K. Y. Goldberg is with the Departments of Industrial Engineering and Operations Research (IEOR) and Electrical Engineering and Computer Science (EECS), University of California, Berkeley, CA 94720 USA (e-mail: goldberg@ieor.berkeley.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2007.907488

I. INTRODUCTION

THE RECENT development of low-power networked robotic cameras provides low-cost interactive access to remote sites. Robotic pan-tilt-zoom cameras can cover a large region without using excessive communication bandwidth. With applications in natural environment observation or surveillance, a single robotic camera is often concurrently controlled by many online users and networked *in situ* sensors such as motion detectors. Since there are multiple simultaneous requests in a dynamic environment, an optimal camera frame needs to be computed quickly to address the resource contention problem and hence to achieve the best observation or surveillance results. In [23], this is proposed as a *single frame selection* (SFS) problem when requests are rectangular regions. However, a majority of requests are not necessarily rectangular in many applications. The shapes of requests are usually determined by factors such as the shapes of objects in the scene and the coverage of *in situ* sensors. On the other hand, the existing algorithms give exact optimal solutions and are not scalable due to their high complexity. A new class of fast and approximate algorithms are favorable for this generalized SFS problem that prefers speed to accuracy.

As illustrated in Fig. 1, the input of a generalized SFS problem is a set of n requests. The output of the problem is a camera frame that maximizes the satisfaction of all requests, which can be intuitively understood as the best tradeoff between the coverage and the detail/resolution of the camera frame. We present a lattice-based approximation algorithm; given n requests and an approximation bound ϵ, we analyze the tradeoff between solution quality and the required computation time and prove that the algorithm runs in $O(n/\epsilon^3)$ time. We also develop a branch-and-bound (BnB)-like approach that can reduce the constant factor of the algorithm by more than 70%. We have implemented the algorithm, and the speed-testing results conform to our design and analysis. We have successfully tested the algorithm in a variety of situations such as construction monitoring, the surveillance of public space, and natural environment observation.

II. RELATED WORK

The SFS problem is related to problems in online robots, facility location in operations research, spatial databases, and video conferencing.

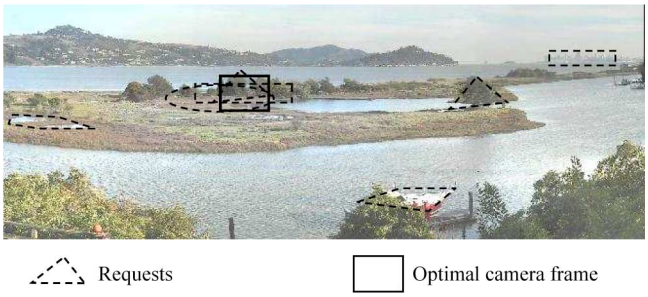


Fig. 1. Illustration of the frame-selection problem. The panorama describes the camera's full workspace (reachable field of view). Each user/sensor positions a request as a dashed closed region in the panorama. Based on these requests, the algorithm computes an optimal camera frame (shown with a rectangle) and moves the camera accordingly. In this case, the wildlife-observing camera is pointed at Richardson Bay Audubon Center and Sanctuary, which is located inside San Francisco Bay.

The Internet provides a low-cost and widely available interface that can make physical resources accessible to a broad range of participants. Online robots, controllable over the Internet, are an active research area [6]. The pan-tilt-zoom camera in the frame-selection problem can be viewed as a robot with 3 DOF that is collaboratively controlled by both online users and *in situ* sensors. Inputs from each user/sensor must be combined to generate a single control stream for the robot. In the taxonomy proposed by Chong *et al*. [3], these are multiple operator single robot (MOSR) systems. An Internet-based MOSR system is described by McDonald *et al*. [2], [16]. 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 in MOSR systems can be found in [5] and [8]. In [7] and [8], Goldberg *et al*. proposed the "spatial dynamic voting" (SDV) interface. SDV collects, displays, and analyzes sets of spatial votes from multiple online operators at their Internet browsers using a Gaussian point clustering algorithm developed to guide the motion of a remote human "Tele-Actor."

Operator inputs are closed regions rather than points in the frame-selection problem. The frame-selection problem is a nonlinear optimization problem with a nondifferentiable objective function. The structure of the problem is closely related to the planar p-center facility location problem that was proven to be NP-complete by Megiddo and Supowit [17]. Using a geometric approach, Eppstein [4] gives $O(n \log^2 n)$ algorithm for the planar two-center problem. Halperin *et al*. [11] give an algorithm for the 2-center problem with m obstacles that runs in expected time $O(m \log^2(mn) + mn \log^2 n \log(mn))$. In almost all nonlinear mathematical programming approaches, a constrained optimization problem is converted to a series of unconstrained problems using barrier or penalty methods. Line search is then used to solve the unconstrained optimization problems. Although there are many different ways of guiding search directions and step size, most of these methods are based on derivatives [19].

Evaluating the objective function for a given candidate, camera frame is related to a special instance of the general "box aggregation" query over spatial objects in database research [25]. The spatial objects could be points, intervals, or rectangles. Aggregation over points is a special case of orthogonal range search queries from computational geometry. Agarwal and Erickson [1] provide a review of geometric range searching and related topics. Grossi and Italiano [9], [10] propose 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.

BnB is a general problem-solving paradigm, which is especially useful for finding optimal solutions to most of the NP-hard combinatorial problems [26]. BnB can efficiently reduce the search space as computation proceeds. Lin *et al*. [13] applied BnB to solve the protein backbone nuclear magnetic resonance peaks assignment problem. Mitchell and Brochers analyzed BnB performance with respect to 0–1 mixed integer nonlinear programming problems [18]. A comprehensive review of BnB can be found in Papadimitriou and Steiglitz [20].

In the multimedia literature, Liu *et al*. combine a fixed panoramic camera with a robotic pan-tilt-zoom camera for collaborative video conferencing [15]. They [14] address a frame-selection problem by partitioning the solution space into small nonoverlapping rectangular 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 the discrepancy in probability-based estimation. Their approach is approximate and relies on how the small nonoverlapping 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 that we address in our paper, where regions of interests are represented as arbitrary closed regions, which is especially appropriate to cameras in outdoor unstructured environments where regions of interest may not necessarily be rectangular.

