Systems and Algorithms for Collaborative Teleoperation

by

Dezhen Song

B.S. (Zhejiang University) 1995

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Industrial Engineering and Operations Research

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Fall 2004

The dissertation of Dezhen Song is approved:

Chair	Date
	Date
	Date
	Date
	Date

University of California, Berkeley

Fall 2004

Systems and Algorithms for Collaborative Teleoperation

Abstract

Systems and Algorithms for Collaborative Teleoperation

by

Dezhen Song

Doctor of Philosophy in Engineering - Industrial Engineering and Operations Research

University of California, Berkeley

Professor Ken Goldberg, Chair

Collaborative Teleoperation (CT) systems allow many users to simultaneously share control of a single remote physical resource such as a robot or human "explorer", with applications in education, health care, journalism, and entertainment. A primary challenge is scalable methods for computing consensus commands. This thesis combines results from two networked CT systems: one with a robotic webcamera (the Co-Opticon) and one with a human explorer (the Tele-Actor).

In the Co-Opticon, n users share control of a single robotic pan, tilt, zoom webcamera. We formulate a new resource allocation problem and a series of exact and approximate geometric algorithms for solving it. We propose a new similarity metric for the degree of satisfaction for each user, which improves over the nonlinear Intersection Over Union (IOU) metric. For an approximation bound ϵ, our best algorithm runs in $O((n+1/\epsilon^3)\log^2 n)$ 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. These algorithms also apply to satellite image selection problem with applications in weather prediction, disaster response, search and rescue, surveillance, and defense.

The Tele-Actor system allows groups of online users to collaboratively "direct" a human exploring a remote environment. We develop an unsupervised scoring metric based on density based clustering to assess individual behavior in terms of leadership and collaboration.

I have implemented both systems and they have been extensively field tested with students and online users. Future research will focus on how new technologies such as Internet 2, broadband videoconferencing, and wireless networking can be used to enhance collaborative teleoperation and its applications in education and health care.

Professor Ken Goldberg
Dissertation Committee Chair

To my parents and to Ye

Contents

List of Figures

List of Tables

Acknowledgments

My gratitude to those who helped me to get through my PhD program in Berkeley.

Special thanks to Ken Goldberg for patiently guiding me in the last four years. Ken is my advisor and my best friend. Ken is always willing to spend time with me and encourage me to attack new problems. From research to other aspect of life, Ken sets up a great model for me to follow. I have to work very hard to keep up with him.

Thanks to Frank van der Stappen for his insightful suggestions and inspiring discussions. To some degree, Frank is my second thesis advisor. I am fortunate to have such a great mentor and friend to collaborate with.

Thanks to Ilan Adler, Satish Rao, and Andrew Lim for serving as my qualifying and thesis committee members. Their inputs helped me to improve my research and refine my results.

Thanks to R. Volz, T. Ioerger, R. Gutierrez-Osuna, J. Chen, D. Pescovitz, J. Donath, E. Paulos, S. Har-Peled, V. Koltun, D. Plautz, A. Pashkevich, C. Cox, Y. Rui, J. Foote, D. Zhang, and Q. Lui, for insightful discussions and feedback. Thanks to K. "Gopal" Gopalakrishnan, R. Alterovitz, A. Levandowski, J. McGonigal, I. Song, W. Zheng, M. Faldu, A. Ho, M. Mckelvin, A. Ho, B. Chen, W. Guan, R. Aust, M. Metz, V. Colburn, Y. Khor, J. Himmelstein, J. Shih, F. Hsu, M. Last, and J. Vidales, for their contributions to my PhD projects.

Chapter 1

Introduction

1.1 Tele-operation

A “telerobot” is a remotely controlled machine equipped with sensors such as cameras and the means to move through and interact with a remote physical environment. NASA’s Mars Sojourner is a well-known example. The Sojourner telerobot, like almost all telerobots to date, is controlled by a single human operator.

Since Nikola Tesla demonstrated a first radio-controlled boat in New York City in 1898 [114], teleoperation has a history of more than a century. Goertz demonstrated one of the first bilateral simulators in the 1950’s at the Argonne National Laboratory[42]. Remotely operated mechanisms have long been desired for use in inhospitable environments such as radiation sites, undersea [7] and space exploration [9]. At General Electric, Mosher [91] developed a complex two-arm teleoperator with video cameras. Prosthetic hands were also applied to teleoperation [115]. More recently, teleoperation is being considered for medical diagnosis [8], manufacturing [41] and micromanipulation [105]. See Sheridan [108] for an excellent review of the extensive literature on teleoperation and telerobotics.

Networked robots, controllable over networks such as 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.

The Mercury Project was the first Internet robot [47, 46]; it went online in August 1994. Working independently, a team led by K. Taylor and J. Trevelyan at the University of Western Australia demonstrated a remotely controlled six-axis telerobot in September 1994 [25, 62]. There are now dozens of Internet robots online, a book from MIT Press [49], and an IEEE Technical Committee on Internet and Online Robots. See [63, 104, 67, 66, 70, 88, 57, 93, 77] examples of recent projects.

1.2 What is a Collaborative Teleoperation System

By "collaborative" we mean a system where a number of participants simultaneously share control. We define a "collaborative telerobot" as a telerobot simultaneously controlled by many participants, where input from each participant is combined to generate a single control stream.

Collaborative Telerobotics (CT) is a novel approach to teleimmersion and teleworking. With CT, participants collaborate rather than compete for access to valuable resources such as historical and scientific sites. Collaboration is a crucial ingredient for education and teamwork. A scalable infrastructure for CT, compatible with the Internet, would allow large groups of students or researchers to simultaneously participate in remote experiences. For example, CT can allow groups of disadvantaged students to collaboratively steer a telerobot through a working steelmill in Japan or the Presidential Inauguration, and allow groups of researchers to collaboratively steer a telerobot around a newly active volcano or a fresh archaeological site.

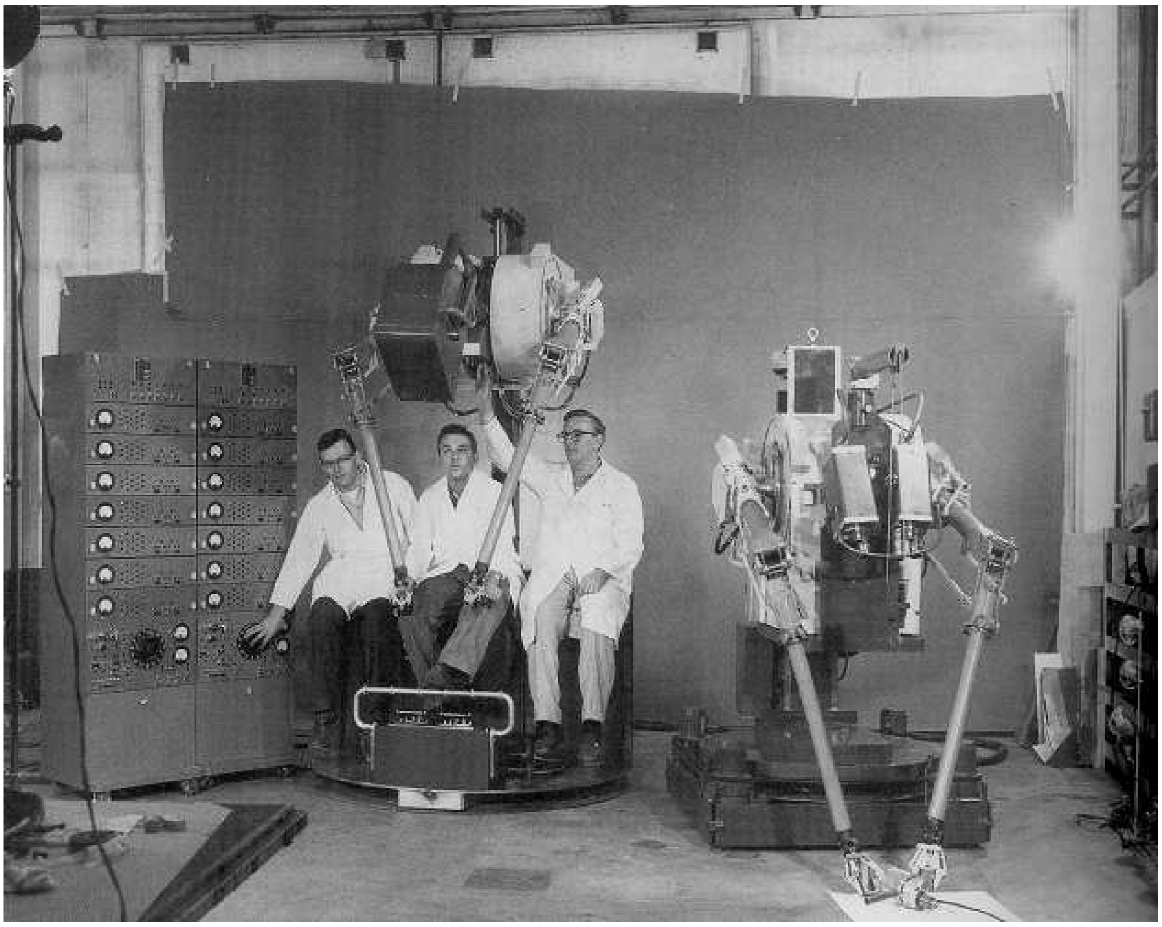


Figure 1.1: *Examples of Collaborative Control System*

Figure 1.1 illustrates a non-remote collaborative control architecture. Before we consider the details of problems, a preliminary question is: Can a group of many participants simultaneously driving the motion of a single resource achieve anything resembling coherent control?

