Collaborative Frame Selection: Exact and Distributed Algorithms for a Networked Robotic Camera with Discrete Zoom Levels *

Dezhen Song†, A. Frank van der Stappen‡ and Ken Goldberg§

†: IEOR Department, University of California, Berkeley, CA 94720, USA

‡: Institute of Information and Computing Sciences, Utrecht University PO Box 80089, 3508 TB Utrecht, The Netherlands

§: IEOR and EECS Departments, University of California, Berkeley, CA 94720, USA

Submission date: May 21, 2003

Abstract

Collaborative Frame Selection arises when one robotic pan, tilt, zoom camera is shared by many users. The problem is to compute optimal camera parameters based on simultaneous frame requests from all users. We formalize the problem using a new metric, Intersection Over Maximum (IOM), to model the degree of satisfaction for each user, and seek to maximize total satisfaction for n users. We assume the zoom parameter is chosen from a discrete set of m levels and consider cases with discrete and continuous pan and tilt paramters. For a discrete set of $w \times h$ pan and tilt values, we give an exact algorithm that runs in $O((n + mwh)\log^2 n)$. For continuous pan and tilt, we give an exact algorithm that runs in $O(n^2 m)$ time. We also give a distributed version that runs in $O(nm)$ time at each client and in $O(n \log n + mn)$ time at the server. An implementation of the second algorithm can be found online at: http://www.tele-actor.net/sharecam/.

KEY WORDS– Internet robot, teleoperation, webcam, collaborative control, videoconferencing

1 Introduction

Consider a robotic camera at a compelling location such as the Sydney boat harbor, United Nations, Academy Awards, or inside the International Space Station. The camera frame is determined by pan, tilt, and zoom parameters that can be changed to observe details of the scene. However there are many viewers contending for control of the camera. One commercial solution uses a user queue: viewers wait patiently for their turn to operate the camera [Canon, 2003]. Obviously, the user queue does not scale well. In this paper, we eliminate the queue and allow many users to share control of the camera simultaneously.

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

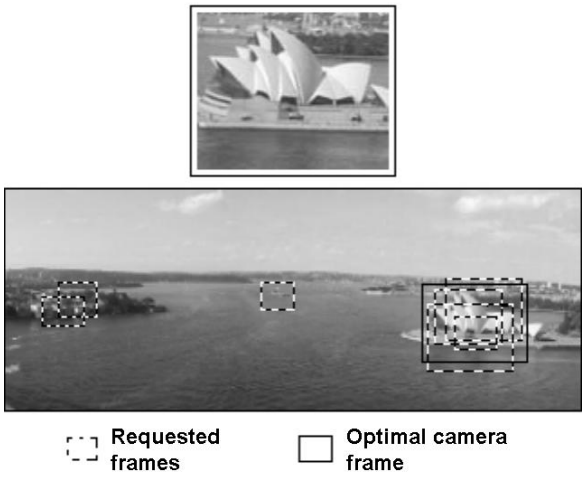

Figure 1: *User interface. Each Internet-based user sees two image windows. The lower window is a fixed image of the camera's reachable range of view. A user requests a camera frame by positioning a dashed rectangle in the lower window. Based on frame requests from all users, we compute an optimal camera frame (shown with solid rectangle), move the camera accordingly, and display the resulting live image in the upper window.*

We are developing network-based applications for education, journalism and entertainment where many users share control of a single physical resource. "Sharecam" is an example of Collaborative Telerobotics, where the camera is a telerobot with 3 degrees of freedom. In the taxonomy proposed by Chong et al. [Chong et al., 2000], this is a Multiple Operator Single Robot (MOSR) system. Our research is motivated by applications 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. There can be benefits to such collaboration: teamwork is a key element in education at all levels [Crouch and Mazur, 2001, Rogoff et al., 1996] and an ensemble of users may be more reliable than a single (possibly malicious) user [Goldberg and Chen, 2001].

As illustrated in Figure 1, the user interface includes two image windows, one with a fixed image for user input and

the other that displays a live image based on all requests. The input is a set of requested camera frames specified as desired fixed aspect-ratio iso-oriented rectangles from n users. The output is a single camera frame.

We propose a new metric for user "satisfaction" based on how a user's requested frame compares with a candidate camera frame. The metric is proportional to the common area of the candidate frame and the requested frame and inversely proportional to the ratio of the sizes of the candidate and the request. The latter discourages excessively large frames, such as the enclosing rectangle of all requests.

Finding the camera frame that maximizes total satisfaction is a non-linear optimization problem. We assume a discrete set of allowable zoom levels and consider cases with discrete and continuous pan and tilt. Let n be the number of users, m be the number of zoom levels, and w and h be the number of pan and tilt values respectively. For discrete pan and tilt, we give an algorithm that runs in $O((n + mwh)\log^2 n)$ based on recent spatial data structures developed for box aggregation. For continuous pan and tilt, we identify a finite set of critical points we call "virtual corners"; we prove that a global maximum must exist at a virtual corner. The latter algorithm runs in time $O(n^2m)$. Finally, we present a distributed version of this that runs in time $O(n \log n + mn)$ on the server and $O(mn)$ on each client.

2 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 [Goldberg et al., 2003, Goldberg and Siegwart, 2002, Hu et al., 2001, Safaric et al., 2001, Jia and Takase, 2001] for recent examples.

Chong et al. [Chong et al., 2000] proposed the following taxonomy for teleoperation systems: Single Operator Single Robot (SOSR), Single Operator Multiple Robot (SOMR), Multiple Operator Single Robot (MOSR), and Multiple Operator Multiple Robot (MOMR). Most online robots are SOSR, where control is limited to one operator at a time. One precedent of an online MOSR system is described by McDonald, Cannon, and colleagues [Cannon, 1992, McDonald et al., 1997]. 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.

