ShareCam Part II: Approximate and Distributed Algorithms for a Collaboratively Controlled Robotic Webcam

Dezhen Song
IEOR Department
University of California,
Berkeley, CA 94720-1777, USA

Anatoly Pashkevich
State University of Informatics and Radioelectronics
Minsk 220600, Republic of Belarus

Ken Goldberg
IEOR and EECS Departments
University of California,
Berkeley, CA 94720-1777, USA

***Abstract*— ShareCam is a robotic pan, tilt, and zoom web-based camera controlled by simultaneous frame requests from online users. Part I describes the system. This paper, Part II, focuses on algorithms. The ShareCam problem is to find a camera frame that optimizes a measure of total user satisfaction. We present a grid-based approximation algorithm: given camera frame requests from n users, and approximation bound ϵ, we analyze the tradeoff between solution quality and processing speed and prove that the algorithm runs in $O(n/\epsilon^3)$ time. The algorithm can be distributed to run in $O(1/\epsilon^3)$ time at each client and in $O(n + 1/\epsilon^3)$ time at the server. Experiments suggest that performance of the distributed algorithm degrades gracefully as clients fail to complete their part of the computation. ShareCam can be found online at: http://www.tele-actor.net/sharecam/.**

I. Introduction

Consider a robotic webcam set up at a panoramic site such as the San Francisco Bay, Sydney harbor, etc. The camera frame (based on pan, tilt, and zoom) can be remotely adjusted by viewers via the Internet to observe details in the scene. Current control methods rely on queueing, where users have to wait patiently for their turn to operate the camera. "ShareCam" is a new system that eliminates the queue, allowing many users to share simultaneous control of the camera. The Sharecam problem is to find a camera frame that maximizes a measure of user satisfaction.

The "Sharecam" system is an example of Collaborative Telerobotics, where the telerobot is a camera with 3 degrees of freedom. In the taxonomy proposed by Tanie et al. [2], ShareCam is a Multiple Operator Single Robot (MOSR) system. Collaborative Telerobotics is motivated by applications such as education and journalism, where groups of users desire simultaneous access to a single robotic resource. Inputs from each user are combined to generate a single control stream for the robot.

As illustrated in Figure 1, the Sharecam java-based interface includes two image windows. Problem input is the set of requested camera frames from n users. Problem output is a camera frame that maximizes user satisfaction as defined in Section 3.

This work was supported in part by the National Science Foundation under IIS-0113147, by Intel Corporation, by Microsoft Corporation, in part by UC Berkeley's Center for Information Technology Research in the Interest of Society (CITRIS), and by International Research and Exchanges Board (IREX). For more information please contact dzsong@ieor.berkeley.edu, goldberg@ieor.berkeley.edu, or pap@gw.bsuir.unibel.by.

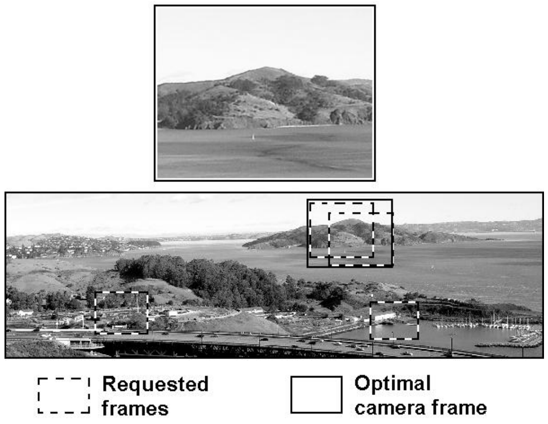

Fig. 1. *Sharecam's Java-based interface on the Internet. The user interface includes two image windows. The lower window displays a fixed "panoramic" image based on the camera's full workspace (reachable field of view). Each user requests a camera frame by positioning a dashed rectangle in the lower window. Based on these requests, the algorithm computes an optimal camera frame (shown with solid rectangle), moves the camera accordingly, and displays the resulting live streaming video image in the upper window.*

II. Related Work

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. In addition to the challenges associated with time delay, supervisory control, and stability, online robots must be designed to be operated by non-specialists through intuitive user interfaces and to be accessible 24 hours a day; see part I [17] for examples of recent projects.