We introduced the frame-selection problem for robotic webcams in a series of papers: exact solution with discrete zoom [22], approximate solution with continuous zoom [21], approximate solution with fixed zoom [12], and exact solution with continuous zoom and rectangular requests with fixed-aspect ratio [24] or variable-aspect ratio [23]. This journal paper collects and expands our results on approximate solutions [21]. It also extends the problem formulation to arbitrary requests as long as the area of their coverage can be computed in a constant time.

III. Problem Definition

In this section, we formulate the frame-selection problem, as an optimization problem, finding the camera frame that maximizes total request satisfaction.

Let c be a vector of control parameters for the pan-tilt-zoom camera with a fixed base. Frame $c = [x, y, z]$, where x, y specify the center point of the camera frame that corresponds to the pan and tilt, and z specifies the size of the rectangular camera frame.

For a camera with a fixed-aspect ratio of $k_x : k_y$, we define z such that the four corners of frame $c = [x, y, z]$ are located at

$$\left(x \pm \frac{k_x z}{2}, y \pm \frac{k_y z}{2}\right). \tag{1}$$

Frame c uniquely defines a rectangular camera frame because the camera has a fixed-aspect ratio. It is worth mentioning that z uniquely defines camera zoom and image resolution and is sometimes referred to as image resolution or camera zoom. A smaller z means a smaller coverage of the camera frame and actually corresponds to a higher zoom. Since charge coupled device (CCD) camera sensors have a fixed number of pixels, a smaller z also refers to higher resolution. Let $Z = [\underline{z}, \bar{z}]$ be the range of camera frame size. Since the camera cannot have infinite resolution and infinite coverage, therefore,

$$\underline{z} > 0 \text{ and } \bar{z} < \infty. \tag{2}$$

For frame $c = [x, y, z]$ with aspect ratio of 4:3, the width of the frame is $4z$, the height of the frame is $3z$, and the area of the frame is $12z^2$. To facilitate our analysis, the aspect ratio of 4:3 is used as the default-aspect ratio. Our algorithm can be easily adapted to cameras with different aspect ratios.

Let w and h be the camera pan and tilt ranges, respectively. Let $\Theta = \{(x, y) : x \in [0, w], y \in [0, h]\}$ be the set of all reachable x, y pairs. Set $C = \Theta \times Z = \{[x, y, z] | [x, y] \in \Theta, z \in Z\}$ as the feasible region of the problem.

Request i is defined as $r_i = [T_i, z_i]$, where T_i specifies the closed requested region and $z_i \in Z$ specifies the desired image resolution using the same units as z in c. The only requirement for T_i is that its area of coverage can be computed in constant time. Given n requests, the system computes an optimal frame c^* that will best satisfy the set of requests.

As described in [23], request "satisfaction" is a coverage-resolution ratio (CRR) function. It is based on how a requested region compares with a candidate camera frame. The metric is a scalar $s_i \in [0, 1]$, the level of "satisfaction" that request i receives. Request i gets no satisfaction if the candidate frame does not intersect r_i: $s_i = 0$ when $c \cap r_i = \emptyset$. In this paper we abuse the set intersection operator $\cap$ as the intersection between the coverage of c and the coverage of r_i. Request i is perfectly satisfied, $s_i = 1$ when T_i is located inside c (full coverage), and $z_i \geq z$ because it means the resolution of the camera frame is equal to or better than the requested z_i. When there is a partial overlap

$$s_i(r_i, c) = \frac{\text{Area}(r_i \cap c)}{\text{Area}(r_i)} \min\left(\frac{z_i}{z}, 1\right). \tag{3}$$

If z is bigger, the candidate frame will be bigger. A sufficiently large z can define a candidate frame that covers all requests, $\text{Area}(r_i \cap c)/\text{Area}(r_i) = 1$ for $i = 1, \ldots, n$. However, request satisfaction is not necessarily high because the request wants to see the camera frame at a desired resolution. The term $\min(z_i/z, 1)$ characterizes this desire; it reaches its maximum of 1 if the candidate frame resolution is better than the requested resolution $z \leq z_i$. We do not consider the case of oversatisfaction in this version. More accurately speaking, the notation r_i in (3) should be T_i. We abuse r_i to make the equation easy to be associated with the request.

For n total requests, let the total request satisfaction be

$$s(c) = \sum_{i=1}^{n} s_i(r_i, c). \tag{4}$$

We want to find c^*, the value of c that maximizes $s(c)$. Since $c = [x, y, z]$, we now have a maximization problem

$$\arg\max_c s(c); \qquad \text{subject to } c \in C. \tag{5}$$

We next present two lattice-based approximation algorithms to solve it.

IV. Algorithms

We begin with a lattice-based approximation algorithm, and derive formal approximation bounds that characterize the tradeoff between speed and accuracy. We then present a BnB-like approach to reduce the constant factor of the lattice-based algorithm.

A. Algorithm I: Exhaustive Lattice Search

Since the camera needs to respond to fast moving objects, a fast approximate solution is more desirable than a slow exact solution. We propose an algorithm that searches a regular lattice for an approximate solution $\tilde{c}$.

Defining the lattice as the set of points with coordinates

$$L = \{(pd, qd, rd_z) | pd \in [0, w], qd \in [0, h] \\ rd_z \in [\underline{z}, \bar{z} + 2d_z], p, q, r \in \mathcal{N}\} \tag{6}$$

where d is the spacing of the pan and tilt samples, d_z is the spacing of the zoom, and p, q, r are positive integers.

To find $\tilde{c}$, we evaluate all $(wh/d^2)(g/d_z)$ candidate points, where $g = \bar{z} - \underline{z}$. According to (4), it takes $O(n)$ computing time to determine the satisfaction for a single candidate frame c. The total amount of computation is

$$O((wh/d^2)(g/d_z)n). \tag{7}$$

How good is the approximate solution in comparison to the optimal solution? Specifically, what is the tradeoff between solution quality and computation speed? Let c^* be an optimal solution. Let ϵ characterize the comparative ratio of the objective values for the two solutions

$$s(\tilde{c})/s(c^*) \; = \; 1 - \epsilon. \tag{8}$$

Since (5) defines a maximization problem, $s(c^*)$ is always greater than or equal to $s(\tilde{c})$, so $0 \leq \epsilon < 1$. As $\epsilon \to 0$, $s(\tilde{c}) \to s(c^*)$.