In [Goldberg et al., 2000, Goldberg and Chen, 2001], Goldberg and Chen described an Internet-based MOSR system that averaged multiple human inputs to simultaneously control a single industrial robot arm. In [Goldberg et al., 2002, Goldberg et al., 2003] Goldberg, Song, et al. propose the "Spatial Dynamic Voting" (SDV) interface. The SDV collects, displays, and analyzes a sequence of spatial votes from multiple online operators at their Internet browsers. The votes drive the motion of a single mobile robot or human "Tele-Actor".

We formulate the collaborative control problem as a nonlinear optimization problem with a non-differentiable objective function. The structure of the problem is closely related to the planar $p-$center problem, which has been proved to be NP-complete by Megiddo and Supowit [Megiddo and Supowit, 1984]. Using a geometric approach, Eppstein [Eppstein, 1997] found an algorithm for the the planar 2-Center problem in $O(n \log^2 n)$. Halperin et al. [Halperin et al., 2002] 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 [Nash and Sofer, 1996].

The camera frame selection problem can be viewed as a special case of the general "box aggregation" problem over spatial objects in database research [Zhang et al., 2002]. The spatial objects could be point objects, intervals, or rectangular objects. Aggregation over point objects is a special case of orthogonal range searching in computational geometry. Agarwal and Erickson [Agarwal and Erickson, 1999] provide a review of geometric range searching and its related topics. Grossi and Italiano [Grossi and Italiano, 1999, Grossi and Italiano, 2000] proposed the cross-tree data structure, a generalized version of balanced tree, to speed up range searching in high-dimensional space.

Work has been reported on aggregation over point objects [Chan and Ioannidis, 1999, Geffner et al., 2000, Ho et al., 1997] and 1D intervals [Yang and Widom, 2001, Zhang et al., 2001]. Zhang, Tsotras, and Gunopulos [Zhang, 2002, Zhang et al., 2002] extend the research to rectangular spatial objects, usually referred to as "aggregation over spatial objects with extents". Such aggregation is not limited to constant weights but also to the more general case where the weight is a polynomial function of the intersection area between the query window and the rectangles. This general box aggregation problem is also known as a

functional box sum query.

As in box aggregation problems, the camera frame selection problem can be reduced to dominance sum problems (defined in section 4.1.2). As shown by Guttman, dominance sum problems anddominance sum queries can be solved using rectangle trees (R-Trees) [Guttman, 1984] or its recent variations [Lazaridis and Mehrotra, 2001, Paradias et al., 2001] by reducing the problem to range search. The aggregate is then computed by identifying the objects that intersect the query range and accumulating their values incrementally. This method is inefficient because it is based on how many objects are in the query range, which can be large. A more efficient data structure is the ECDF-tree [Bentley, 1980], which solves the dominance sum problem using a multi-dimensional divide and conquer technique. The original ECDF-tree is a static data structure but Zhang et al. recently extended it: the dynamic ECDF-B-tree allows update and bulk-loading operations [Zhang, 2002, Zhang et al., 2002]

It is also possible to view camera frame selection as a clustering problem [Jain et al., 1999, Johnson, 1998]. The user satisfaction metric function we define later in the paper is based on the resemblance and containment relationship between users' requested camera frames and real camera frame. The Symmetric Difference (SD) and "Intersection Over Union"(IOU) are well-known similarity metrics [Broder, 1998, Broder et al., 2000, Veltkamp and Hagedoorn, 2000]. The comparison of these metrics and our metric will be discussed later in the paper.

There is also a connection with distributed manipulation. One branch of distributed manipulation uses potential fields defined as "potential-per-unit-area" acting on an object [Bohringer and Choset, 2000, Moon and Luntz, 2002]. It is possible to interpret the satisfaction function as a special "lifted" potential field with some modifications.

Distributed computation has been used for sensor processing [Mumolo et al., 2000], multi-actuator control, and multi-robot systems. Sagawa et al. [Sagawa et al., 2001] developed a parallel algorithm to merge a set of range images into a volumetric implicit surface image representation, which is converted to a surface mesh. Safaric et al. [Ku et al., 2001] designed a distributed control system for an active surface device. The active surface device is a massive parallel micro-actuator array that can generate a pressure field on a planar surface. Applications of distributed algorithms include motion planing [**?**, Parker, 2002], localization [Hayes et al., 2001, Mumolo et al., 2000], and task allocation [Agassounon et al., 2001, Chen et al., 1999].

In independent work, Kimber and Liu et al. describe a multi-user robot camera in [Kimber et al., 2002, Liu et al., 2002]. As we do in Sharecam, they formulate the frame selection for multiple simultaneous requests as an optimization problem based on position and area of overlap. To solve it, they propose an approximation based on comparing the bounding box of all combinations of user frames. This algorithm requires exponential time and does not provide formal bounds on the approximation error.

This paper is an expanded and updated description of results initially presented at the Fifth International Workshop on Algorithmic Foundations of Robotics [Song et al., 2002].

3 Problem definition

In this section we formalize the Collaborative Frame Selection problem: finding the camera frame that maximizes total user satisfaction.

3.1 Input and assumptions

Let c be a vector of parameters. For a robotic camera, $c = [x, y, z]^T$, defines a camera frame: x, y specify the pan and tilt center point of the frame, and z specifies the size of the frame. We assume the camera has the standard aspect ratio of 4:3 and define size z so that frame length is $4z$ and width is $3z$. (Note that z is the inverse of the standard camera zoom parameter).

At each time increment, user i requests a desired frame, $r_i = [x_i, y_i, z_i]$. Given requests from n users, we must compute a single frame c^* that will best satisfy the set of requests.