Anecdotal evidence with Cinematrix [35, 20], a commercial audience interaction system, suggests that collaborative motion control is not only possible but fairly robust to deviations in individual behavior. The inventors of Cinematrix, Loren and Rachel Carpenter, performed a series of experiments in large theaters in the 1990s. Each audience member is given a plastic paddle, colored red on one side and green on the other. By rotating his or her paddle each player simultaneously provides input. Overhead cameras detect which color is being presented by each participant in real time. The camera output is used to drive a live display projected onto the front screen of the theater. The average level of red or green conveyed by the group provides an aggregate audience signal that is re-computed several times a second.

The theater is divided down the central aisle and a cursor is projected on the screen. Participants on the right control the horizontal motion of the cursor, participants on the left control the vertical motion. A large circle is displayed on the screen and the audience is requested to move the shared cursor to trace a trajectory around the circle. Since each player only controls one small component of the average signal, and the participants are a heterogeneous group with different personalities, one might conjecture that the shared cursor motion would resemble random Brownian motion.

But in repeated experiments, groups of participants were quickly able to adapt their individual paddle signals to achieve coherent control of the shared cursor. Groups were not only able to track given trajectories, but to play competitive games such as Pong and Pac Man, and even to collaboratively control an airplane flight simulator! Audiences ranged from 5000 graphics professionals at Siggraph 1991 to groups of unruly high school students in Pittsburgh PA.

1.3 Characteristics of CT Systems

In our report, although our research can be applied to a wider class of CT systems, we focus on web-based CT systems in particular. Web-based CT systems utilize Internet as their media and usually do not require users to install specialized software, which makes them widely accessible. A CT system usually has the following characteristics,

- Sharing valuable resources and providing live access to remote environments:

  As a special type of teleoperation system, CT allows multiple people to share a single resource simultaneously. People are very interested in valuable resources like robots. They want to learn more about robots and play more with them. The problem is that they are too expensive to be affordable by a normal user. On the other hand, many advanced robots are not fully utilized in research labs and universities. Web-based CT systems can provide public access to those valuable resources, which is proven to be very usefully for education. For example, Colton et al. [61] designed an online heat exchanger in MIT, which allows a class of students to do experiments online.

- User interaction:

  CT system' interfaces usually provide functionality to allow people to see others' decisions and to be involved in others' decision processes. Users can interact with each other using the bulletin boards, chat rooms and voice/video conferencing systems. The exchanging of ideas helps them to get a better understanding of what is happening in the system. This feature can be useful for training and educational purposes. For example, novice users can observe experienced users' behavior and learn from it. This also fits the idea of education: teamwork is a key element in education at all levels [103, 23, 22].

- Collaboration improves reliability:

  A CT system combines users' requests to control a single shared device. This makes CT systems less sensitive to individual errors and malicious behavior. The result from Cinematrix [20] confirms that collaborative control may be surprisingly robust in practice. What's more interesting is that group diversity may actually *improve* performance [44]. Similar effects have been observed in very different contexts [68].

1.4 Our Research on CT Systems

In this thesis, we focus on systems and algorithms for CT systems. We begin with the two web-based CT systems developed in the ALPHA Lab, IEOR Department, UC Berkeley:

- The Co-Opticon/Satellite Frame Selection: A controllable camera that allows many clients to share control its camera parameters. Users indicate where they want to see by drawing rectangles on a panoramic image. The algorithm will comput an optimal camera frame with respect to the user satisfaction function. (See: http://www.co-opticon.net). The Satellite Frame Selection problem is a generalized version of the Co-Opticon problem. A two-axis satellite camera can access a wide rectangular region of the Earth but can only take high resolution images on specific regions. For multiple and competing client requests, how to set satellite parameters to maximize total benefit presents variations of the Co-Opticon problem, and

- The Tele-Actor: A human equipped with audio/video device and controlled by a group of online users. Users indicate their intensions by positioning their votes on a 320×240-pixel voting image during the voting interval. Votes are collected at the server and used to determine the Tele-Actor's next action based on the most requested region on the voting image . (See: http://www.tele-actor.net).

We report algorithms developed for these systems. For the Co-Opticon system, we focus on fast online algorithms. We develop approximation algorithms and derive formal approximation bound that characterize the tradeoff between accuracy and solution. For Satellite Frame Selection problem, we focus on exact algorithms that yield maximum reward. For tele-actor system, our algorithmic efforts focus on how to compute consensus regions, measure user collaboration, and score individual user performance.

Chapter 2

The Co-Opticon System: Interface, System Architecture, and Implementation of a Collaboratively Controlled Robotic Webcam

2.1 Introduction

Robotic webcameras with pan, tilt, and zoom controls are now commercially available and are being installed in dozens of locations[1] around the world. In these systems, the

[1]See: http://www.x-zone.canon.co.jp/WebView-E/index.htm

camera parameters can be remotely adjusted by viewers via the Internet to observe details in the scene. Current control methods restrict control to one user at a time; users have to wait in a queue for their turn to operate the camera. In this chapter we describe the Co-Opticon, a new system that eliminates the queue and allows many users to share control of the robotic camera simultaneously.

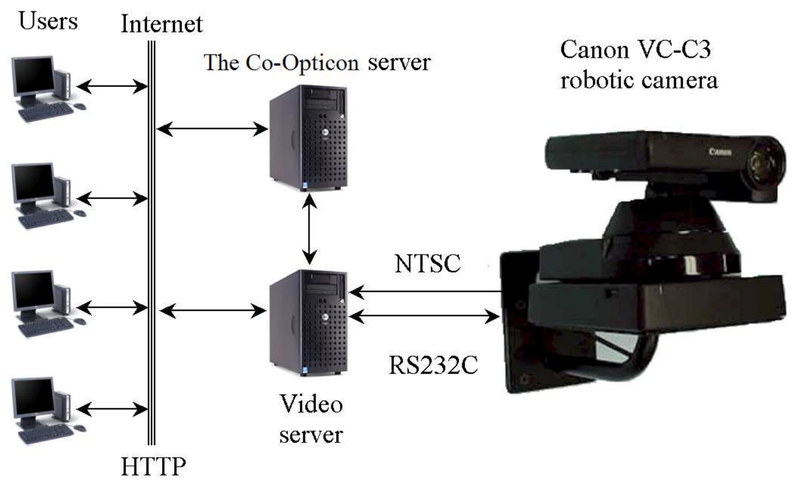


Figure 2.1: *The Co-Opticon System Architecture. http://www.tele-actor.net/sharecam/*

As illustrated in Figure 2.1, the Co-Opticon system includes the camera and two servers that communicate with users via the Internet. Streaming video is captured at the camera server and streamed back to the remote users using a Java interface. User responses are collected at the Co-Opticon server and used to compute optimal camera positions, which are sent to camera server to control the camera.

The Co-Opticon's Java-based interface includes two image windows, one fixed for user input and the other a live streaming video image. The interface collects requested camera frames (specified as desired rectangles) from n users, computes a single camera frame based on all inputs, and moves the camera accordingly. Below we describe system details and two frame selection models based on user "satisfaction".

2.2 Related Work

The Co-Opticon is an example of Collaborative Telerobotics, in this case the telerobot is a camera with 3 degrees of freedom. In the taxonomy proposed by Tanie et al. [21], the Co-Opticon 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.

The Internet provides a low-cost and widely-available interface that can make physical resources accessible to a broad range of participants. There are now thousands of webcams, dozens of "online robots", a book from MIT Press [49], and an IEEE Technical Committee on Internet and Online Robots.

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 [57, 63, 67, 66, 70, 77, 88, 93, 104] for examples of recent projects.

Tanie, Matsuhira, Chong, et al. [21] proposed the following taxonomy for teleoperation systems: Single Operator Single Robot (SOSR), Single Operator Multiple Robot (SOMR), Multiple Operator Multiple Robot (MOMR). and Multiple Operator Single Robot

(MOSR). Most online robots are SOSR, where control is limited to one operator at a time. Tanie et al. analyzed an MOMR system where each operator controls one robot arm and the robot arms have overlapping workspaces. They show that predictive displays and scaled rate control are effective in reducing pick-and-place task completion times that require co-operation from multiple arms.

A number of SOSR systems have been designed to facilitate remote interaction. Paulos and Canny's Personal Roving Presence (PRoP) telerobots, built on blimp or wheeled platforms, were designed to facilitate remote social interaction with a single remote operator [95, 96]. Fong, Thorpe and colleagues study SOSR systems where collaboration occurs between a single operator and a mobile robot that is treated as a peer to the human and modeled as a noisy information source [36]. Related models of SOSR "cobots" are analyzed in [2, 11, 36, 78, 109].