We will establish theorems that give an upper bound ϵ_u such that $\epsilon \leq \epsilon_u(d, d_z)$ for given d and d_z. This characterizes the tradeoff between solution quality and computation speed. We first prove the lemmas based on two observations.

1) As illustrated in Fig. 2, consider a set of requests r_i, each with zoom level z_i. Now consider two candidate frames for the camera c_a, c_b, with z_a, z_b such that for all i, $r_i \subset c_a \subset$

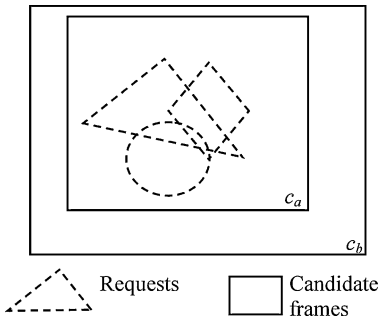


Fig. 2. Example illustrating the lower bound on solution quality.

c_b[1] and $z_i < z_a < z_b$. This case provides a lower bound where $s(c_b)/s(c_a) = z_a/z_b$. In general, some requests r_i will be included in c_b but outside c_a, which will only increase the ratio.

2) Now, consider the smallest frame on the lattice that contains an optimal frame. Its size is a function of the size of the optimal frame z^*, d, and d_z, as derived in Lemma 2.

We now prove these formally in the general case to obtain a bound on solution quality.

Lemma 1: For two candidate frames $c_a = [x_a, y_a, z_a]$ and $c_b = [x_b, y_b, z_b]$, if c_a is within c_b, then $s(c_b)/s(c_a) \geq z_a/z_b$.

Proof: Recall that r_i is the ith request. Let us assume the following.

1) $a_i = \text{Area}(r_i)$.
2) $p_{ai} = \text{Area}(c_a \cap r_i)$.
3) $p_{bi} = \text{Area}(c_b \cap r_i)$, then $p_{bi} \geq p_{ai}$.
4) $I_a = \{i | r_i \cap c_a \neq \emptyset\}$ be the set of requests that intersect with frame c_a.
5) $I_b = \{i | r_i \cap c_b \neq \emptyset\}$ be the set of requests that intersect with frame c_b; $I_a \subseteq I_b$.
6) I' be the set of requests that intersect with frame c_a and are bigger than c_a, $I' = \{i | i \in I_a \text{ and } z_i \geq z_a\}$.
7) I'' be the set of requests that intersect with frame c_a and are bigger than c_b, $I'' = \{i | i \in I_a \text{ and } z_i \geq z_b\}$; $I'' \subseteq I' \subseteq I_a$ because $z_b \geq z_a$.

We can classify the proof into two cases:

1) $s(c_b) \geq s(c_a)$; here, we have $s(c_b)/s(c_a) \geq 1 \geq z_a/z_b$. The lemma holds.
2) $s(c_b) < s(c_a)$; this is nontrivial. Let us focus on this case in the rest of the proof.

We have

$$s(c_a) = \sum_{i=1}^{n} (p_{ai}/a_i) \min(z_i/z_a, 1)$$

and because $I_a \subseteq I_b$

$$s(c_b) = \sum_{i \in I_b} (p_{bi}/a_i) \min(z_i/z_b, 1) \geq \sum_{i \in I_a} (p_{bi}/a_i) \min(z_i/z_b, 1).$$

[1]We use set operators $\subset$ and $\cap$ to represent the coverage relationship between a camera frame and a requested region.

Therefore

$$\begin{aligned} s(c_b)/s(c_a) &\geq \frac{\sum_{i \in I_a} (p_{bi}/a_i) \min(z_i/z_b, 1)}{\sum_{i \in I_a} (p_{ai}/a_i) \min(z_i/z_a, 1)} \\ &\geq \frac{\sum_{i \in I_a} (p_{ai}/a_i) \min(z_i/z_b, 1)}{\sum_{i \in I_a} (p_{ai}/a_i) \min(z_i/z_a, 1)} \\ &= \frac{\sum_{i \in I''} (p_{ai}/a_i) + \sum_{i \in I_a - I''} (p_{ai}/a_i)(z_i/z_b)}{\sum_{i \in I'} (p_{ai}/a_i) + \sum_{i \in I_a - I'} (p_{ai}/a_i)(z_i/z_a)}. \end{aligned} \quad (9)$$

We also know that a generic function $f(x) = \frac{x+a'}{x+b'}$ is an increasing function of x if $b' \geq a' \geq 0$ and $x \geq 0$. We know that $1 > s(c_b)/s(c_a)$. If we simultaneously decrease the nominator and the denominator of (9) by a nonnegative quantity $\sum_{i \in I''} (p_{ai}/a_i)$, then, the following holds:

$$s(c_b)/s(c_a) \geq \frac{\sum_{i \in I_a - I''} (p_{ai}/a_i)(z_i/z_b)}{\sum_{i \in I' - I''} (p_{ai}/a_i) + \sum_{i \in I_a - I'} (p_{ai}/a_i)(z_i/z_a)}. \quad (10)$$

Defining $S_l = \sum_{i \in I' - I''} (p_{ai}/a_i)(z_i/z_a)$, we know that

$$z_i/z_a \geq 1, \forall i \in I' - I''.$$

Then

$$\sum_{i \in I' - I''} (p_{ai}/a_i) \leq S_l.$$

If we use S_l to replace $\sum_{i \in I' - I''} (p_{ai}/a_i)$ in the denominator of (10), then we have

$$\begin{aligned} s(c_b)/s(c_a) &\geq \frac{\sum_{i \in I_a - I''} (p_{ai}/a_i)(z_i/z_b)}{S_l + \sum_{i \in I_a - I'} (p_{ai}/a_i)(z_i/z_a)} \\ &= \frac{\sum_{i \in I_a - I''} (p_{ai}/a_i)(z_i/z_b)}{\sum_{i \in I_a - I''} (p_{ai}/a_i)(z_i/z_a)} \\ &= \frac{(1/z_b) \sum_{i \in I_a - I''} (p_{ai}/a_i) z_i}{(1/z_a) \sum_{i \in I_a - I''} (p_{ai}/a_i) z_i} \\ &= \frac{1/z_b}{1/z_a} = \frac{z_a}{z_b}. \end{aligned}$$

■

Now, we are ready to find the smallest frame on the lattice that contains the optimal frame.