In [5], [4], Goldberg and Chen describe an Internet-based MOSR system that averaged multiple human inputs to simultaneously control a single industrial robot arm. In [6], [7] Goldberg, Song, et al. propose the "Spatial Dynamic Voting" (SDV) interface. The 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".

Sharecam is a system for a robot camera, where operator inputs are frames rather than points. ShareCam suggests a nonlinear optimization problem with a non-differentiable objective function. The structure of the problem is closely related to the planar $p-$center Facility Location problem, which was proved to be NP-complete by Megiddo and Supowit [15]. Using a geometric approach, Eppstein [3] gave an $O(n \log^2 n)$ algorithm for the the planar 2-Center problem. Halperin et al. [10] gave an algorithm for the 2-center problem with m obstacles that runs in randomized expected time $O(m \log^2(mn) + mn \log^2 n \log(mn))$.

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 direction and step size, most of these methods are based on derivatives [16].

Rectangle-related problems in computational geometry include range searching and rectangle intersection. Agarwal and Erickson [1] provide a review of geometric range searching and its related topics. Grossi and Italiano [8], [9] proposed the cross-tree data structure, a generalized version of balanced tree, to speed up range searching in high-dimensional space. Kankanhalli and Franklin [12] developed a parallel algorithm that can compute the area and perimeter of the union of a set of iso-oriented rectangles.

In independent work, Kimber, Liu, Foote et al develop a multi-user robot camera for videoconferencing [13], [14]. Similar to Sharecam, they formulate the frame selection for multiple simultaneous requests as an optimization problem based on position and area of overlap. To solve their version, they propose an approximation based on the bounding boxes of all combinations of user frames. This algorithm requires exponential time and does not provide formal bounds on approximation error.

In [18], Song, van der Stappen, and Goldberg gave the first algorithms for the ShareCam problem. We defined the Generalized Intersection Over Maximum (GIOM) metric for user "satisfaction" based on how the user's requested frame compares with a candidate camera frame and reported an $O(n^2 m)$ exact algorithm for a continuous pan and tilt with discrete m levels of zoom. Har-Peled et al. [11] improved the exact algorithm to $O(mn^{3/2} \log^3 n)$ and proposed a near linear $\epsilon-$approximation algorithm. In the present paper, we relax the assumption that zoom has to be discrete and report approximation algorithms for continuous pan, tilt, and zoom, which runs in $O(n/\epsilon^3)$ time. ShareCam Part I [17], the companion paper presented in this conference, describes system interface, architecture, and implementation.

III. Problem Definition

In this section, we formulate the ShareCam problem as an optimization problem: finding the camera frame that maximizes total user satisfaction.

Let c be a vector of parameters that users can control. In the Sharecam system, frame $c = [x, y, z]$, where x, y specify the center point of the input rectangle, which is corresponding to pan and tilt, and $z = Size(c)$ specifies size of the rectangle, which can be used to control zoom. c defines a rectangular camera frame (the camera has a fixed aspect ratio of 4:3). For frame $c = [x, y, z]$, the width of the frame is $4z$, the height of the frame is $3z$, and the area of the frame is $12z^2$. User i requests a desired frame $r_i = [x_i, y_i, z_i]$. Given requests from n users, the system computes a single global frame c^* that will best satisfy the set of requests.

Define w and h to be the width and height of the panoramic image, let $\Theta = \{(x, y) : x \in [0, w], y \in [0, h]\}$ be the set of all reachable x, y pairs. Let $Z = [z_l, z_u]$ be the range of zoom. Set $C = \Theta \times Z = \{[x, y, z] | [x, y] \in \Theta, z \in Z\}$ as the feasible region of the problem.