Let Θ be the set of all admissible $[x, y]$ pan, tilt pairs. Let Z be a small set of attainable zoom levels. The solution space is:

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

3.2 Metric for User Satisfaction

Recall that r_i is the frame requested by user i, and let $c = [x, y, z]^T$ be a candidate camera frame. We define a scalar $s_i \in [0, 1]$ as the level of "satisfaction" that user i receives. User i gets no satisfaction if the candidate frame does not intersect r_i, so $s_i = 0$ when $c \cap r_i = \emptyset$. User i is perfectly satisfied when the candidate frame is identical to r_i: $s_i = 1$ when $c = r_i$. To characterize the satisfaction of user i, we propose the Intersection Over Maximum (IOM) function:

$$s_i(r_i, c) = \frac{Area(r_i \cap c)}{\max(Area(c), Area(r_i))}$$

Let $z = Size(c)$ and $z_i = Size(r_i)$, then for the IOM function

$$s_i(r_i, c) = \frac{Area(r_i \cap c)}{Area(r_i)} \min\Big(\big(\frac{Size(r_i)}{Size(c)}\big)^2, 1 \Big)$$

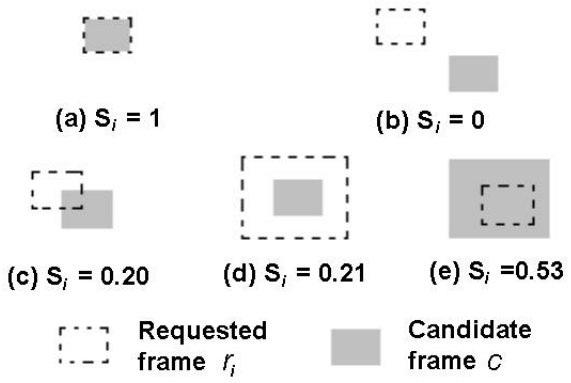

Figure 2: Examples of the satisfaction metric for user i and candidate frame c.

we can define a Generalized Intersection Over Maximum function (GIOM):

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

where b is a small positive number ($b = 2$ for the IOM function).

Note that the second term in 1 characterizes the tradeoff between union and intersection. Small values of b produce camera frames that cover all requested frames (which does not reflect desired user zoom levels) and large values produce frames that only cover the intersection of all requested frames. As described in Section 5, we experimented with different b values and found that $b = 1$ provides a good balance as illustrated in Figures 2 and 8.

The total satisfaction for n users is

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

We want to find $c^* = \arg\max_c s(c)$, the frame that maximizes total satisfaction. To describe the objective function with respect to x, y, and z,

$$s(x, y, z) = s(c).$$

3.3 Properties of the Satisfaction Metric

The GIOM satisfaction function s is nonsmooth and piecewise linear in both x and y, in contrast to the Intersection-Over-Union (IOU) metric as shown below.

3.3.1 Nonsmoothness

Recall that $Area(r_i) = 12z_i^2$, $z = Size(c)$, and $z_i = Size(r_i)$, since we solve this problem for each attainable zoom level z, we treat z as a constant. Therefore, $Area(c \cap r_i) = p_i(x, y)$ is a function of (x, y). The objective function defined by equation 1 becomes a function of the center point of the candidate frame,

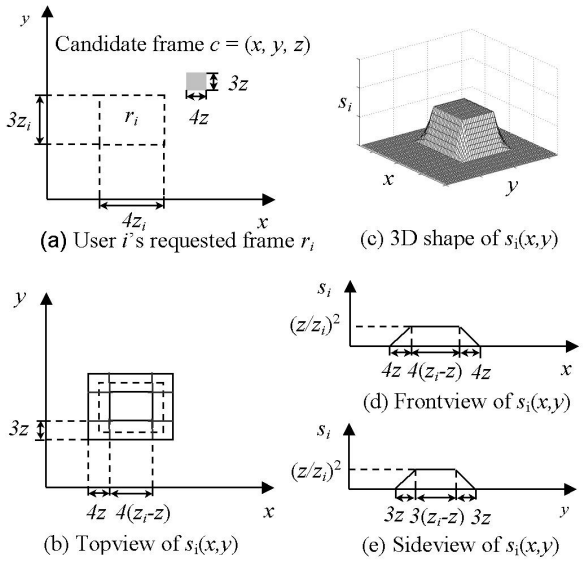

Figure 3: *The IOM user satisfaction function, $s_i(x, y)$, for a given candidate frame. Since $z \leq z_i$ is given, we can move the candidate frame (gray rectangle) around the user i's requested frame to observe how $s_i(x, y)$ changes. The function is plateau-like with a maximum height of $Area(c \cap r_i)/Area(r_i) = 12z^2/12z_i^2 = (z/z_i)^2$. The function consists of 5 planar and 4 quadratic surfaces at the corners.*

$$s(x, y) = \sum_{i=1}^{n} \omega_i p_i(x, y) \qquad (3)$$

where

$$\omega_i = \frac{1}{12z_i^2} \min((z_i/z)^b, 1) \qquad (4)$$

is a constant for each user. We know that $p_i(x, y)$ is the area of the intersection of the requested frame of user i and the candidate frame (x, y, z). Therefore, the maximum value of $p_i(x, y)$ is $\min(Area(c), Area(r_i))$. This property determines that the shape of user i's satisfaction function is plateau-like. Figure 3 shows the shape of $s_i(x, y)$ given $z \leq z_i$, i.e. the candidate frame is smaller than the requested frame of user i. Note that s_i is non-differentiable with respect to x and y so we cannot use derivative-based approaches to solve this problem.