In an MOMR project by Fukuda, Liu, Xi, and colleagues [30], two remote human operators collaborate to achieve a shared goal such as maintaining a given force on an object held at one end by a mobile robot and by a multi-jointed robot at the other. The operators, distant from the robots and from each other, each control a different robot via force-feedback devices connected to the Internet. The authors show both theoretically and experimentally that event-based control allows the system to maintain stable synchronization between operators despite of variable time-lag on the Internet.

MOMR models are also relevant to online collaborative games such as *Quake*, where players remotely control individual avatars in a shared environment.

In SOMR systems, one tele-operator or process controls multiple robots. This bears some relation to Cooperative (behavior-based) robots, where groups of *autonomous* robots interact to solve an objective [3]. Recent results are reported in [18, 28, 102, 99].

One precedent of an online MOSR system is described in McDonald, Cannon and colleagues [85]. For waste cleanup, several users assist in waste cleanup using Point-and-Direct (PAD) commands [19]. Users point to cleanup locations in a shared image and a robot excavates each location in turn. In this Internet-based MOSR system, collaboration is serial but pipelined, with overlapping plan and execution phases. The authors demonstrate that such collaboration improves overall execution time but do not address conflict resolution between users.

In [44] Goldberg and Chen analyze a formal model of collaborative control and in [45] describe Internet-based MOSR system that averaged multiple human inputs to simultaneously control a single industrial robot arm. In [50, 51] we 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”.

Research on controllable webcams or Internet cameras are focus on two perspectives: system architectures and applications. Desmet, Verkest, Mignolet et al. [26, 64, 119] designed webcams using reconfigurable hardware and embedded software. They implemented a secure VPN (Virtual Private Network) with 3DES encryption and Internet camera server (including JPEG compression). Brooks and McKee [12] implemented an automated camera which is placed during teleoperation using Visual Acts theory and architecture to provide operators with task relevant information in a timely manner. The applications of webcams is not limited to surveillance [69] or teleconferencing [71, 76, 97]. Schmid, Maule, and Roth [106] used a controllable webcam to perform all the tests for industrial robots given by ISO 9283 “Performance criteria and related test methods”. Pollak and Hutter [100] installed a Phillips webcam on an Olympus BX60 light microscope to record movies of investigated samples. Zhang, Navab, and Liou [124] used webcams to creat an interactive sales model for web customers.

In independent work, Kimber, Liu, Foote et al describe a multi-user robot camera in [71, 76]. The application is designed for videoconferencing system. They use multiple cameras in the systems: panoramic cameras and a pan-tilt-zoom camera. Panoramic cameras generate a dynamic panoramic view of the conference site. Users control the pan-tilt-zoom camera by drawing on panoramic view. The system is well suitable for videoconferencing environment, where illumination condition is constantly good so that the image quality of panoramic view can be guaranteed. We believe multiple camera systems are good but not necessary for scenic sites where dynamic information is not necessary. The panoramic image can be generated by the same pan-tile-zoom camera resulting in less bandwidth requirement.

An earlier paper [112], published in the Workshop on Algorithmic Foundations of Robotics, formulated the Co-Opticon problem geometrically and reported initial results on exact algorithms: for n users and m zoom levels, the exact algorithm runs in $O(n^2m)$ time. Har-Peled et al. [58] improved the exact algorithm to $O(mn^{3/2}\log^3 n)$ and proposed a near linear $\epsilon-$approximation algorithm. In [111], we describe approximate and distributed algorithms for solving the Co-Opticon frame selection problem.

2.3 The Co-Opticon Interface

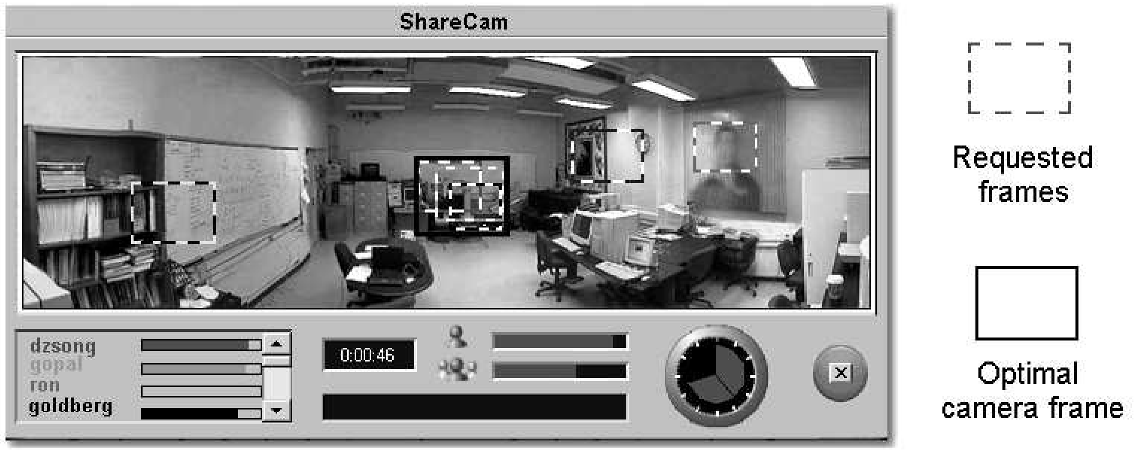


Figure 2.2: *This figure illustrates the Co-Opticon's Java-based user interface, which currently runs on most Windows based PCs. Users view two windows. One (not shown) displays a live video stream as captured by the robotic camera. The second window, illustrated here, contains the user interface. The panoramic image is a fixed photo of the camera's reachable range of view. The snapshot above shows 6 active users listed in the scrollable window at the left. Each user requests a camera frame by positioning a dashed rectangle over the panoramic image. Based on these requests, the algorithm computes an optimal camera frame (shown with solid rectangle), and servoes the camera accordingly to displays the resulting live video stream. The horizontal bars indicate levels of user satisfaction as described below. The system is installed in our research lab at Berkeley and moved outdoors in June 2003. See: http://www.co-opticon.net*

The Co-Opticon interface facilitates interaction and collaboration among remote users. Users register online to participate by selecting a characteristic color and submitting their email address to the Co-Opticon server, which stores this information in our database and immediately sends back a password via email. The server also maintains a tutorial and an FAQ section to familiarize new users with how the systems works.

The Co-Opticon interface contains two windows: The video window shows the current camera view. Figure 2.2 illustrates the panoramic window and the Co-Opticon user interface.

The interface also facilitates chat between users. Each user can type in a short sentence, which is displayed underneath his/her requested frame in the panoramic image. A clocklike timer is located at the bottom right of the interface indicating the time before the next camera movement (typically 5-10 seconds).

2.4 Hardware

The Co-Opticon server is an AMD K7 950Mhz PC with 1.2GB SDRAM connected to a 100Mbs T3 line. The camera server is an AMD K7 950Mhz PC with 640MB SDRAM connected to an 100Mbs T3 line at the remote site. It has a video-capture card, which captures video at 320×240 resolution. It also serves as video server running InetCam [2] software to broadcast video.

We used the Canon controllable camera, model VC-C3. A comparable camera is available from Sony. The Canon camera has motorized pan, tilt and zoom with a 10x power zoom lens. It has PAL, composite, and S-video output with a resolution of 450 horizontal lines. It can communicate with a PC via a RS232C link at 14,400bps. Its pan, tilt, and zoom speed is 76 degrees per second at maximum and 0.5 degrees per second at minimum. It has an accuracy of 0.5 degrees and a 380,000 pixel CCD array.

2.5 Software

As illustrated in Figure 2.3, custom software includes: (1) the Co-Opticon server, (2) the camera control software and video capturing package at the video server, and (3)

[2] http://www.inetcam.com

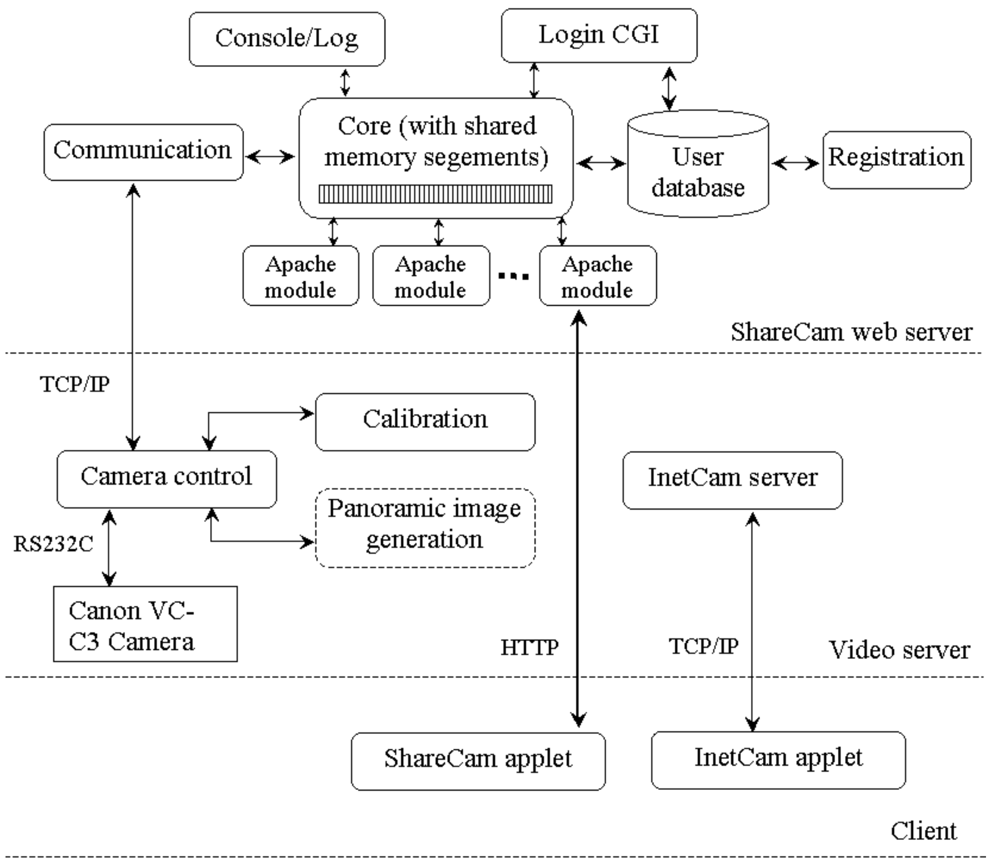


Figure 2.3: *The Co-Opticon system software diagram.*

the client side Co-Opticon Java applet.