As described in [18], user "satisfaction" is a Generalized Intersection Over Maximum (GIOM) function. It is based on how a user's requested frame compares with a candidate camera frame. Recall that r_i is the frame requested by user i, and let $c = [x, y, z]$ be a candidate camera frame. The metric is a scalar $s_i \in [0, 1]$, the level of "satisfaction" that user i receives. User i gets no satisfaction if the candidate frame does not intersect r_i: $s_i = 0$ when $c \cap r_i = \emptyset$. User i is perfectly satisfied when the candidate frame is identical to r_i: $s_i = 1$ when $c = r_i$. When there is partial overlap,

$$s_i(r_i, c) = \frac{Area(r_i \cap c)}{Area(r_i)} \min(\frac{Size(r_i)}{Size(c)}, 1) \qquad (1)$$

If $z = Size(c)$ is bigger, the candidate frame will be bigger. A sufficiently large z can define a candidate frame that covers all requested frames: $\frac{Area(r_i \cap c)}{Area(r_i)} = 1$ for $i = 1, ..., n$. However, user satisfaction is not necessarily high because a user wants to see the requested frame at a desired zoom level. The term $\min(\frac{Size(r_i)}{Size(c)}, 1) = \min(z_i/z, 1)$ characterizes this desire: it reaches its maximum of 1 if the candidate frame is the same or smaller than the requested frame.

Each of n users submits a request. Let the total user satisfaction be

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

We want to find c^*, the value of c that maximizes $s(c)$. Since $c = [x, y, z]$, we now have a maximization problem: $\max_c s(c)$. We next present two grid-based approximation algorithms to solve it.

IV. Algorithms

We begin with a grid-based approximation algorithm and derive formal approximation bounds that characterize the tradeoff between speed and accuracy. We then describe a distributed version of the algorithm.

A. Approximation Algorithm

Since requested frames are drawn by hand by each user, an approximate solution may be acceptable. We propose an algorithm that searches a regular lattice for an approximate solution $\tilde{c}$.

Define the lattice as the set of points with coordinates,

$$\begin{aligned} L &= \{(pd, qd, rd_z) | pd \in [0, w], qd \in [0, h], \\ & \quad rd_z \in [z_l, z_u + 2d_z], p, q, r \in \mathcal{N}\} \end{aligned} \tag{3}$$

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 $g = z_u - z_l$. According to Equation 2, it takes $O(n)$ computing time to determine the satisfaction for a single candidate frame c. The total amount of computation of the algorithm is

$$O((wh/d^2)(g/d_z)n). \tag{4}$$

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 the of objective values for the two solutions:

$$s(\tilde{c})/s(c^*) = 1 - \epsilon \tag{5}$$

Since equation 2 defines a maximization problem, $s(c^*)$ is always greater than or equal to $s(\tilde{c})$ so the $0 \leq \epsilon < 1$. As $\epsilon \to 0$, $s(\tilde{c}) \to s(c^*)$.

We will establish theorems that give an upper bound ϵ_u such that $\epsilon \leq \epsilon_u$ for given d and d_z. This characterizes the tradeoff between solution quality and computation speed. We first prove lemmas based on 2 observations:

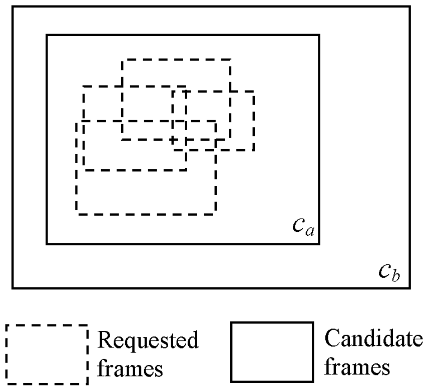

Fig. 2. *Example illustrating the lower bound on solution quality.*

- As illustrated in Figure 2, consider a set of user requested frames 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, $r_i \subset c_a \subset c_b$ 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 cases, some user frames r_i will be included in c_b but outside c_a, which will only increase the ratio.
- 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 $\frac{s(c_b)}{s(c_a)} \geq \frac{z_a}{z_b}$.*