3.3.2 Piecewise linearity in x and y.

Since all requested frames 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 equation 3 is

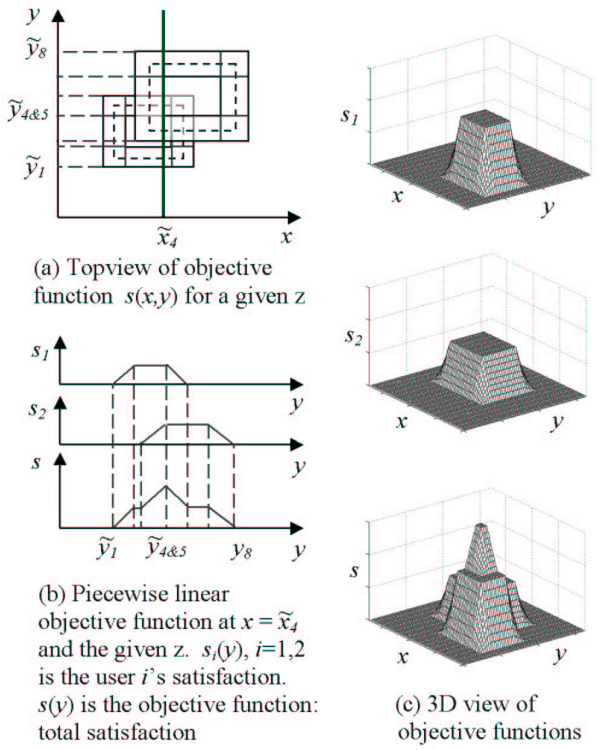

Figure 4: *The IOM user satisfaction function $s_i(y)$ for two users. Ordered sets $\{\tilde{y}_k\}$ and $\{\tilde{x}_k\}$, $k = 1, ..., 8$ are corresponding to horizontal and vertical edges of plateaus. Note that $\tilde{y}_4$ and $\tilde{y}_5$ overlap in this case.*

either 0 or the area of the rectangle formed by 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 satisfaction metric $s(x, y)$ is a linear combination of $p_i(x, y)$, $i = 1, ..., n$, it has the same property. Figure 4 shows an example for a case with two requested frames.

3.3.3 Comparison to other metrics.

In pattern recognition and computational geometry standard similarity metrics are Symmetric Difference (SD) and Intersection Over Union (IOU)[Broder, 1998, Broder et al., 2000, Veltkamp and Hagedoorn, 2000]. For a requested frame r_i and a candidate frame c, the SD metric is

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

The intersection-over-union metric is

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

Compared with IOU, our satisfaction metric has similar properties: (i) both attain their minimum value of 0 if and only if $c \cap r_i = \emptyset$, (ii) both attain their maximum value of 1 if and only if $c = r_i$, (iii) both are proportional to the area of $c \cap r_i$, and (iv) both depend—albeit differently—on the sizes of c and r_i. However, neither the IOU nor the SD metrics are piecewise linear in x or y.

4 Algorithms

In this section we present algorithms for two versions of the Collaborative Frame Selection problem. We start with a version in which the pan (x) and tilt (y) are restricted to a discrete set of equally-spaced values. Subsection 4.1 describes a algorithm for this discrete version of the problem. In Subsection 4.2 we allow the pan and tilt to vary continuously. This more general continuous version allows for an efficient exact algorithm. The algorithm exploits a geometric characteristic of the optimal solution (captured in the notion of a "virtual corner"). The exact algorithm can also be distributed across the client machines and the server. The distributed algorithm is given in Subsection 4.3.

4.1 Algorithms for discrete pan and tilt

4.1.1 Brute force approach.

Let w be the width (in pixels) of the camera's total pan range, and h be the height (in pixels) of the camera's total tilt range. A brute force search for finding $c^* = \arg\max_c s(c)$ evaluates all whm candidate frames. A straightforward approach takes $O(n)$ computing time to determine the satisfaction for a single candidate frame c. The total amount of computation of the algorithm is $O(whmn)$. Although this is linear in n, the constants are large (typically $w = 600, h = 200$).

4.1.2 Efficient function evaluation using Functional Box Sum (FBS) query.

Computing the satisfaction function value for a given candidate frame can be viewed as a box aggregation problem over a group of requested frames. As shown in equation 3, the query result should be a weighted sum of intersected areas between the query window (i.e. the candidate frame) and requested frames/rectangles. This special box aggregation problem can be solved by the functional box sum query introduced by Zhang, Tsotras, and Gunopulos in [Zhang et al., 2002]. Given a set of pairs consisting of a rectangular box and an associated value function,

> a functional box sum (FBS) query with a box q asks for the total value contributed by all boxes r intersected by q, where the value contributed by a box r is the integral of the value function associated with r over $q \cap r$.

Our satisfaction function (for a fixed zoom level) is a simple example of a value function. Zhang et al. have shown that an FBS query in dimension d can be reduced to 2^d dominance-sum queries. We say that a point (x_1, y_1) is dominated by a point (x_2, y_2) if and only if $x_1 \leq x_2$ and $y_1 \leq y_2$. Given a set of pairs consisting of a point and an associated value,

> a dominance sum query with a point q asks for the sum of all values associated with the points dominated by q.

The reduction is done by first reducing an FBS query to 2^d Origin Involved Functional Box Sum (OIFBS) queries and then showing that each OIFBS query is equivalent to a dominance sum query. The OIFBS query is a special FBS query that has bottom left corner of the query window located at the origin, which is to the left and below all rectangles. For $d = 2$ in our case, figure 5 from [Zhang et al., 2002] shows how to convert an FBS query into 4 OIFBS queries.

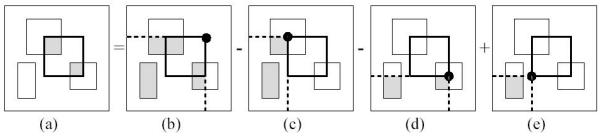

Figure 5: Convert an FBS query to 4 OIFBS queries in [Zhang et al., 2002].

The transformation from an OIFBS query to a dominance sum query converts rectangular spatial objects to point objects by assigning each vertex a point value function. The point value function is a linear combination of integrals of value functions in the original FBS query. An important contribution of [Zhang et al., 2002] is that it shows how to generate those point value functions based on the original value functions. The dominance sum query is then used to compute the sum of point value functions for dominated points. Then an evaluation of aggregated point value functions will give the result of the FBS query. In order to ensure the validity of the transformation, the point value functions have to be closed under $+$ or $-$ operators. For example, a polynomial function can satisfy the condition: adding two second-order polynomials will generate a new polynomial of the same order. Since point value functions are linear combinations of integrals of the value functions in the original FBS query, the requirement is equivalent to condition that the value function in the original FBS query have to be closed under $+$ or $-$ operator.