Lemma 2: Recall that d is the spacing of the lattice, and d_z is the spacing for zoom levels. For any frame $c = [x, y, z] \in C$, there exists $c' = [x', y', z'] \in L$ such that c' is the smallest frame on the lattice that ensures c is within c', which implies

$$|x - x'| \leq d/2 \text{ and } |y - y'| \leq d/2 \quad (11)$$

$$z' \leq \left\lceil \frac{3z + d}{3d_z} \right\rceil d_z. \quad (12)$$

If we choose $d = 3d_z$, then

$$z' \leq z + 2d_z. \quad (13)$$

Proof: The center (point B in Fig. 3) of the given frame c must have four neighboring lattice points. Without loss of generality,

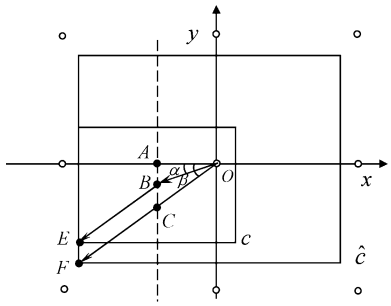


Fig. 3. Relationship between frame c and the smallest frame $\hat{c}$ on the lattice that encloses it. $\alpha = \angle AOB$, $\beta = \angle AOC$, and the frame c is centered at point B.

let us assume that the nearest lattice point of the center is the top-right lattice point, which is point O in Fig. 3. Other cases can be proven by symmetry.

Since frame c' is the smallest frame on the lattice that ensures c is within c', (x', y') has to be the closest neighboring lattice point of (x, y) on the Θ plane, which implies that (11) has to be true.

Recall that d is the spacing of the lattice. To ensure the point O is the nearest lattice point, (11) states that point B must satisfy the following constraints:

$$|\overline{OB}| \sin\alpha \le d/2, \quad \text{and} \quad |\overline{OB}| \cos\alpha \le d/2. \tag{14}$$

Let us define frame $\hat{c} = [x', y', \hat{z}]$ to be the smallest frame containing frame c such that $(x', y') \in \Theta$ and $\hat{z} \in \mathcal{R}^+$. In other words, frame $\hat{c}$ is located at the lattice point (x, y), but with continuous zoom $\hat{z}$. It is not difficult to find the relationship between c' and $\hat{c}$

$$z' = \lceil \hat{z}/d_z \rceil d_z. \tag{15}$$

Since point F at (x_F, y_F) is the bottom-left corner of frame $\hat{c}$, and point E at (x_E, y_E) is the bottom-left corner of frame c, the condition that frame c is located inside frame $\hat{c}$ is equivalent to following conditions:

$$x_F \le x_E \text{ and } y_F \le y_E. \tag{16}$$

Since the frames are iso-oriented rectangles and have the same aspect ratio, their diagonal lines have to be parallel to each other

$$\overline{BE} \parallel \overline{OF}.$$

Therefore, when $0 \le \alpha \le \beta$, $\overline{BE}$ is always above $\overline{OF}$, and if $x_F = x_E$, then $y_F \le y_E$. The boundary conditions for $\hat{c}$ can be simplified as follows.

Case 1: $x_F = x_E$, if $0 \le \alpha \le \beta$.
Case 2: $y_F = y_E$, if $\beta \le \alpha \le \pi/2$.

Fig. 3 describes case 1. We draw a vertical line at point B, which intersects the x-axis at point A and $\overline{OF}$ at point C. Since $x_F = x_E$, we know that $\overline{EF} \parallel \overline{AC}$. Therefore, we have $|\overline{CF}| = |\overline{BE}|$ and

$$|\overline{OC}| = |\overline{OF}| - |\overline{CF}| = |\overline{OF}| - |\overline{BE}|. \tag{17}$$

Also, since $\overline{AC} \perp \overline{OA}$, we have

$$|\overline{OC}| \cos\beta = |\overline{OB}| \cos\alpha$$
$$\Rightarrow (|\overline{OF}| - |\overline{BE}|) \cos\beta = |\overline{OB}| \cos\alpha.$$

According to (14)

$$\Rightarrow (|\overline{OF}| - |\overline{BE}|) \cos\beta \le d/2.$$

The aspect ratio of the frame is 4:3$\Rightarrow \cos\beta = 4/5$

$$\Rightarrow |\overline{OF}| \le |\overline{BE}| + 5d/8.$$

Similarly, we can get $|\overline{OF}| \le |\overline{BE}| + 5d/6$ from case 2. Combining the two cases, we know

$$|\overline{OF}| \le |\overline{BE}| + 5d/6.$$

Since $|\overline{OF}| = 5\hat{z}/2$ and $|\overline{BE}| = 5z/2$

$$\hat{z} \le z + d/3.$$

Putting it into (15), we get

$$z' \le \left\lceil \frac{3z + d}{3d_z} \right\rceil d_z.$$

If we choose $d = 3d_z$, (12) can be simplified as

$$z' \le \left\lceil \frac{z}{d_z} \right\rceil d_z + d_z \le z + 2d_z. \tag{18}$$

■

Remark 1: It is worth mentioning that the result of Lemma 1 depends on the camera-aspect ratio. Taking a close look, it is not difficult to find that the constant 3 in the proof is from $\min(k_x, k_y)$ for a camera with an aspect ratio of 4:3 ($k_x = 4$, $k_y = 3$). Therefore, the choice of lattice spacing in *x*–*y* plane

$$d = \min(k_x, k_y) d_z = 3d_z.$$

in (18) also depends on the camera-aspect ratio. It is not difficult to extend it to cameras with different aspect ratios because it only changes the constant factors in the analysis.

Theorem 1: Recall $\underline{z}$ is the smallest allowable z value and $d = 3d_z$. The approximation factor ϵ of the deterministic lattice-based algorithm is bounded with some constant ϵ_u, $0 \le \epsilon \le \epsilon_u$, where

$$\epsilon_u = \frac{2d_z}{\underline{z} + 2d_z}.$$

.