Proof: Recall that r_i is user i's requested frame. Let

- $a_i = Area(r_i)$,
- $p_{ai} = Area(c_a \cap r_i)$,
- $p_{bi} = Area(c_a \cap r_i)$, then $p_{bi} \geq p_{ai}$,
- $S_a = \{i | r_i \cap c_a \neq \emptyset\}$ be the set of users whose requested frames intersect with frame c_a.
- $S_b = \{i | r_i \cap c_b \neq \emptyset\}$ be the set of users whose requested frames intersect with frame c_b. $S_a \subseteq S_b$
- $S^{'}$ be the set of users whose requested frames intersect with frame c_a and are bigger than the c_a, $S^{'} = \{[x, y, z] | [x, y, z] \in S$ and $z \geq z_a\}$.
- $S^{''}$ be the set of users whose requested frames intersect with frame c_a and are bigger than the c_b, $S^{''} = \{[x, y, z] | [x, y, z] \in S$ and $z \geq z_b\}$. $S^{''} \subseteq S^{'} \subseteq S_a$ because $z_b \geq z_a$.

we have,

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

and because $S_a \subseteq S_b$,

$$\begin{aligned} s(c_b) &= \sum_{i \in S_b} (p_{bi}/a_i) \min(z_i/z_b, 1) \\ &\geq \sum_{i \in S_a} (p_{bi}/a_i) \min(z_i/z_b, 1) \end{aligned}$$

Therefore, $s(c_b)/s(c_a)$

$$\begin{aligned} &\geq \frac{\sum_{i \in S_a} (p_{bi}/a_i) \min(z_i/z_b, 1)}{\sum_{i \in S_a} (p_{ai}/a_i) \min(z_i/z_a, 1)} \\ &\geq \frac{\sum_{i \in S_a} (p_{ai}/a_i) \min(z_i/z_b, 1)}{\sum_{i \in S_a} (p_{ai}/a_i) \min(z_i/z_a, 1)} \\ &= \frac{\sum_{i \in S''} (p_{ai}/a_i) + \sum_{i \in S_a - S''} (p_{ai}/a_i)(z_i/z_b)}{\sum_{i \in S'} (p_{ai}/a_i) + \sum_{i \in S_a - S'} (p_{ai}/a_i)(z_i/z_a)} \\ &\geq \frac{\sum_{i \in S_a - S''} (p_{ai}/a_i)(z_i/z_b)}{\sum_{i \in S' - S''} (p_{ai}/a_i) + \sum_{i \in S_a - S'} (p_{ai}/a_i)(z_i/z_a)} \end{aligned}$$

Define $S_l = \sum_{i \in S'-S''} (p_{ai}/a_i)(z_i/z_a)$. We know that

$$z_i/z_a \geq 1, \forall i \in S^{'} - S^{''}.$$

Then

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

So,

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

■

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,*

$$|x - x'| \leq d/2 \ \textit{ and } \ |y - y'| \leq d/2, \quad (6)$$

$$z' \leq \lceil \frac{3z + d}{3d_z} \rceil d_z. \quad (7)$$

If we choose $d = 3d_z$, then

$$z' \leq z + 2d_z. \quad (8)$$

Proof: The center (point B in figure 3) of the given frame c must have four neighboring lattice points. Without loss of generality, let's assume the nearest lattice point of the center is the top right lattice point, which is point O in figure 3. Other cases can be proven by symmetric.

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 the (x, y) on the Θ plane, which implies that equation 6 have to be true.

Recall that d is the spacing of the lattice. To ensure the point O is the nearest lattice point, equation 6 mean the point B must satisfy following constraints,

$$|\overline{OB}| \sin\alpha \leq d/2, \text{and } |\overline{OB}| \cos\alpha \leq d/2. \quad (9)$$

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

$$z' = \lceil \hat{z}/d_z \rceil d_z \quad (10)$$

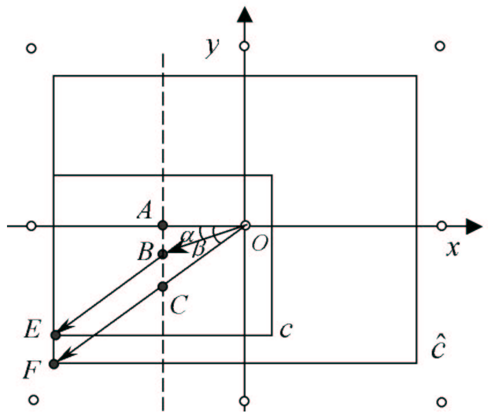

Fig. 3. *Relationship between frame c and the smallest frame $\hat{c}$ on the lattice that encloses it. In the figure, $\alpha = \angle AOB$, $\beta = \angle AOC$, and the frame c is centered at point B.*