The dominance sum query can be efficiently performed by preprocessing the requested frames into ECDF tree or its variations. The 2D ECDF-tree is a two-level tree. The primary structure is a simple binary tree on the first coordinate of the points. Each internal node v stores a key and the sum of the values associated with all the points stored in the subtree rooted at v. These points are (evenly) distributed across the left and right child of v. All points with a first coordinate smaller than or equal to the key are in the subtree rooted at the left child of v, while the remaining nodes are in the subtree rooted at the right child of v. In addition to the key and sum, the node v stores a secondary structure. This secondary structure is a similar binary tree (1D ECDF tree) on the second coordinates of all the points stored in the subtree rooted at v.

A query in the 2D ECDF-tree proceeds as follows. If the query point is located in the left child, a recursive query of the left sub tree is executed. If the query point is located in the right child, the query splits into two sub-queries. The first sub query is with respect to the 1D ECDF tree in the left child, which takes care of points falling into the left child and are dominated by the query point. The second sub-query is a recursive query on the right sub 2D ECDF tree. The sum of the two queries gives the result. As shown in [Bentley, 1980], a query of the 2D ECDF tree requires $O(\log^2 n)$ for n points. The construction of the tree takes $O(n \log^2 n)$.

The basic ECDF-tree is a static data structure. Zhang et al. expanded it to the ECDF-B-tree to allow dynamic updates. The ECDF-B-tree and its variations are proposed as disk based data structures, which can also be modified to memory data structures. Zhang et al. have shown that the query time of ECDF-B^q-tree, which is a variation of ECDF-B-tree with query time optimized, is $O(\log^2 n)$ for 2D. Therefore, each FBS query should also be $O(\log^2 n)$. Applying the FBS query, we have following algorithm,

Basic FBS Query Based Algorithm

i. For each zoom level z	(m in total)
a. Preprocess requested frames into	
an ECDF-B^q-tree	$O(n \log^2 n)$
b. For each pan and tilt pair (x, y)	(wh in total)
Do an FBS query with the frame	
(x, y, z) to find its satisfaction.	$O(\log^2 n)$
ii. Report the pan, tilt, and zoom	
with the largest FBS query value.	$O(1)$

This will yield an algorithm with running time of $O(m(n+wh) \log^2 n)$. However, we notice that it is not necessary to rebuild the tree for each zoom level. As we mentioned earlier, the dominance sum query is used to compute

the sum of point value functions, which have to be closed under + or - operation. For equation 4, we know that our value function $\omega_i(z)$ will not satisfy the condition for every zoom due to the difficulty caused by the $\min$ function. However, at all zoom levels except $z = z_i$, where $\omega_i(z)$ changes from $(z_i/z)^b$ to 1, the function $\omega_i(z)$ is a polynomial function. This indicates that we only need to do a tree updates at zoom levels z_i for $i = 1, ..., n$ instead of rebuilding the whole tree. The update involves a deletion of the old value function and an insertion of the new value function at the corresponding nodes and leaves, which should take $O(\log^2 n)$ for 2D ECDF-trees. Therefore, we have following algorithm,

FBS Query Based Algorithm with Tree Updates

i. Construct an ECDF-B^q-tree	
for the smallest zoom	$O(n \log^2 n)$
ii. For each zoom level z	
in an ascended order	(m in total)
a. Update the ECDF-B^q-tree if	
the zoom level just cross	
one of z_i for $i = 1, ..., n$,	$(**)$
b. For each pan and tilt pair (x, y)	(wh in total)
Do an FBS query with the frame	
(x, y, z) to find its satisfaction.	$O(\log^2 n)$
iii. Report the pan, tilt, and zoom	
with the largest FBS query value.	$O(1)$

Since there are n request frames, step (**) should cost $O(n \log^2 n)$ in total. Other steps will cost $O((n + mwh) \log^2 n)$ in total. Hence, the following theorem is true.

Theorem 1. *Using an ECDF-B^q-tree, we can solve the ShareCam problem with discrete pan and tilt in $O((n + mwh) \log^2 n)$.*

4.2 An algorithm for continuous pan and tilt

We now focus on the more general problem where camera pan and tilt parameters vary continuously. We show that user inputs define a finite set of critical points in the objective space.

4.2.1 Virtual corners

Recall that the objective function for one user $s_i(x, y)$ is plateau-like as shown in Figure 3. The function consists of 9 facets: 1 top plane, 4 side planes, and 4 quadratic surface at 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 (see Figure 6(a)).

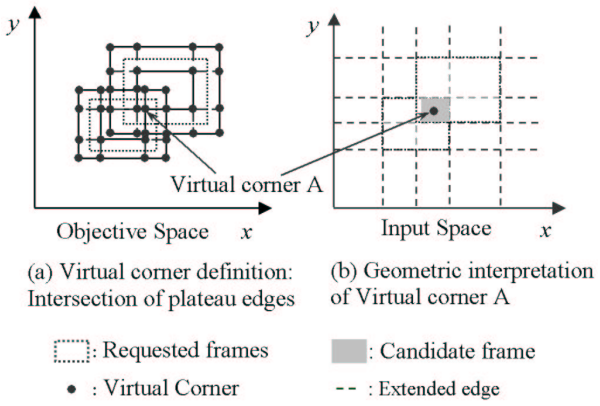

Figure 6: *Illustration of "virtual corners" with geometric interpretation for two requested frames. A virtual corner corresponds to a candidate frame that has one corner at the intersection of one extended vertical edge of a requested frame and one horizontal extended edge of a requested frame.*

For n users, there are n plateaus. We define a *virtual corner* as an intersection between any two boundaries, which includes both intersections of facet boundaries induced by a single plateau or by two distinct plateaus. Since all plateaus are iso-oriented, one of the extensions is horizonal and the other is vertical. For n requested frames, there are $O(mn^2)$ virtual corners. Figure 6(b) shows the geometric interpretation of virtual corner in the input space. If we map the virtual corner in the objective space, which describes the shape of the satisfaction function, back to input space, we see that the virtual corner corresponds to a candidate frame that has one corner overlapping with the intersection of two extensions of edges of requested frames.