Proof: Recall the following.

1) $c^* = [x^*, y^*, z^*]$ is the optimal frame.
2) $c' = [x', y', z']$ is the closest lattice point with the smallest zoom level that ensures c^* is within c'.
3) $\tilde{c} = [\tilde{x}, \tilde{y}, \tilde{z}]$ is the lattice point found by the approximation algorithm.

It is worth mentioning that we do not know c^* and c'. The geometric enclosure relationship between c^* and c' is known. We also know that c' is just one of the points in lattice L. We note that $\tilde{c}$ is the solution to: $\arg\max_{c \in L} s(c)$. Since $c' \in L \subset C$, we know that $s(c') \le s(\tilde{c})$. Therefore

$$s(c') \le s(\tilde{c}) \le s(c^*).$$

Applying Lemma 1

$$1-\epsilon = s(\tilde{c})/s(c^*) \geq s(c')/s(c^*) \geq \frac{z^*}{z'}.$$

Applying (13) of Lemma 2, we have

$$z' \leq z^* + 2d_z.$$

Using this result, we have

$$1-\epsilon \geq \frac{z^*}{z^* + 2d_z}.$$

On the other hand, we know $z^* \geq \underline{z}$; so

$$1-\epsilon \geq \frac{\underline{z}}{\underline{z}+2d_z} \leftrightarrow \epsilon \leq \frac{2d_z}{\underline{z}+2d_z}.$$

■

Theorem 1 states that the approximation bound $2d_z/(\underline{z}+2d_z)$ is a monotonically increasing function of d_z. It characterizes the tradeoff between accuracy and computation speed for the lattice-based approximation algorithm.

Algorithm 1: Exhaustive Lattice Search

input : requests, r_i, $i = 1, ..., n$, approximation bound ϵ
output: approximate solution $\tilde{c}$
compute the appropriate lattice spacing: choosing $d = 3d_z$, according to Theorem 1, we set

$$\epsilon = \frac{2d_z}{\underline{z}+2d_z} \Rightarrow d_z = \frac{1}{2}\left(\frac{\epsilon}{1-\epsilon}\right)\underline{z}. \quad (19)$$

/* This is the maximum d_z that ensures the objective function value is bounded above $(1-\epsilon)s(c^*)$. */
$s(\tilde{c}) = 0$; $O(1)$
for *each lattice point* $c = (x, y, z)$ **do**
  Compute the objective function value s(c); $O(n)$
  if $s(c) > s(\tilde{c})$ **then**
    $s(\tilde{c}) = s(c)$; $O(1)$
    $\tilde{c} = (x, y, z)$; $O(1)$
Report $\tilde{c}$ and $s(\tilde{c})$; $O(1)$

The relationship between solution quality and computation speed is summarized by Theorem 2.

Theorem 2: We can solve the frame-selection problem in $O(n/\epsilon^3)$ for a given approximation bound ϵ.

Proof: Since $d = 3d_z$ and $d_z = \frac{1}{2}(\frac{\epsilon}{1-\epsilon})\underline{z}$, we need to evaluate all $(wh/d^2)(g/d_z) = whg/(9/4)\{\epsilon/(1-\epsilon)\}^3\underline{z}^3$ points. According to (4), each point will take $O(n)$ time. Removing the constants, ϵ approaches zero; thus, the computation time approaches $O(n/\epsilon^3)$. ■

B. Algorithm II: BnB Implementation

In the lattice-based approximation algorithm, we evaluate the objective function at each lattice point. However, we may not need to check them all. The proof of Theorem 1 implies the following corollary. ■

Corollary 1: Given that frame $\hat{c}$ is currently the best known solution, if a candidate frame $c = [x, y, z]$ does not satisfy the condition

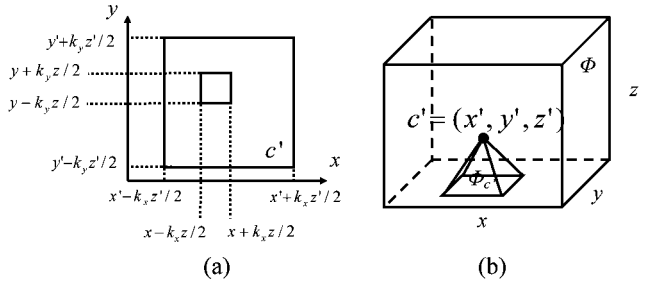


Fig. 4. Illustration of the solution space formed by frames contained in the given frame c'. The constants k_x and k_y are determined by the camera-aspect ratio. For an aspect ratio of 4:3, $k_x = 4$ and $k_y = 3$.

$$s(c) \geq s(\hat{c})(\underline{z}/z) \quad (20)$$

then, the candidate frame does not contain any optimal frame.

Proof: Assume that $c^* = [x^*, y^*, z^*]$ is an optimal solution, if the candidate frame contains it. Then, according to Lemma 1, the following is true:

$$\frac{s(c)}{s(c^*)} \geq \frac{z^*}{z} \geq \underline{z}/z.$$

■

Since c^* is the optimal solution, then $s(\hat{c}) \leq s(c^*)$. Therefore, (20) is true if the candidate frame contains an optimal frame. The corollary is true.

Corollary 1 allows us to improve the lattice-based algorithm by using a BnB-like approach. We check if the condition in (20) is satisfied. If not, we know that the optimal frame is not contained in the candidate frame. Hence, we can delete frames that are contained in the candidate frame. Let the candidate frame be $c' = [x', y', z']$, then, the frames contained in c' define a subset of the solution space, i.e., $\Phi_{c'}$.

Recall that $k_x : k_y$ is the camera-aspect ratio in (1). As illustrated in Fig. 4(a), if a frame (x, y, z) is contained in c', it has to satisfy the following set of conditions:

$$\begin{aligned} x - k_x z/2 &\geq x' - k_x z'/2 \\ x + k_x z/2 &\leq x' + k_x z'/2 \\ y - k_y z/2 &\geq y' - k_y z'/2 \\ y + k_y z/2 &\leq y' + k_y z'/2 \\ (x, y, z) &\in \Phi. \end{aligned} \quad (21)$$

Therefore, $\Phi_{c'} = (x, y, z)|(x, y, z)$ satisfies (21). Recall that the solution space Φ is a 3-D rectangle. Fig. 4(b) illustrates that the shape of $\Phi_{c'}$ is a pyramid within the 3-D rectangle and has its top located at c'. The volume of the pyramid is determined by its height z' of the candidate frame. A larger z' means a larger candidate frame, which leads to a bigger cut in Φ if the candidate frame does not satisfy Corollary 1. This suggests that we should evaluate the lattice points in descending order relative to the z-axis.