The Co-Opticon server runs Mandrake Linux 9.0 and the Apache web server 1.3.26. All modules are written in GNU C++ with optimization of running speed. The Co-Opticon server package consists of core process, Apache modules, communication process, user databases, registration module, console/log module, and login CGI script. The customized Apache module deals with communication between web clients and the server via HTTP. It accepts the requested frame from a client and sends him/her the requested frames of others every second. It can be viewed as a CGI script but with much higher scalability. The communication module connects to the video server via a socket link to send camera control commands. A console/log module allows us to monitor and record system status in real time.

The overall design emphasizes data sharing among all processes. Collaborative control requires that all clients are able to see each other's information in real time. This is achieved by sharing memory segments among all server processes. Therefore the shared memory segment managed by the core process is the key data structure.

Clients download two applets: the Co-Opticon applet and the InetCam applet. The Co-Opticon applet is a customized software, which is shown in Figure 2.2. Part of the frame selection computation is done at the client side, which is implemented in the Co-Opticon applet. The Co-Opticon applet is written in Java 1.1.8 to ensure the compatibility with most browsers. The InetCam applet is a third party software that functions as a video terminal.

The video server package includes camera control, InetCam server, calibration, and panoramic image generation. The camera control module written in Microsoft Visual C++ is the primary module. It accepts camera control commands from the Co-Opticon server and translates it into the RS232C protocol, which is built on packages provided by Lawrence Berkeley National Laboratory[3].

[3]http://www-itg.lbl.gov/mbone/devserv/

2.6 Frame Selection Models

In this section, we will present two frame selection models. We begin with a review of definitions and notation. More details can be found in the next chapter.

We consider two models for the optimal camera frame, the first is memoryless based only on the current set of frame requests. The second is a temporal model based on the history of frame requests with exponentially decaying weights.

2.6.1 Memoryless Frame Selection Model

In the Co-Opticon system, c is a vector of camera parameters that users can control. Let c define a camera frame $[x, y, z]$, where x, y specify the center point of the frame, which is corresponding to pan and tilt, and z specifies size of the frame, which corresponds to zoom level. c defines a rectangular camera frame (the camera has a fixed aspect ratio of 4:3). User i requests a desired frame r_i. Given requests from n users, the system computes a single global frame c^* that will best satisfy the set of requests.

We define a Generalized Intersection Over Maximum (GIOM) metric for user "satisfaction" $s(c, r_i)$ based on how the user's requested frame r_i compares with a candidate camera frame c. Each of n users submits a request. Let

$$s(c) = \sum_{i=1}^{n} s_i(r_i, c) \tag{2.1}$$

In the memoryless frame selection model, we want to find c^*, the value of c that maximizes $s(c)$ based only on the current set of requests:

$$\max_c s(c).$$

In each motion cycle, we servo the camera to this frame.

2.6.2 Temporal Frame Selection Model

An alternative frame selection model is based on the history of user frame requests over multiple motion cycles. We extend equation 2.1 using a weighted sum of the user satisfaction. In this case total satisfaction is a function of time t:

$$s(c,t) = \sum_{i=1}^{n} \alpha_i(t) s_i(r_i(t), c(t)) \tag{2.2}$$

where the weight $\alpha_i(t)$ for user i is a function of the user's previous "dissatisfaction" level: $u_i(t) = 1 - s_i(r_i(t), c(t))$. One candidate form for weights is

$$\alpha_i(t) = \sum_{k=0}^{t-1} \frac{u_i(k)}{2^{t-1-k}}$$

which yields the recursive formulation:

$$\alpha_i(t) = u_i(t-1) + \alpha_i(t-1)/2$$

If user i does not get satisfied by the camera frame computed during the current frame, his weight $\alpha_i(t)$, will increase over future motion cycles, eventually dominating the weights of other users to satisfy his desired frame request. In this sense fairness is guaranteed over time.

These frame optimization problems can be solved with exact algorithms [112] or fast new approximation algorithms in next chapter.

Figure 2.4 shows four examples with the Memoryless Frame Selection model. Note that the optimal frame grows in image (b) after a large requested frame is added. In Figure

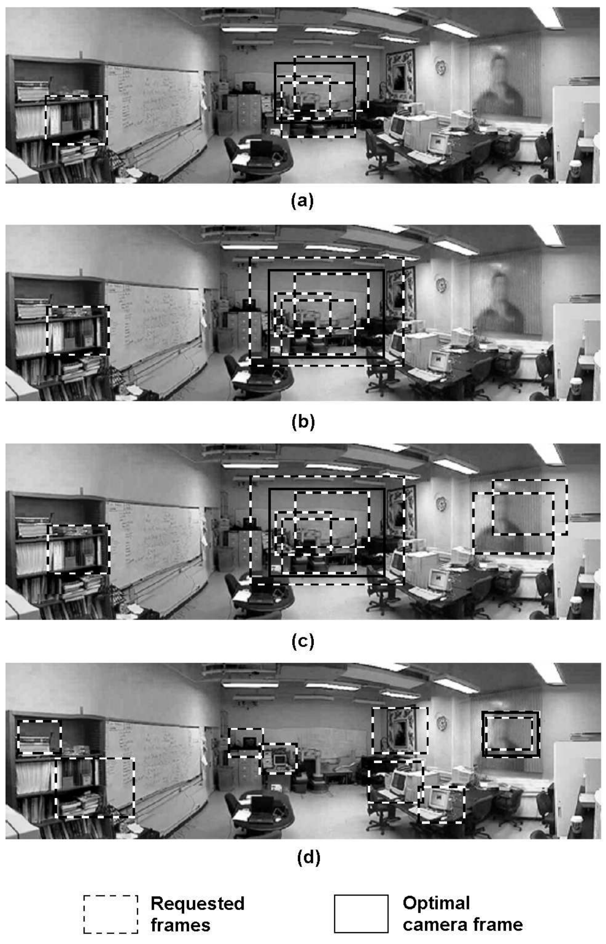


Figure 2.4: *Examples using Memoryless Frame Selection model defined by equation 2.1. Four different sets of requested frames and the corresponding optimal frame are displayed. Note that the resulting frame is very different than what would be determined by simple averaging, and that some requests never get satisfied.*

2.4(c), two more frames are requested. Since they can not compete with the central group of requested frames, the optimal frame remains unchanged. Figure 2.4(d) shows a case with all but two requested frames disjoint, the algorithm selects a frame that covers the two overlapping frames. Figure 2.4 also illustrates that some users can be starved indefinitely.

Figure 2.5 shows four examples with the Temporal Frame Selection model, where frame selection is based on user satisfaction over multiple motion cycles. A sequence of

4 motion cycles is illustrated with the same set of requested frames. Note that with this model, the camera frame changes to balance overall user satisfaction over time.