Lemma 1. *At least one optimal frame is centered at a virtual corner.*

Proof. Let $c^* = [x^*, y^*, z^*]$ be an optimal solution. 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 line $y = \tilde{y}$ in (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 virtual corner $(\tilde{x}, \tilde{y})$. □

4.2.2 Brute force approach.

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

4.2.3 FBS query approach

We can do slightly better using the Functional Box Sum query. Querying with the $O(n^2)$ virtual corners at each of the m zoom levels will yield an algorithm with complexity of $O(mn^2 \log^2 n)$. However, we can do better if (for a fixed zoom level) we handle all virtual corners with the same x-coordinate consecutively in order of increasing y-coordinate, and take advantage of the fact that the objective function only changes slightly between two consecutive virtual corners.

4.2.4 Efficient traversal of virtual corners.

For n requested frames, we have $4n$ horizontal plateau facet boundaries $\{\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_{4n}\}$ and $4n$ vertical plateau facet boundaries $\{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_{4n}\}$ for plateaus. As shown in Virtual Corner Traversal Algorithm, we can reduce the computation complexity from $O(n^3 m)$ to $O(n^2 m)$. Recall that $r_i = [x_i, y_i, z_i]$, $i = 1, ..., n$ are the requested frames.

In step iii of the Virtual Corner Traversal Algorithm, we traverse the vertical facet boundaries of the plateaus one by one. For each vertical edge, we find the maximum by forcing the candidate frame to center at it. Using Theorem 1, we know that this procedure will find an optimal solution. It remains to show 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 $\{\tilde{y}_1, \tilde{y}_2 ..., \tilde{y}_{4n}\}$. The objective function is a "summation" of n plateaus, which is shown in Figure 4. For fixed x and z, this piecewise linear function only changes slope at $\{\tilde{y}_i\}$, $i = 1, ..., 4n$. 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 $O(n^2)$ and the following theorem is true.

Theorem 2. *We can solve the Sharecam problem with continuous pan and tilt in time $O(n^2 m)$ for n users and m zoom levels.*

Virtual Corner Traversal Algorithm

$s^* = 0$, $O(1)$

Sort $\{y_i + 1.5z_i\}$, $i = 1, ...n$ $O(n \log n)$

Sort $\{y_i - 1.5z_i\}$, $i = 1, ...n$ $O(n \log n)$

For each zoom level z (m in total)

i. Compute vertical extended plateau edges

$\{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_{4n}\}$ $O(n)$

For each user i, $i = 1, ..., n$,

$\tilde{x}_{4i-3} = x_i - 2(z_i + z)$,

$\tilde{x}_{4i-2} = x_i - 2(z_i - z)$,

$\tilde{x}_{4i-1} = x_i + 2(z_i - z)$,

$\tilde{x}_{4i} = x_i + 2(z_i + z)$,

ii. Compute the sorted sequence

$\{\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_{4n}\}$, $O(n)$

For each i, $i = 1, ..., n$

$\tilde{y}_{4i-3} = (y_i - 1.5z_i) + 1.5z$,

$\tilde{y}_{4i-2} = (y_i - 1.5z_i) - 1.5z$,

$\tilde{y}_{4i-1} = (y_i + 1.5z_i) - 1.5z$,

$\tilde{y}_{4i} = (y_i + 1.5z_i) + 1.5z$,

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 ordered sequence $\{\tilde{y}_1, \tilde{y}_2 ..., \tilde{y}_{4n}\}$, where $\tilde{y}_1$ is the smallest.

iii. For $x = \tilde{x}_i$, $i = 1, ..., 4n$,

$s = \max_y \ s(\tilde{x}_i, y, z)$,

if $s > s^*$ then $s^* = s, x^* = \tilde{x}_i$,

$y^* = y, z^* = z$.

Output s^* as optimal objective function value and (x^*, y^*, z^*) as optimal frame.

4.3 A distributed algorithm

In the system, n is the number of users online, which is also the number of computers connecting to our server. The larger the value of n, the more computation power we have in our system. This suggests that a distributed computing strategy can further improve computational speed. The algorithm described in the previous section can be divided into client and server components.

The server should do the following.

Server Algorithm

i. Send all requested frames r_i, $i = 1, ..., n$ to all clients,

ii. Sort sequence $\{y_i + 1.5z_i\}$ and sequence $\{y_i - 1.5z_i\}$, $i = 1, ...n$ and send them to all clients,

$O(n \log n)$

iii. Wait until all clients send their solutions $\{s_1^*, ..., s_n^*\}$ back.

iv. Pick the largest. $O(n)$

Let us assume that $r_i = (x_i, y_i, z_i)$ is client i's requested frame. After client i receives the data from the server, it executes the following algorithm.

Client Algorithm

$s_i^* = 0$

For each zoom level z

i. Compute the sorted sequence $\{\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_{4n}\}$,

(Same as the centralized version.) $O(n)$

ii.a. $s_a = \max_y \ s(x_i - 2(z_i + z), y, z)$, $O(n)$

if $s_a > s_i^*$ then $s_i^* = s_a$,

ii.b. $s_b = \max_y \ s(x_i - 2(z_i - z), y, z)$, $O(n)$

if $s_b > s_i^*$ then $s_i^* = s_b$,

ii.c. $s_c = \max_y \ s(x_i + 2(z_i - z), y, z)$, $O(n)$

if $s_c > s_i^*$ then $s_i^* = s_c$,

ii.d. $s_d = \max_y \ s(x_i + 2(z_i + z), y, z)$, $O(n)$

if $s_d > s_i^*$ then $s_i^* = s_d$,

Send the s_i^* and its corresponding candidate frame to server.

As we can see from the algorithm, the server runs at $O(n \log n + mn)$ and each client runs at $O(nm)$. The following theorem holds.