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

$$x_F \leq x_E \text{ and } \ y_F \leq y_E. \quad (11)$$

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

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

Therefore, when $0 \leq \alpha \leq \beta$, $\overline{BE}$ is always on top of $\overline{OF}$, if $x_F = x_E$, then $y_F \leq y_E$. The boundary conditions for the $\hat{c}$ can be simplified:

- case 1: $x_F = x_E$ if $0 \leq \alpha \leq \beta$, and
- case 2: $y_F = y_E$ if $\beta \leq \alpha \leq \pi/2$.

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

$$|\overline{OC}| = |\overline{OF}| - |\overline{CF}| = |\overline{OF}| - |\overline{BE}|. \quad (12)$$

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 equation 9,

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

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

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

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

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

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

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

Plug it into equation 10,

$$z' \leq \lceil \frac{3z + d}{3d_z} \rceil d_z.$$

If we choose $d = 3d_z$, equation 7 can be simplified as,

$$z' \leq \lceil \frac{z}{d_z} \rceil d_z + d_z \leq z + 2d_z$$

■

Theorem 1: *Recall z_l is the smallest allowable z value and $d = 3d_z$. The approximation factor of the deterministic lattice based algorithm ϵ is bounded below by some constant c, $0 \leq \epsilon \leq c$, where*

$$c = \frac{2d_z}{z_l + 2d_z}$$

Proof: Recall that,

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

Note that $\tilde{c}$ is the solution to: $\max_{c \in L} s(c)$. Since $c' \in L \subset C$, we know that $s(c') \leq s(\tilde{c})$. Therefore

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

Therefore, applying Lemma 1,

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

Apply equation 8 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 $z^* \geq z_l$, so

$$1 - \epsilon \geq \frac{z_l}{z_l + 2d_z} \leftrightarrow \epsilon \leq \frac{2d_z}{z_l + 2d_z}$$

■

Theorem 1 says that the approximation bound $\frac{2d_z}{z_l+2d_z}$ is a monotonic increasing function of d_z. It characterizes the tradeoff between accuracy and computation speed for the grid-based approximation algorithm:

Grid Based Approximation Algorithm

I. For a given approximation bound ϵ, compute the appropriate lattice spacing: choose $d = 3d_z$, according to theorem 1, we set

$$\epsilon = \frac{2d_z}{z_l + 2d_z} \Rightarrow d_z = \frac{1}{2}(\frac{\epsilon}{1 - \epsilon})z_l$$

This is the maximum d_z that ensures the objective function value is bounded above $(1 - \epsilon)s(c^*)$.

II. Compute the objective function value for each lattice point, output the the lattice point with the largest objective function value as the approximated solution.

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

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

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

B. Distributed algorithm

In the ShareCam system, n is the number of users online, which is also the number of computers connected to our server. The larger the value of n, the more computation power in the system. This suggests that a distributed algorithm, where each client shares in the computation, can reduce overall computation time. In particular, we propose an algorithm where each client searches a coarse lattice with appropriate offsets. The more clients that participate, the more lattice points that are searched. The algorithm described in the previous section can be divided into client and server components by dividing the lattice L into n sub lattices L_i, $i = 1, ..., n$, where sub lattice L_i will be searched by user i.

Recall that $[z_l, z_u]$ is the zoom range, define

$$o_i = (i - 1)d_z + z_l$$

be initial zoom offset for user i. Then the sub lattice L_i is:

$$\begin{aligned} L_i \quad = \quad & \{(pd, qd, rnd_z + o_i) | pd \in [0, w], qd \in [0, h], \\ & rnd_z + o_i \in [z_l, z_u + 2d_z], p, q, r \in \mathcal{N}\}. \end{aligned}$$

Therefore, $\bigcup_{i=1}^{n} L_i = L$. Let s_i^* be the optimal solution given by i^{th} user, the server should do the following.