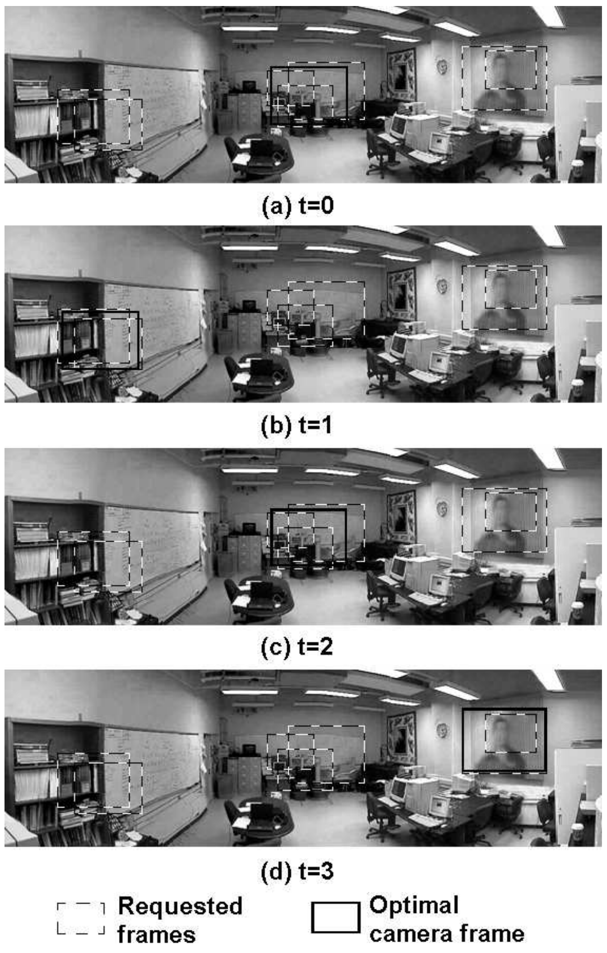
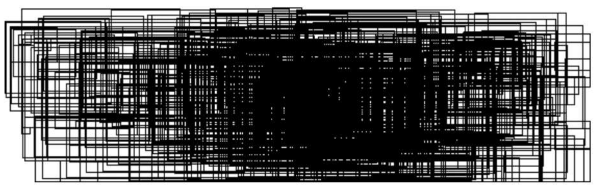


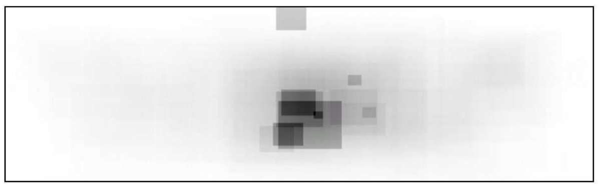
Figure 2.5: *Examples with the Temporal Frame Selection Model defined by equation 2.2. The set of requested frames is held constant, but weights evolve so that the camera frame changes to facilitate "fairness".*

2.6.3 Experiments

The Co-Opticon system went online in June of 2002 with the camera installed in our Alpha Lab from June 8, 2002 to February 2003 as shown in the previous figures. An illustration of the total requested frames is shown in figure 2.6.



(a) 4822 Requested frames



(b) Interest density distribution in grayscale

Figure 2.6: *Data from June 8, 2002 to February 6, 2003.*

Figure 2.6(a) displays all 4822 requested frames for the experiment duration. We are interested in how user interest is distributed in the panorama. To compute the interest distribution, we define $g(x, y)$ be the interest for point (x, y) in gray scale, i.e. $0 \leq g(x, y) \leq 255$, $r_j : 1 \leq j \leq 4822$ be the j^{th} requested frame, and an indicator variable,

$$I(x, y, j) = \begin{cases} 1 & \text{if } (x, y) \in r_j \\ 0 & \text{otherwise} \end{cases}$$

Say a darker point means more interest, the interest for point (x, y) is $g(x, y)$, and define $g_{max} = \arg\max_{(x,y)} g(x, y)$,

$$g(x, y) = 255(1 - \frac{\sum_{j=1}^{4822} I(x, y, j)}{g_{max}}).$$

We compute $g(x, y)$ for each point in the panorama and generate the figure 2.6(b). As shown in the figure, the most popular region is the center of the camera workspace, looking at the Adept robot arm in our lab, where one our colleague was often performing robot calibration tests.

The Co-Opticon system was moved to an outdoor location on the UC Berkeley campus in June 2003, and is available online at http://www.co-opticon.net. As shown on data of June 28, 2004, we have 1360 registered users and have received 71050 requested frames.

2.7 Conclusions

This chapter describes the Co-Opticon, a MOSR teleoperation system that allows a group of Internet users to simultaneously share control of a pan, tilt, and zoom camera. We described the Co-Opticon interface, system architecture, and experiments with two frame selection models.

Chapter 3 describes how to solve the Frame Selection problem.

Chapter 3

The Co-Opticon Algorithms: Approximate and Distributed Algorithms for a Collaboratively Controlled Robotic Webcam

In last Chapter, we have introduced the “Co-Opticon” system, which is a new system that eliminates the queue, allowing many users to share simultaneous control of the camera. In this Chapter, we concentrate on how to solve the frame selection problem for the Co-Opticon: how to find a camera frame that maximizes a measure of user satisfaction.

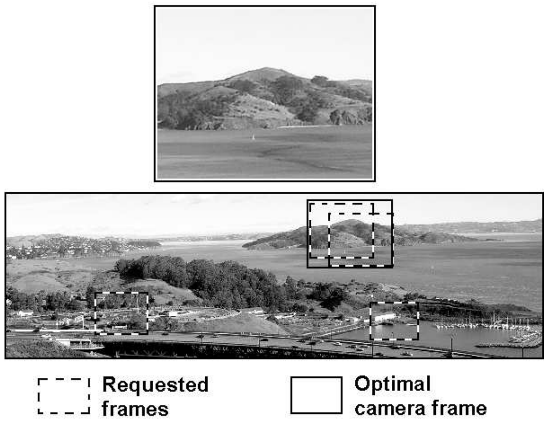


Figure 3.1: *Co-Opticon's Java-based interface on the Internet. The user interface includes two image windows. The lower window displays a fixed "panoramic" image based on the camera's full workspace (reachable field of view). Each user requests a camera frame by positioning a dashed rectangle in the 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.*

As illustrated in Figure 3.1, the Co-Opticon 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 3.2. We present a grid-based approximation algorithm: given n triangulated user requests, and an 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. We also develop an Branch and Bound (BnB) like approach, which can reduce the constant factors at least by 70%. Using newly developed data structure in database research, we can further improve the running time to $O((n + 1/\epsilon^3)\log^2 n)$.

3.1 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 Chapter 2 for examples of recent projects.

In [45, 44], Goldberg and Chen describe an Internet-based MOSR system that averaged multiple human inputs to simultaneously control a single industrial robot arm. In [50, 51] 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".

The Co-Opticon is a system for a robot camera, where operator inputs are frames rather than points. The Co-Opticon 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 [86]. Using a geometric approach, Eppstein [32] gave an $O(n \log^2 n)$ algorithm for the the planar 2-Center problem. Halperin et al. [55] 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 [92].

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 [122]. The spatial objects could be points, intervals, or rectangles. Aggregation over points is a special case of the orthogonal range search queries from computational geometry. Agarwal and Erickson [1] provide a review of geometric range searching and related topics. Grossi and Italiano [52, 53] proposed the cross-tree data structure, a generalized version of a balanced tree, to speed up range search queries in high-dimensional space. The continuity of the solution space of our problem makes it impossible to simply evaluate a fixed set of candidate frames through queries.

Branch-and-bound (BnB) is a general problem solving paradigm, especially useful for finding optimal solutions to most NP-hard combinatorial problems [123]. BnB can efficiently reduce the search space as the computation proceeds. Lin and his colleagues [75] applied BnB to solve protein backbone nuclear magnetic resonance peaks assignment problem. Mitchell and Brochers analyze BnB performance with respect to $0 - 1$ Mixed Integer Non-Linear Programming problems [89]. A comprehensive review of BnB can be found in Papadimitriou's book [94].

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 [14, 90]. It is possible to interpret the satisfaction function as a special "lifted" potential field with some modifications.

In independent work, Kimber, Liu, Foote et al develop a multi-user robot camera for videoconferencing [71, 76]. 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 [112], Song, van der Stappen, and Goldberg gave the first algorithms for the frame selection 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 $O(n^2m)$ exact algorithm for a continuous pan and tilt with discrete m levels of zoom. Har-Peled et al. [58] improved the exact algorithm to $O(mn^{3/2}\log^3 n)$ and proposed a near linear $\epsilon-$approximation algorithm. In the present chapter, 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. Chapter 2 describes system interface, architecture, and implementation.

3.2 Problem Definition

In this section, we formulate the Co-Opticon frame selection 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 Co-Opticon 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 $Z = [\underline{z}, \bar{z}]$ 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 [112], 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,

$$s_i(r_i, c) = \frac{Area(r_i \cap c)}{Area(r_i)} \min(\frac{Size(r_i)}{Size(c)}, 1) \tag{3.1}$$

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

$$s(c) = \sum_{i=1}^{n} s_i(r_i, c) \tag{3.2}$$

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.

3.3 Algorithms

We begin with a grid-based approximation algorithm and derive formal approximation bounds that characterize the tradeoff between speed and accuracy. We present a Branch and Bound like idea to reduce the constant factor in the algorithm. The speed of the algorithm can be further improved using a new data structure. We then describe a distributed version of the algorithm.