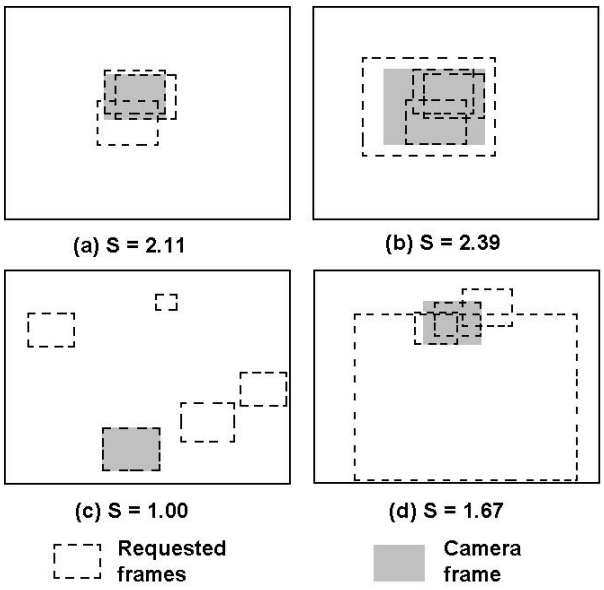

Figure 7: *Examples of computed optimal frames (shown in grey).*

Theorem 3. *We can distribute the Sharecam algorithm among the server and clients resulting in a running time of* $O(n \log n + mn)$.

One can also see from the algorithm that the speed of computation is limited by the slowest client. One idea is to set a timeout for clients and have the server run the computation for clients that do not respond in time.

5 Results

We have implemented the algorithms on a PC with 950Mhz AMD Athlon CPU and 1GB memory. The machine runs under Redhat Linux 7.1. The algorithm is programmed in both Matlab and Java.

Figure 7 shows the results for four different scenarios. As we can see from Figure 7(a) and (b), the optimal frame does not necessarily have one corner overlapping with a corner of a requested frame. However, one corner of the optimal frame does overlap with one of the virtual corners. Figure 7(b) has three requested frames exactly the same as those in (a) and one more big requested frame. It is interesting to see how the optimal frame changes after the big requested frame joined in the system. Figure 7(c) shows that if all input rectangles fall far way from each other, the algorithm functions as a chooser, which selects one input rectangle as the output. Since the algorithm searches optimum bottom-up, it picks the lowest requested frame as the solution. Fig-

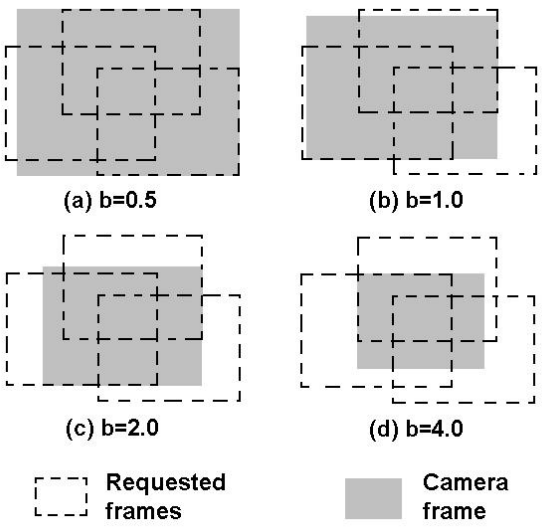

Figure 8: *Relationship between the optimal frame size and the choice of the b value in GIOM satisfaction metric in Subsection 3.2.*

ure 7(d) shows that a large requested frame does not necessarily yield a large optimal frame.

Figure 8 shows the relationship between the optimal frame size and the choice of the b value in the GIOM satisfaction metric. This demonstrates the tradeoff: large b leads to large optimal camera frames. As $b \rightarrow 0^+$, the optimal camera frame becomes the smallest frame that contains all request frame: $Area(c \cap r_i) = Area(r_i)\ \forall i$. If $b \rightarrow \infty$, then the optimal frame will converge to the rectangle area that most of requested frame insect each other. The parameter b allows us to find the best tradeoff between union and intersection.

6 Conclusions and future Work

This paper introduces the Collaborative Frame Selection problem, where multiple users share control of a single remote robotic camera. Each user requests a desired camera frame by drawing a rectangle over a static global image. The problem is to find a camera frame that maximizes the overall user satisfaction. We define a new metric for user satisfaction and study algorithms for solving the nonlinear optimization problem.

We review related work and study properties of the objective function and inherent constraints on the location of extremal points. We define "virtual corners" and prove that a global maximum must coincide with one of the virtual corners. We present algorithms and complexity analysis. For a discrete set of m distinct zoom levels and a discrete set of $w \times h$ pan and tilt pairs, we give an exact algorithm runs in $O((n + whm)\log^2 n)$. For continuous pan and tilt with m discrete zoom levels, we give an exact algorithm that runs in $O(n^2 m)$ time. An implementation of this algorithm can be found online at: http://www.tele-actor.net/sharecam/. This algorithm can be distributed to run in $O(nm)$ time at each client and in $O(n \log n + mn)$ time at the server.

In future work, we will consider versions of the problem with continuous zoom levels ($m = \infty$) and on approximation algorithms. We will also consider extensions to cases where the solution contains $p > 1$ frames: allowing p sequential views from one camera produces a path planning problem, and allowing p different cameras produces a variant of the p-center problem from facility location. 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 and reliabilities. We are interested in distributed algorithms that optimize performance under such uncertainties.

Acknowledgments

Our thanks to M. Overmars, V. Koltun, S. Har-Peled, A. Lim, S. Rao, D. Zhang, D. Halperin, J. Luntz, P. Wright, R. Bajcsy, D. Plautz, C. Cox, D. Kimber, Q. Liu, J. Foote, L. Wilcox, and Y. Rui for insightful discussions and feedback. Thanks also to J. Alton, M. Faldu, F. Hsu, and W. Guan, K. "Gopal" Gopalakrishnan, Ron Alterovitz, A. Levandowski, I. Y. Song, M. Mckelvin, A. Ho, and J. Vidales for their roles in the Alpha Lab at UC Berkeley.