Fig. 5 illustrates how to perform the BnB-like search using a $3 \times 3 \times 3$ lattice. We divide the lattice points into different layers with respect to their z values. The search starts with the topmost layer and follows a descending order in z. In this lattice,

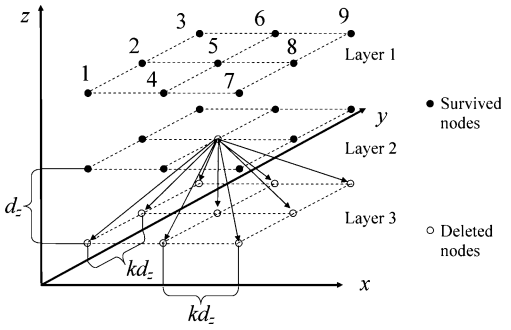


Fig. 5. Illustration of the BnB-like approach.

we set $d = 3d_z$, which will be used as the default setting in the rest of the section.

In each layer, we evaluate the objective function at each lattice point in lexicographic order (i.e., the numbered sequence in layer 1 in Fig. 5). After the evaluation, we test if the point satisfies the condition in Corollary 1. If so, we refer to this point as a survived node. Otherwise, this is a deleted node. If a node is deleted, it will also cause some nodes in the next layer to be deleted because of the shape of the pyramid. We refer to those nodes in the next layer as the child nodes of the deleted node. Since we choose $d = kd_z$ (where $k = \min(k_x, k_y)$. Therefore, $k = 3$ for most cameras, which have an aspect ratio of 4:3.), we have the following lemma.

Lemma 3: If a lattice point (x, y, z) is deleted and is not a boundary node, then its nine child nodes in the next layer (frames with zoom $z - d_z$)

$$\begin{array}{lll} (x - kd_z, y - kd_z), & (x - kd_z, y), & (x - kd_z, y + kd_z), \\ (x, y - kd_z), & (x, y), & (x, y + kd_z), \\ (x + kd_z, y - kd_z), & (x + kd_z, y), & (x + kd_z, y + kd_z) \end{array}$$

should also be deleted.

Lemma 3 can be proven by checking if all nine child nodes are located inside the frame (x, y, z), and their union region covers no more area than the frame (x, y, z) does. Fig. 5 also illustrates this relationship. The central node in layer 2 is deleted and causes all nine children to be deleted. If the deleted node is a boundary node, the number of children is less than nine. If one node concludes that it is not viable and the neighbor node concludes that it is viable, then, the shared child nodes should be nonviable that leads to great computation savings.

Lemma 3 unveils an iterative scheme that we can use to cut the solution space. Recall that we need to evaluate the lattice points in descending order in z, and follow a lexicographic order in the x–y plane. Recall that $\hat{c}$ is currently the best known solution. Initially, we set $\hat{c}$ to be an arbitrary feasible frame and every node on the lattice to be a survived node. Combining the aforementioned information, we can reduce the computational effort required and present the BnB-like approach.

Algorithm 2: BnB-like Approach

```
input : requests, r_i, i = 1, ..., n, approximation bound ε
output: approximate solution c̃
compute d and d_z following Algorithm 1;                O(1)
s(c̃) = 0;                                               O(1)
for each lattice point c = (x, y, z) do
    if the node was deleted then
        if not the lowest layer then
            delete its child notes in lower layer
    else
        Compute the objective function value s(c);      O(n)
        if Equation (20) holds then
            if s(c) > s(c̃) then
                s(c̃) = s(c);                            O(1)
                c̃ = (x, y, z);                          O(1)
        else
            if not the lowest layer then
                delete its child notes in lower layer
Report c̃ and s(c̃);                                      O(1)
```

In the worst case scenario, this approach does not improve the complexity bound. For example, if all requested viewing zones are identical to the accessible region, the approach is not able to cut any computation. Since such worst cases are rare, the approach has its value. We will show the numerical test results in Section V.

V. Experiments

We have implemented the algorithms in Sections IV-A and IV-B. The algorithms are programmed in C++ that is compatible with both Microsoft Visual C++ and Gnu C++. Both numerical experiments and field applications have been used to test the performance. During numerical experiments, we test algorithm speed under different parameter settings, such as the number of requests n and the approximation bound ϵ. The extensive field tests are conducted over three years across a variety of applications. We first report the results of the numerical experiments.

A. Numerical Experiments

The testing computer in the numerical experiments is a PC laptop with a 1.6-GHz Intel Centrino CPU and 512 MB RAM. The operating system is Windows XP. Fig. 1 shows the algorithm's sample result for an example with seven requests.

During the speed test, triangular random inputs are used. The random inputs are generated for testing in two steps. First, we generate four random points, which are uniformly distributed in the reachable field of view of the robotic camera. The four points represent locations of interest that are referred to as seeds. For each seed, we use a random number to generate a radius of interest. Then, we generate the requested regions in the second step. We generate a requested region using eight random

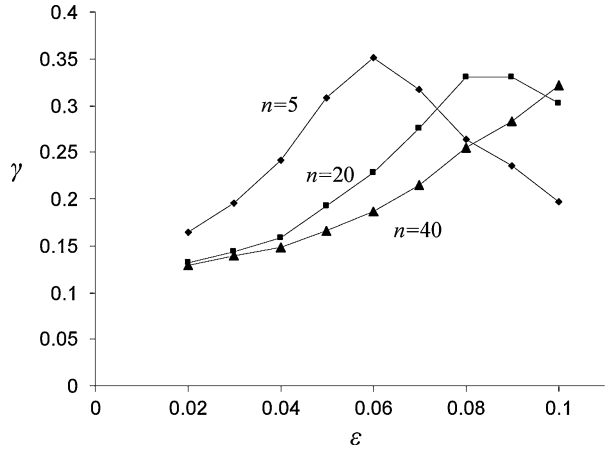


Fig. 6. Efficiency of the BnB-like approach. Smaller γ is more desirable. Recall that n is the number of requests.

numbers: one is used to determine which seed the request is associated with; six are used to generate the location of the request (two random numbers per vertex for a triangular request), which is located within the corresponding radius of the associated seed; and the remaining random number is used to generate the resolution for the request.