3.3.1 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,

$$\begin{aligned} L &= \{(pd, qd, rd_z) | pd \in [0, w], qd \in [0, h], \\ & \quad rd_z \in [\underline{z}, \bar{z} + 2d_z], p, q, r \in \mathcal{N}\} \end{aligned} \tag{3.3}$$

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 $g = \bar{z} - \underline{z}$. According to Equation 3.2, 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

$$O((wh/d^2)(g/d_z)n). \tag{3.4}$$

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:

$$s(\tilde{c})/s(c^*) \; = \; 1 - \epsilon \tag{3.5}$$

Since equation 3.2 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:

- As illustrated in Figure 3.2, 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

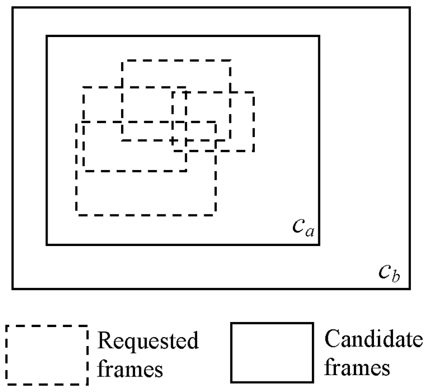


Figure 3.2: *Example illustrating the lower bound on solution quality.*

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 $p_{bi} \geq p_{ai}$,
- $I_a = \{i | r_i \cap c_a \neq \emptyset\}$ be the set of users whose requested frames intersect with frame c_a.
- $I_b = \{i | r_i \cap c_b \neq \emptyset\}$ be the set of users whose requested frames intersect with frame c_b. $I_a \subseteq I_b$
- I' be the set of users whose requested frames intersect with frame c_a and are bigger than the c_a, $I' = \{i | i \in I_a \text{ and } z_i \geq z_a\}$.
- I'' be the set of users whose requested frames intersect with frame c_a and are bigger than the 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 have,

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

and because $I_a \subseteq I_b$,

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

Therefore,

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

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

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

Then

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

So,

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

□

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,

$$|x - x'| \leq d/2 \quad \text{and} \quad |y - y'| \leq d/2, \tag{3.6}$$

$$z' \leq \lceil \frac{3z + d}{3d_z} \rceil d_z. \tag{3.7}$$

If we choose $d = 3d_z$, then

$$z' \leq z + 2d_z. \tag{3.8}$$

Proof. The center (point B in figure 3.3) 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 3.3. 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 3.6 have to be true.

Recall that d is the spacing of the lattice. To ensure the point O is the nearest lattice point, equation 3.6 mean the point B must satisfy following constraints,

$$|\overline{OB}| \sin \alpha \leq d/2, \text{and } |\overline{OB}| \cos \alpha \leq d/2. \tag{3.9}$$

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 $\hat{c}$:

$$z' = \lceil \hat{z}/d_z \rceil d_z \tag{3.10}$$

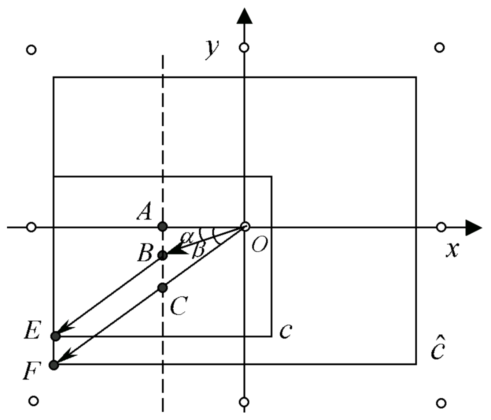


Figure 3.3: *Relationship between frame c and the smallest frame $\hat{c}$ on the lattice that encloses it. In the figure, $\alpha = \angle AOB$, $\beta = \angle AOC$, and the frame c is centered at point B.*

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,

$$x_F \leq x_E \text{ and } \; y_F \leq y_E. \tag{3.11}$$

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 3.3 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

$$|\overline{OC}| = |\overline{OF}| - |\overline{CF}| = |\overline{OF}| - |\overline{BE}|. \tag{3.12}$$

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 3.9,

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

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 3.10,

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

If we choose $d = 3d_z$, equation 3.7 can be simplified as,

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

$\square$

Theorem 1. Recall $\underline{z}$ 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

$$c = \frac{2d_z}{\underline{z} + 2d_z}$$

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 3.8 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 $z^* \geq \underline{z}$, so

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

□

Theorem 1 says that the approximation bound $\frac{2d_z}{\underline{z}+2d_z}$ 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

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

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 Co-Option 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 $(wh/d^2)(g/d_z) = \frac{whg}{(9/4)(\frac{\epsilon}{1-\epsilon})^3\underline{z}^3}$ points. According to equation 3.2, each point will take $O(n)$ time. Removing constants, ϵ approaches zero, so computation time approaches $O(n/\epsilon^3)$ □

3.3.2 Branch and bound implementation.

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

Corollary 1. *Given a frame $\hat{c}$ is currently the best known solution, if a candidate frame (x, y, z) does not satisfy the following condition,*

$$s(x, y, z) \geq s(\hat{c})(\underline{z}/z), \tag{3.13}$$

then the candidate frame does not contain any optimal frame.

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

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

Since c^* is the optimal solution, then $s(\hat{c}) \leq s(c^*)$. Therefore, Equation (3.13) is true if the candidate frame contains an optimal frame. The corollary is true. □

Corollary 1 allows us to improve the grid-based algorithm using a BnB like approach. We check if the condition in Equation (3.13) is satisfied. If not, we know that the optimal frame is not contained in the candidate frame. Hence we can delete the frames that are contained in the candidate frame. Say that the candidate frame is $c' = [x', y', z']$, then the frames contained in c' define a subset of solution space, which is $\Phi_{c'}$.

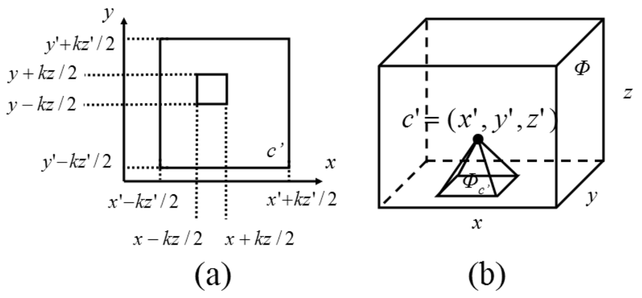


Figure 3.4: *An illustration of the solution space formed by frames that contained in given frame c'. The constant k is determined by the camera aspect ratio. For a aspect ratio of 4:3, $k = 3$.*

As illustrated in Figure 3.4(a), if a frame (x, y, z) is contained in c', it has to satisfy the following set of conditions,

$$\begin{aligned} x - kz/2 &\geq x' - kz'/2 \\ x + kz/2 &\leq x' + kz'/2 \\ y - kz/2 &\geq y' - kz'/2 \\ y + kz/2 &\leq y' + kz'/2 \\ (x, y, z) &\in \Phi \end{aligned} \tag{3.14}$$

Therefore, $\Phi_{c'} = \{(x, y, z) | (x, y, z) \text{ satisfies Equation (3.14)}\}$. Recall that the solution space Φ is a 3D rectangle, Figure 3.4(b) illustrates that the shape of $\Phi_{c'}$ is a pyramid within the 3D rectangle and has its top located at the c'.

The volume of the pyramid is determined by its height, z' of the candidate frame. Bigger z' means bigger candidate frame, which leads to a bigger cut in Φ if the candidate frame does not satisfy Corollary 1. This suggests that we should follow a descending order in z axis when evaluating the lattice points.

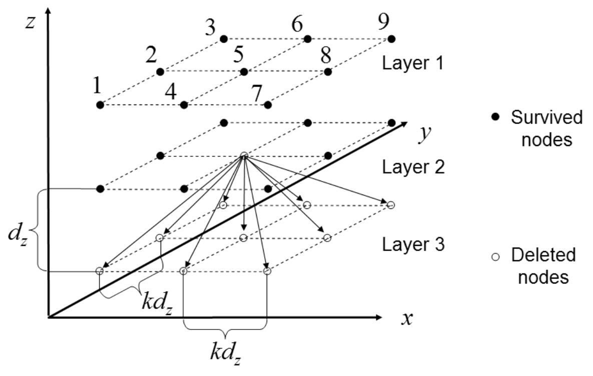


Figure 3.5: *An illustration of the BnB like approach.*

Figure 3.5 illustrates how to perform the BnB like search using a $3 \times 3 \times 3$ lattice. We divide lattice points into different layers with respect to their z values. The search starts with topmost layer and follows a descending order in z. In this lattice, we set $d = 3d_z$, which will be used as the default settings in the rest of section.

In each layer, we evaluate the objective function at each lattice point following lexicographic order (i.e. the numbered sequence in layer 1 of Figure 3.5). After the evaluation, we test if the point satisfies the condition in Corollary 1. If so, we name this point as a survived node. Otherwise, this is a deleted node. If a node is deleted, it will cause some nodes in next layer to be deleted also because of the shape of the pyramid. We name those nodes in next layer as the children of the deleted node. Since we choose $d = kd_z$ (k=3 for camera 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 the 9 children in the next layer (frames with zoom at $z - d_z$),

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

should also be deleted. The union area of 9 children covers the same area as the frame (x, y, z) does.

Lemma 3 can be proved by checking if all 9 children are located inside the frame (x, y, z) and their union region is identical to that of frame (x, y, z). Figure 3.5 also illustrates this relationship. The central node in layer 2 is deleted and causes all 9 children to be deleted. If the deleted node is a boundary node, the number of children is less than 9.

Combining the information above, we can reduce the computation effort required and present the Bnb like approach. Recall that we need to evaluate the lattice point in a descending order in z and follow a lexicographic order in xy plane. Recall that $\hat{c}$ is the currently best known solution. Initially, we set $\hat{c}$ to be any arbitrary feasible frame and every node in the lattice to be survived node.