References

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

[Agassounon et al., 2001] Agassounon, W., Martinoli, A., and Goodman, R. (2001). A scalable, distributed algorithm for allocating workers in embedded systems in embedded systems. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics Conference, Tucson, AZ, USA*, volume 5, pages 3367–3373.

[Bentley, 1980] Bentley, J. L. (1980). Multidimensional divide-and-conquer. *Communication of the ACM*, 23(4):214–229.

[Bohringer and Choset, 2000] Bohringer, K. F. and Choset, H., editors (2000). *Distributed Manipulation*. Kluwer Academic Publishers.

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

[Broder et al., 2000] Broder, A. Z., Charikar, M., Frieze, A., and M.Mitzenmacher (2000). Min-wise independent permutations.

Special Issue for STOC, Journal of Computer and System Sciences, 60(3):630–659.

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

[Canon, 2003] Canon (2003). **http://www.x-zone.canon.co.jp/WebView-E/all/list.htm**.

[Chan and Ioannidis, 1999] Chan, C. and Ioannidis, Y. (1999). Hierarchical cubes for range-sum queries. In *Proceedings of the Twenty-Fifth International Conference on Very Large Data Bases (VLDB), Edinburgh, Scotland, UK*, pages 675–686.

[Chen et al., 1999] Chen, R., Wu, Z., Wang, Y., and C. Dong, D. T. (1999). Cooperative assembly algorithm of multiple distributed agent based robotic system. In *30th International Symposium on Robotics*, pages 545–552.

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

[Crouch and Mazur, 2001] Crouch, C. H. and Mazur, E. (2001). Peer instruction: Ten years of experience and results. *American Journal of Physices*, 69(9):970–977.

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

[Geffner et al., 2000] Geffner, S., Agrawal, D., and Abbadi, A. E. (2000). The dynamic data cube. In *EDBT, Conference on Extending Database Technology*, pages 237–253.

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

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

[Goldberg and Siegwart, 2002] Goldberg, K. and Siegwart, R., editors (2002). *Beyond Webcams: An Introduction to Online Robots*. MIT Press.

[Goldberg et al., 2002] Goldberg, K., Song, D., Khor, Y., Pescovitz, D., Levandowski, A., Himmelstein, J., Shih, J., Ho, A., Paulos, E., and Donath, J. (2002). 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.

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

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

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

[Guttman, 1984] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proc. SIGMOD Conf., Boston, MA, USA*, pages 47–57.

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

[Hayes et al., 2001] Hayes, A., Martinoli, A., and Goodman, R. (2001). Swarm robotic odor localization. In *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, volume 2, pages 1073–1078.

[Ho et al., 1997] Ho, C., Agrawal, R., Megiddo, N., and Srikant, R. (1997). Range queries in OLAP data cubes. In *Proceedings of SIGMOD, ACM, Tucson, Arizona, USA*, volume 26, pages 73–88.

[Hu et al., 2001] Hu, H., Yu, L., Tsui, P. W., and Zhou, Q. (2001). Internet-based robotic systems for teleoperation. *Assemby Automation*, 21(2):143–151.

[Jain et al., 1999] Jain, A., Murty, M., and Flyn., P. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

[Jia and Takase, 2001] Jia, S. and Takase, K. (2001). A corba-based internet robotic system. *Advanced Robotics*, 15(6):663–673.

[Johnson, 1998] Johnson, D. (1998). *Applied Multivariate Methods for Data Analysis*. Duxbury Press.

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

[Ku et al., 2001] Ku, P., Winther, K., Stephanou, H., and Safaric, R. (2001). Distributed control system for an active surface device. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3417–3422.

[Lazaridis and Mehrotra, 2001] Lazaridis, I. and Mehrotra, S. (2001). Progressive approximate aggregate queries with a multi-resolution tree structure. In *Proc. of SIGMOD, Santa Barbara, CA, USA*, volume 30, pages 401–412.

[Liu et al., 2002] Liu, Q., Kimber, D., Wilcox, L., Cooper, M., Foote, J., and Boreczky, J. (2002). 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.

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

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

[Moon and Luntz, 2002] Moon, H. and Luntz, J. (2002). Distributed manipulation by superposition of logarithmic-radial potential fields. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1197–1202.

[Mumolo et al., 2000] Mumolo, E., Nolich, M., and Vercelli, G. (2000). Algorithms and architectures for acoustic localization based on microarray in service robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2966–2971.

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

[Paradias et al., 2001] Paradias, D., Kalnis, P., Zhang, J., and Tao, Y. (2001). Efficient olap operations in spatial data warehouses. In *Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD, Redondo Beach, CA, USA*, pages 443–459.

[Parker, 2002] Parker, L. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3):231–255.

[Rogoff et al., 1996] Rogoff, B., Matusov, E., and White, C. (1996). *Models of teaching and learning: Participation in a community of learners*. Oxford, England: Blackwell.

[Safaric et al., 2001] Safaric, R., Debevc, M., Parkin, R., and Uran, S. (2001). Telerobotics experiments via internet. *IEEE Transactions on Industrial Electronics*, 48(2):424–31.

[Sagawa et al., 2001] Sagawa, R., Nishino, K., Wheeler, M., and Ikeuchi, K. (2001). Parallel processing of range data merging. In *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, volume 1, pages 577–583.

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

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

[Yang and Widom, 2001] Yang, J. and Widom, J. (2001). Incremental computation and maintenance of temporal aggregates. In *Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany*, pages 51–60.

[Zhang, 2002] Zhang, D. (2002). *Aggregation Computation over Complex Objects*. PhD thesis, Department of Computer Science, University of California, Riverside.

[Zhang et al., 2001] Zhang, D., Markowetz, A., Tsotras, V. J., and Gunopulos, D. (2001). Efficient computation of temporal aggregates with range predicates. In *Proc. of Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Santa Barbara, CA, USA*, pages 237–245.

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