To measure the effectiveness of the BnB-like approach in Section IV-B, we define

$$\gamma = \frac{\text{computation time using BnB}}{\text{computation time using exhaustive lattice search}} \quad (22)$$

as the performance index variable. Equation (22) shows that a smaller γ is more desirable because it means that less computation time is needed. Each data point in Fig. 6 is an average of five iterations using different sets of random requests. The result in Fig. 6 can be classified into two cases according to ϵ values.

If $\epsilon \leq 0.05$, the curves in Fig. 6 appear to have two trends. The first trend is that γ decreases as the number of requests n increases. The second trend is that γ decreases as ϵ decreases. Both trends are very desirable because it means BnB becomes more efficient as more computation is needed.

If $\epsilon > 0.05$ and $n = 5$, the overhead cost of the BnB-like approach dominates the computation, and hence, γ decreases as ϵ increases. When n increases, the point for γ to change its trend from increasing to decreasing does not show up until ϵ is very large. However, it is very rare for us to set $\epsilon > 0.1$ because it is faster enough for our applications as illustrated later. Therefore, it is not an interesting trend for us. Nevertheless, the BnB-like approach can cut the constant factor of the algorithm by more than 70% when $\epsilon < 0.05$, which speeds up computation by more than three times.

It is also worth mentioning that the effectiveness of the BnB-like approach also depends on $\underline{z}$ according to (19). If $\underline{z} = 0$, the BnB-like approach fails to cut the computation time. Fortunately, $\underline{z} = 0$ means that the camera has an infinite resolution, which cannot happen according to (2).

We have also compared the approximation algorithm to the exact algorithm in [23]. Since the exact algorithm takes $O(n^3)$ computation time and only accepts rectangular requests, we have modified our algorithm accordingly to ensure a fair comparison. Fig. 7 illustrates the speed comparison between the approximation algorithm and the exact algorithm. The implementation is the basic lattice-based algorithm in Section IV-A. We set $w = h = 500$, $\underline{z} = 40$, $\bar{z} = 80$, and $k = 3$. The result conforms to our analysis. The computation time of our approximation algorithm is linear in the number of requests, and the slope of the line is determined by the approximation bound ϵ.

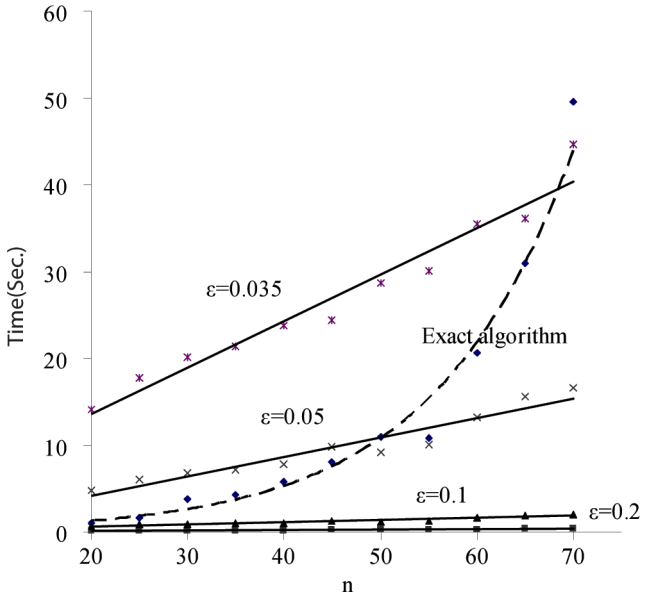


Fig. 7. Speed comparison between the basic lattice-based approximation algorithm and the exact algorithm in [23].

B. Field Tests

Our algorithms have also been continuously tested in the field since September 2002. Applications of the algorithm include construction monitoring, public space surveillance, and natural environment observation. Table I summarizes the five testing sites and the corresponding durations and applications. The algorithm runs on a server with dual 2.5 GHz Intel Xeon CPUs and 2 GB RAM. The operating system is Mandrake Linux. According to our records (as of February 22, 2006), we have received more than 301 340 requests.

For the duration when the system is not idle, the experimental data show that requests follow an interesting 95–5 distribution pattern for human inputs, which means only 5% of requests occupy 95% of system time and 95% of users compete for the remaining 5% of the time. Our conjecture is that users tend to log on to the system when there are some activities going on such as a political rally, crane operations for construction of building, and a moving wild animal.

The irregular traffic pattern favors our fast approximation algorithm because the system can satisfy a majority of users in a timely manner. We know that camera servo time is around 1 s. If it takes more than 1 s to compute an optimal frame, the

TABLE I
FIELD-TEST HISTORY

	Duration	Location	Application
1	Sep. 2002 - May 2003	Alpha Lab, UC Berkeley	system testing
2	Jun. 2003 - Sep. 2004	Hearst-Mining Hall construction, UC Berkeley	construction monitoring
3	Aug. 2004 - Sep. 2004	Sproul Plaza, UC Berkeley	public surveillance
4	Sep. 2004 - present	CITRIS II building construction, UC Berkeley	construction monitoring
5	Nov. 2005 - present	Richardson Bay Audubon Center and Sanctuary, CA	natural environment observation

significant delay can make the system difficult to retain users and track dynamic events. Since accuracy in optimality is not a big concern on web cameras, we set $\epsilon = 0.1$ to favor speed. The ability of choosing the tradeoff between speed and accuracy is another advantage of the approximate algorithm.

Even when the number of requests is small (95% of time with 5% users), the exact algorithm still has the difficulty to match the computation speed because of its overhead in the construction of complex data structures.

Although the number of concurrent requests from human users is rarely more than 50, the number of requests generated from sensory inputs (i.e., requests from motion detection) can easily be more than 100. The exact algorithm is not able to handle such amount of inputs.