BnB like approach

```
for node $(x, y, z)$
    if the node was deleted
        if not the lowest layer
            delete its childern in lower layer
        end if
    else
        compute $s(x, y, z)$                              $O(n)$
        if equation 3.13 holds
            if $s(x, y, z) > s(\hat{c})$
                $\hat{c} = (x, y, z)$
            end if
        else
            if not the lowest layer
                delete its childern in lower layer
            end if
        end if
    end if
end for
```

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 the result section.

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

In this sub section, we improve the running time of the algorithm by reducing the problem to Functional Box Sum (FBS) query and hence can be solved more efficiently. We begin with the problem reduction.

Define the weight function

$$\omega_i(z) = (1/a_i)\min(\frac{z_i}{z}, 1). \tag{3.15}$$

Then Equation (4.1) can be rewritten as,

$$s_i(z) = \omega_i(z) Area(r_i \cap c).$$

Therefore, for a fixed zoom level, computing the objective function value for a given candidate frame can be viewed as a box aggregation problem over a group of requested viewing zones. As shown in Equation (4.1), the query result should be a weighted sum of intersected areas between the query window (i.e. the candidate frame) and requested viewing zones.

When the requested viewing zones are rectangles, this special box aggregation problem can be solved by the functional box sum query introduced by Zhang, Tsotras, and Gunopulos in [122]. 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 objective 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 δ can be reduced to 2^δ 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^δ 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 $\delta = 2$ in our case, Figure 3.6 from [122] shows how to convert an FBS query into 4 OIFBS queries.

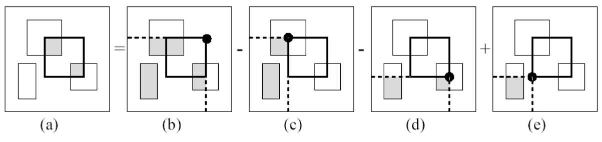


Figure 3.6: *Convert an FBS query to 4 OIFBS queries in [122].*

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 [122] 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 viewing zones 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 [10], 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 $z$                                  ($g/d_z$ in total)
   a. Preprocess requested viewing zones into an
         ECDF-B$^q$-tree                                    $O(n \log^2 n)$
   b. For each $(x, y)$ pair                                ($wh/d^2$ in total)
         Do an FBS query with the frame $(x, y, z)$
            to find $s(x, y, z)$.                           $O(\log^2 n)$
ii. Report the frame $(x, y, z)$ with the
      largest FBS query value.                              $O(1)$
```

This will yield an algorithm with running time of $O((g/d_z)(n + wh/d^2) \log^2 n)$. 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 (4.1), 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 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 $z$
                                                        $O(n \log^2 n)$
ii. For each zoom level $z$ in an ascended order
                                                        ($g/d_z$ 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 $(x, y)$                                ($wh/d^2$ in total)
         Do an FBS query with the frame $(x, y, z)$
             to find $s(x, y, z)$.                      $O(\log^2 n)$
iii. Report the frame (x,y,z) with the
       largest FBS query value.                         $O(1)$
```

Since there are n requested view zones, step (**) should cost $O(n \log^2 n)$ in total. Other steps will cost $O((n + (g/d_z)wh/d^2) \log^2 n)$ in total. Set $d = kd_z$ and $d_z = \frac{1}{2}(\frac{\epsilon}{1-\epsilon})\underline{z}$, the following theorem is true.

Theorem 3. Using an ECDF-B^q-tree, we can solve the Frame Selection problem with rectangular requests in $O((n + 1/\epsilon^3) \log^2 n)$ for a given approximation bound ϵ.

3.3.4 Distributed algorithm

In the Co-Opticon 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 $[\underline{z}, \bar{z}]$ is the zoom range, define

$$o_i = (i-1)d_z + \underline{z}$$

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

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

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

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

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

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

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 3.10 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

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

Hence, $\epsilon \leq \frac{3d_z}{\underline{z}+3d_z}$ □

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 5 shows that the approximation bound change is very small. We can also extend the proof to cases where more than one user drop out.

3.3.5 Numerical Experiment

We have implemented the algorithms in section 3.3.1, 3.3.4, and 3.3.2 on a PC laptop with 1.6Ghz Intel Centrino CPU and 512MB RAM. The algorithm is programmed in Microsoft Visual C++.

Random inputs are generated for testing, which are generated in two steps. First, we generate four random points, which are uniformly distributed in Φ. The four points represent locations of interests, which are referred as seeds. For each seed, we use a random number to generate a radius of interest. Then we generate requested regions in the second step. We generate a requested region using four random numbers. One of them is used to determine which seed the request will be associated with. two of them will be used to generate the location of the request , which are located within the corresponding radius of the associated seed. The remaining random number is used to generate resolution for the request.

Figure 3.7 shows the algorithm's result for a random example with 16 requested frames.

We compare the approximation algorithm to the exact algorithms in Chapter 4. Figure 3.8 illustrates the speed comparison between the approximation algorithm and the exact algorithm. The implementation is the basic grid based algorithm in section 3.3.1 that does not use the BnB like approach. In this test, we set $w = h = 500$, $\underline{z} = 40$, $\bar{z} = 80$, and

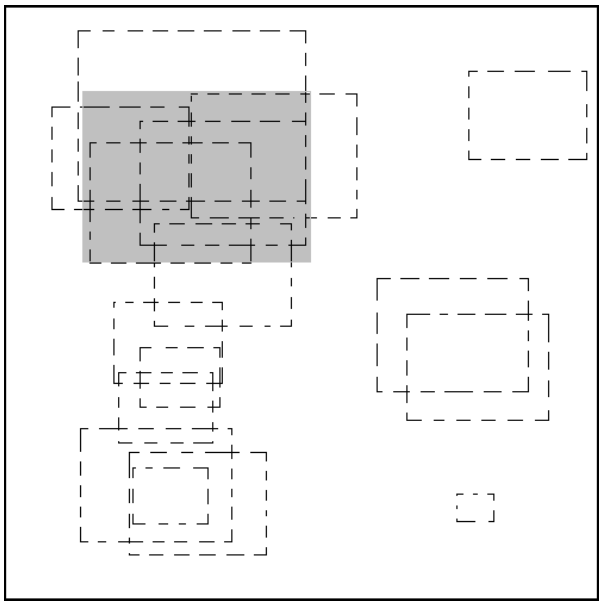


Figure 3.7: *Sample output of algorithm with 16 requested frames and $d_z = 2$. The shaded frame is optimal.*

$k = 3$. The result is conformal to our analysis. The computation time for approximation time is linear with respect to the number of requests with its slope determined by the approximation bound.

We have also tested the effectiveness of the BnB like approach in section 3.3.2 using the same settings. We define

$$\gamma = \frac{\text{Computation Time using BnB}}{\text{Computation Time using Basic Grid}}$$

as the performance index variable. Again, we use randomly generate requested viewing zones. Each data point in Figure 3.9 is an average of 5 iterations using different requested viewing zones. Figure 3.9 shows that γ is not sensitive to user number n but is very sensitive to the approximation bound ϵ. If ϵ is smaller, γ gets smaller, too. This is desirable because smaller ϵ means finer grid with more computation. If the number of the requested

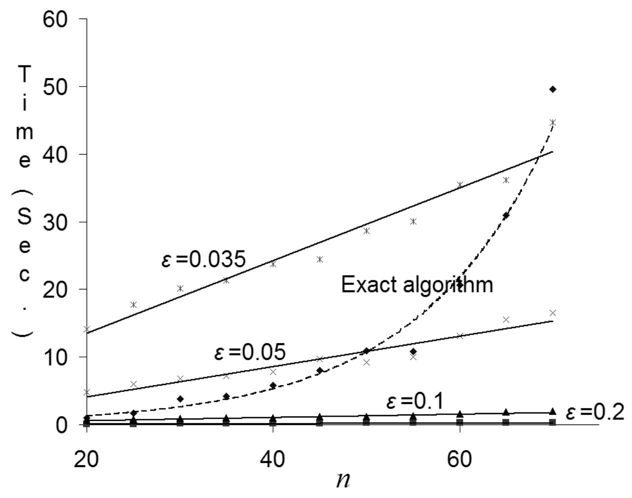


Figure 3.8: *Speed comparison between approximation algorithm and exact algorithm in Chapter 4.*

viewing zones is small (i.e $n = 5$), the overhead cost of BnB like approach dominates the computation, the desirable trend does not appear until the ϵ is small enough. It worths mentioning that the effectiveness of the BnB like approach also depends on the $\underline{z}$. If $\underline{z} = 0$, the BnB like approach fails to cut the computation time. Fortunately, $\underline{z} = 0$ means the satellite can see things infinitely small, which will not happen. In our settings, the BnB like approach can speed up the computation at least by three time.

Figure 3.10 shows how solution quality decreases as network clients fail, based on the example in figure 3.7. This supports the analysis in the previous section: the approximation algorithm degrades gracefully with client failures.