Server Algorithm

```
I. Compute the $d_z$, set $d = 3d_z$,
II. Send all $d_z$, $d$, and all requested frames to clients,
III. Wait until all clients send their solutions $\{s_1^*, ..., s_n^*\}$ back and pick the largest.
```

The i^{th} user should do following after receives d_z, d, and requested frames from server,

Client Algorithm

```
I. Compute zoom offset $o_i$.
II. Evaluate objective function with respect to sub lattice $L_i$.
III. Send the $s_i^*$, the optimal on lattice $L_i$ to server.
```

Since each sub lattice only contains $1/n$ points of L, the computation complexity is,

Theorem 3: *Each client runs in $O(1/\epsilon^3)$ time and the server runs in $O(n + 1/\epsilon^3)$ time.*

We distribute computation to client computers under the risk that some clients may drop out the system in the middle of the computation. One can also see from the algorithm that the speed of computation is limited by the slowest client. We may not have the luxury to wait for the slowest client to send his/her result back to server. Therefore, we are interested in the algorithm performance if one or more clients fail to submit their computation results before a time-out.

Theorem 4: *If one client fails to submit his/her result, the approximation bound ϵ will gracefully increase from $\frac{2d_z}{z_l+2d_z}$ to $\frac{3d_z}{z_l+3d_z}$.*

Proof: Without loss of generality, let's assume user i drops out. Let $c' = [x', y', z'] \in L - L_i$ be the smallest frame that ensures c^* is within c' and $\tilde{c}$ be the optimal found on lattice $L - L_i$. We consider two scenarios:

i). $c' \notin L_i$. Then

$$1 - \epsilon = \frac{s(\tilde{c})}{s(c^*)} \geq \frac{s(c')}{s(c^*)} \geq \frac{z_l}{z_l + 2d_z} > \frac{z_l}{z_l + 3d_z}$$

still holds. Therefore $\epsilon < \frac{3d_z}{z_l+3d_z}$.

ii).$c' \in L_i$. $\frac{s(\tilde{c})}{s(c^*)} \geq \frac{s(c')}{s(c^*)}$ is invariant. From the result of Lemma 1, we know $\frac{s(c')}{s(c^*)} \geq \frac{z^*}{z'}$. Since $c' \in L_i$, equation 10 will not hold. Instead, for any frame $c' \in L_i$,

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

Then

$$z' \leq z + 3d_z.$$

Similar to the proof of Theorem 1, we have

$$1 - \epsilon = \frac{s(\tilde{c})}{s(c^*)} \geq \frac{z_l}{z_l + 3d_z}.$$

Hence, $\epsilon \leq \frac{3d_z}{z_l+3d_z}$ ■

It is good to see that the approximation bound does not change dramatically after one client drops out of the system. In fact, the proof of theorem 4 shows that the approximation bound change is very small. We can also extend the proof to cases where more than one user drop out.

C. Numerical Experiment

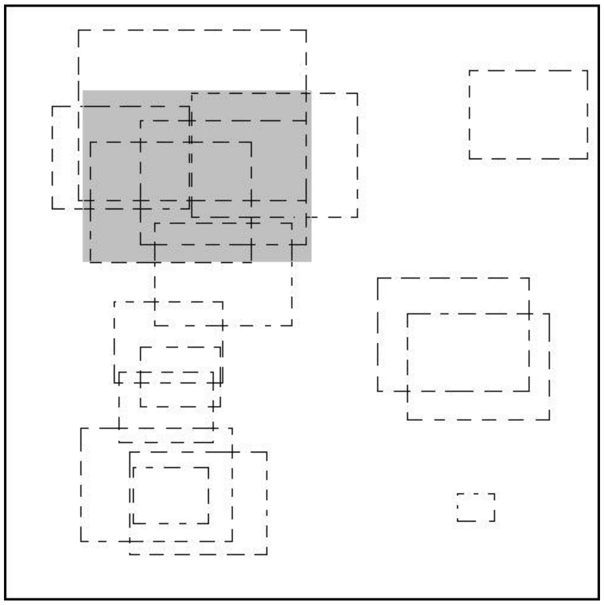

Fig. 4. *Sample output of algorithm with 16 requested frames and $d_z = 2$. The shaded frame is optimal.*

Figure 4 shows the algorithm's result for a random example with 16 requested frames.