During the tests, the algorithm successfully combines real-time inputs from online users, preprogrammed commands, and sensory inputs, and drives the robotic camera automatically and efficiently. The cameras used in our systems include Panasonic HCM 280, Canon VCC3, and Canon VCC4. After analyzing the data from multiple deployments, we have an interesting finding that the average user satisfaction level is inversely proportional to the number of concurrent activities on the site. In other words, the users tend to spread their requests evenly across the activities. Our latest research goal is to incorporate the algorithm into sensor/human-driven natural environment observation (visit us at http://www.c-o-n-e.org for details).

VI. CONCLUSION AND FUTURE WORK

We present new algorithms for the frame-selection problem: controlling a single robotic pan-tilt-zoom camera based on n simultaneous requests. With approximation bound ϵ, the algorithm runs in $O(n/\epsilon^3)$ time. We also introduce a BnB-like approach that can reduce the constant factor of the algorithm by more than 70%. The algorithms have been implemented and tested in both numerical experiments and extensive field applications. The results of the numerical experiments conform to our design and analysis. The three-and-half-year field tests in five different sites have demonstrated that the algorithm has successfully addressed the problem of effective collaborative camera control in a variety of applications.

In the future, we plan to investigate the problems involving more than one camera frame. As an expansion of the SFS problem, the multiple frame-selection problem has to consider both camera travel time and camera task allocation in addition to request satisfaction. The optimization problem is nontrivial given its complex geometric and combinatorial nature. We will report our findings as new results emerge.

ACKNOWLEDGMENT

The authors would like to thank E. Brew and C. Newmark for providing camera installation sites; A. F. van der Stappen, M. Overmars, V. Koltun, S. Har-Peled, D. Volz, N. Amato, A. Lim, S. Rao, D. Zhang, D. Halperin, J. Lunar, P. Wright, and R. Baccy for insightful discussions and feedback; and N. Qin, H. N. Lee, Y. Xu, Z. Goodwin, B. Green, H. Wang, and C. Kim for their contributions to the Networked Robotics Laboratory, Texas A&M University.

REFERENCES

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

[2] 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.

[3] 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. Robot. Autom.*, Apr. 2000, vol. 4, pp. 3138–3343.

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

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

[6] K. Goldberg and R. Siegwart, Eds., *Beyond Webcams: An Introduction to Online Robots*. Cambridge, MA: MIT Press, 2002.

[7] K. Goldberg, D. Song, Y. Khor, D. Pescovitz, A. Levandowski, J. Himmelstein, J. Shih, A. Ho, E. Paulos, and J. Donath, "Collaborative online teleoperation with spatial dynamic voting and a human "tele-actor," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2002, vol. 2, pp. 1179–1184.

[8] 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.

[9] R. Grossi and G. F. Italiano, "Efficient cross-trees for external memory," in *External Memory Algorithms and Visualization*, J. Abello and J. S. Vitter, Eds. Providence, RI: Amer. Math. Soc., 1999, pp. 87–106.

[10] 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.

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

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

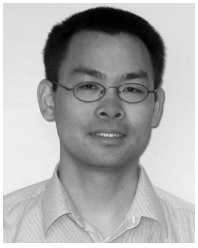
[13] G. Lin, D. Xu, Z. Chen, T. Jiang, J. Wen, and Y. Xu, "An efficient branch-and-bound algorithm for the assignment of protein backbone NMR peaks," in *Proc. IEEE Comput. Soc. Bioinform. Conf. (CSB 2002)*, pp. 165–174.

[14] Q. Liu, D. Kimber, J. Foote, and C. Liao, "Multichannel video/audio acquisition for immersive conferencing," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Baltimore, MD, Jul. 2003, pp. 45–48.

[15] 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)*, Lausanne, Switzerland, Aug. 2002, vol. 2, pp. 13–16.

[16] 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. Robot. Autom.*, Apr. 1997, vol. 1, pp. 418–424.

[17] 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.
[18] J. Mitchell and B. Borchers, "A comparison of branch and bound and outer approximation methods for 0-1 MINLPS," *Comput. Oper. Res.*, vol. 24, pp. 699–701, 1997.
[19] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*. New York: McGraw-Hill, 1996.
[20] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*. New York: Dover, 1998.
[21] D. Song, A. Pashkevich, and K. Goldberg, "Sharecam part II: Approximate and distributed algorithms for a collaboratively controlled robotic webcam," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots (IROS)*, Las Vegas, NV, Oct. 2003, vol. 2, pp. 1087–1093.
[22] 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.
[23] D. Song, A. F. van der Stappen, and K. Goldberg, "Exact algorithms for single frame selection on multi-axis satellites," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 1, pp. 16–28, Jan. 2006.
[24] D. Song, A. F. van der Stappen, and K. Goldberg, "An exact algorithm optimizing coverage-resolution for automated satellite frame selection," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA 2004)*, New Orleans, LA, pp. 63–70.
[25] D. Zhang, V. J. Tsotras, and D. Gunopulos, "Efficient aggregation over objects with extent," in *Proc. 21th ACM Int. SIGACT-SIGMOD-SIGART Symp. Principles Database Syst. (PODS)*, Madison, WI, Jun. 2002, pp. 121–132.
[26] W. Zhang, "Depth-first branch-and-bound versus local search: A case study," in *Proc. AAAI/IAAI*, 2000, pp. 930–935.



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

He is currently an Assistant Professor with Texas A&M University, College Station. His current research interests include networked robotics, computer vision, optimization, and stochastic modeling.

Dr. Song is the recipient of the Kayamori Best Paper Award of the 2005 IEEE International Conference on Robotics and Automation and the recipient of the NSF Faculty Early Career Development Award in 2007.

Kenneth Y. 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.

He is currently a Professor of industrial engineering and operations research with the University of California, Berkeley, with joint appointments in Department of Electrical Engineering and Computer Science, and the School of Information. He held faculty and visiting positions with the University of Southern California, Los Angeles, and the MIT Media Laboratory, Cambridge, MA. His current research interests include algorithmic automation and networked robots.

Dr. Goldberg is the Vice President of technical activities for the IEEE Robotics and Automation Society (2006–2009) and is the 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 NSF Presidential Faculty Fellowship in 1995, the Joseph Engelberger Award in 2000, and the IEEE Major Educational Innovation Award in 2001.