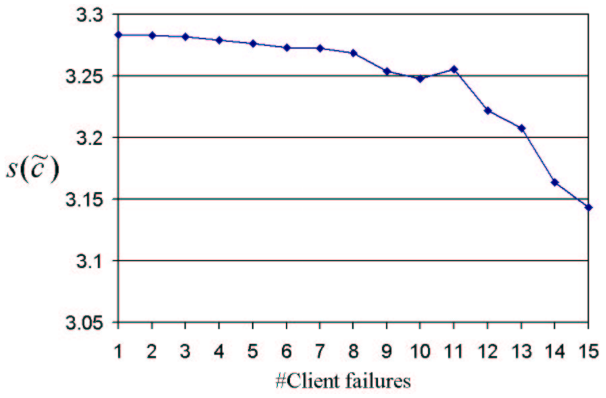

Fig. 5. *Performance of the Distributed Algorithm: solution quality as network clients fail. Each data point is an average of 10 runs with random dropouts. Initially there are 16 clients and 16 requested frames as shown in the previous figure. Here we plot solution quality as more and more clients fail to complete their lattice searches. Note that solution quality decreases, but not dramatically: even when only one client remains (after 15 have failed) searching only one coarse lattice is enough to find a reasonable solution.*

Figure 5 shows how solution quality decreases as network clients fail, based on the example in figure 4. This supports the analysis in the previous section: the approximation algorithm degrades gracefully with client failures.

V. Conclusions and Future Work

We present new algorithms for the ShareCam Problem: controlling a single robotic pan, tilt, zoom camera based on simultaneous frame requests from n online users. We formalize the problem with continuous pan, tilt, and zoom. With approximation bound ϵ, the algorithm runs in $O(n/\epsilon^3)$ time.

We also show that the algorithm can be distributed to run in $O(1/\epsilon^3)$ time at each client and in $O(n+1/\epsilon^3)$ time at the server. Unlike computing with multiple processors in a single supercomputer, distributed computing over the Internet requires input from a variety of heterogenous processors, each with different and varying communication delays. We show that the grid-based algorithm handles client failures gracefully.

In future work, we will explore how these algorithms can be further improved with Brand and Bound techniques and dynamic data structures. The Sharecam system was moved to an outdoor location on the UC Berkeley campus in June 2003, and is available online at http://tele-actor.net/sharecam.

VI. Acknowledgments

We are grateful to Frank van der Stappen, Sariel Har-Peled, Vladlen Koltun, Dana Plautz, Anatoly Pashkevich, Cory Cox, Yong Rui, J. Foote, Q. Lui, and Eric Paulos for insightful discussions and feedback. Thanks to M. Faldu and W. Guan for their contributions to the ShareCam system, and I.Y. Song, K. "Gopal" Gopalakrishnan , Ron Alterovitz, A. Levandowski, M. Mckelvin, A. Ho, and J. Vidales for helpful suggestions.

VII. REFERENCES

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

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

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

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

[5] K. Goldberg, B. Chen, R. Solomon, S. Bui, B. Farzin, J. Heitler, D. Poon, and G. Smith. Collaborative teleoperation via the internet. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 2019–2024, April 2000.

[6] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1179–84, May 2002.

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

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

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

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

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

[12] M.S. Kankanhalli and W. R. Franklin. Area and perimeter computation of the union of a set of iso-rectangles in parallel. *Journal of Parallel and Distributed Computing*, 27:107–117, 1995.

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

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

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

[16] S.G. Nash and A. Sofer, editors. *Linear and Nonlinear Programming*. The McGraw-Hill Companies, Inc, 1996.

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

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