3.4 Conclusions

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

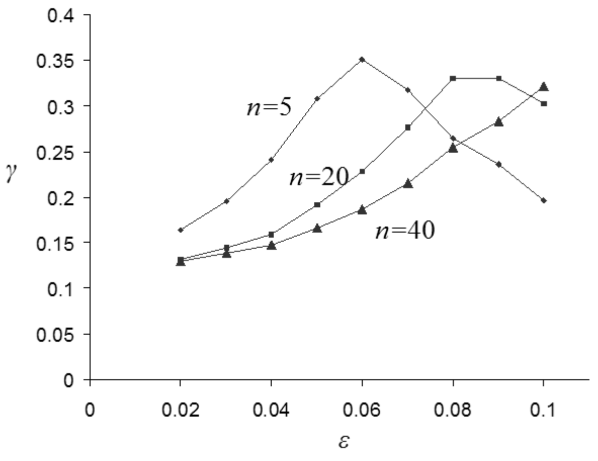


Figure 3.9: *Efficiency of the BnB like approach.*

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 our application, speed is a more important issue than accuracy because the responsiveness of the web camera is a critical performance index. However, there are other applications of Frame Selection problem, where an exact solution may be desirable. In Chapter 4, we present the exact algorithms with a more general metric for Frame Selection problem.

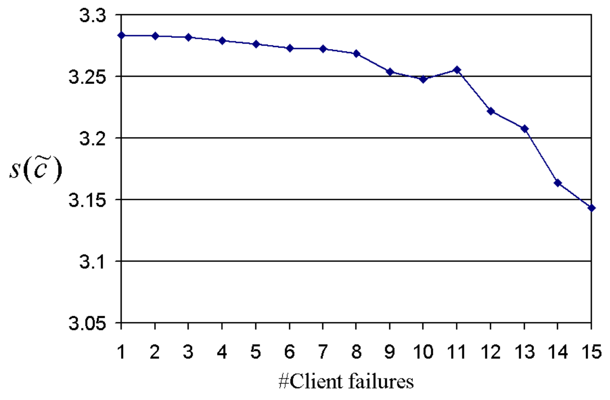


Figure 3.10: *Performance of the Distributed Algorithm: solution quality as network clients fail. Each data point is an average of 10 runs with random dropouts. Initially there are 16 clients and 16 requested frames as shown in the previous figure. Here we plot solution quality as more and more clients 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.*

Chapter 4

Exact Algorithms for Automated Satellite Frame Selection

4.1 Introduction

The first commercially-available high-resolution optical satellite, IKONOS, was launched in 1999 [27]. Since then, satellite imaging has developed into a rapidly growing industry. According to the data from the Imaging and Geospatial Information Society [121], the market is $2.44 billion in 2001 and growing at a rate of fifteen percent annually. Clients include weather prediction, search and rescue, disaster recovery, journalism, and government. Commercial satellites are equipped with sophisticated cameras, which allow them to take high-resolution images as they fly over the Earth. Commercial cameras offer pan, tilt, and zoom (image resolution) control. Near Real Time (NRT) Imaging refers to freshly captured images that are delivered as quickly as possible, depending on the satellite's

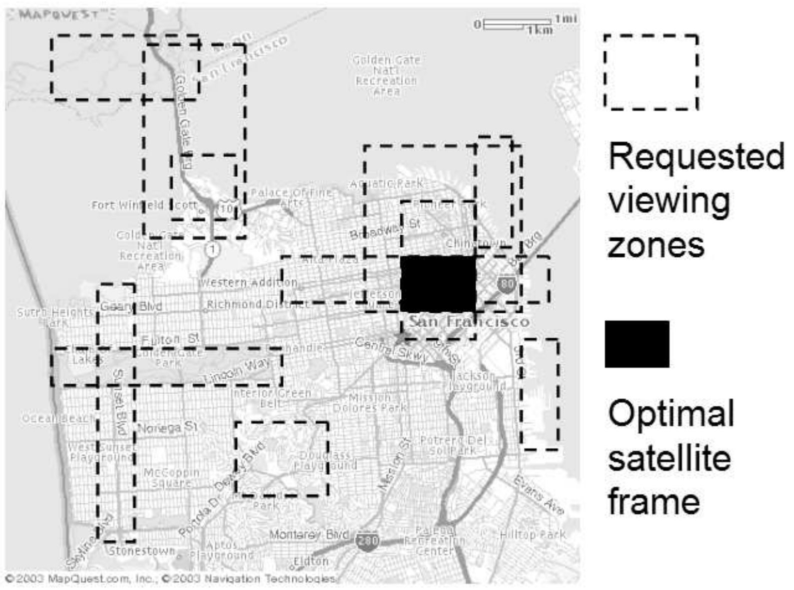


Figure 4.1: *The Satellite Frame Selection (SFS) problem: each time window defines the camera's possible field of view. Client requests for images are shown as dashed rectangles. Given a set of requests, the objective is to compute the satellite frame that optimizes the coverage-resolution metric. The solution in this case is illustrated with a solid rectangle.*

trajectory: at any given time, the camera's field of view is restricted to a zone on the Earth's surface. During each time window, a number of client requests for images are pending, and only one image can be captured. We consider the problem of automatically selecting pan, tilt, and zoom parameters to capture images that maximize reward.

The Satellite Frame Selection problem is illustrated in Figure 4.1. The camera image frame is a rectangle with a fixed aspect ratio. Input is the set of n iso-oriented requested rectangular regions from users. We propose a reward metric based on how closely a requested viewing zone compares with a candidate satellite image frame. The metric is proportional to the intersection of the candidate frame and the requested viewing zone and to the ratio of the resolution of the candidate and the request. Satellite Frame Selection is a non-linear optimization problem; we exploit the structure of the problem to develop polynomial time algorithms that compute the exact solution for cameras with discrete and continuous resolution levels.

4.2 Related work

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

The Satellite Space Mission problem (SM) [54] is to select and schedule a set of jobs on a satellite. Each candidate job has fixed duration, available time window, and weight. The goal is to select a feasible sequence of that jobs maximizes the sum of weights. This combinatorial optimization problem is known to be NP-hard. Recent research [29, 37, 59, 117] on the SM problem and its variations focuses on developing exact and approximate methods using numerical methods such as column generation, Tabu search, and genetic algorithms.

Lemaitre et al. [73] study a related problem for the Earth Observing Satellite (EOS), which has a three-axis robotic camera that can be steered during each time window. Given a set of requested zones, they consider the problem of finding a trajectory for the camera that will maximally cover the requested zones (they do not consider variations in zoom/resolution). Their coverage problem is analogous to planning optimal routes for lawn mowers and vacuum cleaners [38]. Researchers have proposed greedy algorithms, dynamic programming algorithms, and methods based on constraint programming and Local Search. In our model, the time window is shorter and the objective is to servo the camera to a single optimal position with optimal zoom/resolution setting.

The structure of the SFS problem is related to the planar $p-$center problem, which Megiddo and Supowit [86] showed to be NP-complete. Given a set of point demand centers on the plane, the goal is to optimally locate p service centers that will minimize the worst case travel distance between client and server. Using a geometric approach, Eppstein [32] found an algorithm for the the planar 2-Center problem in $O(n \log^2 n)$. Halperin et al. [55] gave an algorithm for the 2-center problem with m obstacles that runs in randomized expected time $O(m \log^2(mn) + mn \log^2 n \log(mn))$.

The SFS problem is also related to "box aggregation" querying in spatial database research [122]. The spatial objects could be points, intervals, or rectangles. Aggregation over points is a special case of the orthogonal range search queries from computational geometry. Agarwal and Erickson [1] provide a review of geometric range searching and related topics. Grossi and Italiano [52, 53] proposed the cross-tree data structure, a generalized version of a balanced tree, to speed up range search queries in high-dimensional space. The continuity of the solution space of our problem makes it impossible to simply evaluate a fixed set of candidate frames through queries.

In the multimedia literature, Kimber and Liu et al. describe a multi-user robot camera for videoconferencing [71, 76]. They formulate frame selection for multiple simultaneous requests as an optimization problem based on position and area of overlap. To solve it, they propose an approximation based on comparing the bounding box of all combinations of user frames. The main concern of their algorithm is speed rather than accuracy. Although they did not provide bounds on their approximation, their approach is sufficient for videoconferencing applications.

Our lab at Berkeley is studying collaborative teleoperation systems where many users share control of a single physical resource. Inputs from each user must be combined to generate a single control stream for the robot. In the taxonomy proposed by Chong et al. [21], these are Multiple Operator Single Robot (MOSR) systems. An Internet-based MOSR system is described by McDonald, Cannon, and colleagues [19, 85]. In their work, several users assist in waste cleanup using Point-and-Direct (PAD) commands. Users point to cleanup locations in a shared image and a robot excavates each location in turn. More recent developments on MOSR systems can be found in [44, 51].

The SFS problem is closely related to controlling a shared robotic webcam. We introduced the frame selection problem for robotic webcams in a series of conference chapters: exact solution with discrete zoom [112], approximation solution with continuous zoom [110, 111], approximate solution with fixed zoom [58]. This journal paper presents greatly expanded, generalized versions of those chapter, extending to image requests of any aspect ratio and introducing new reward metric.

4.3 Problem definition

In this section we formalize the SFS problem using a reward metric based on satellite image resolution.

4.3.1 Input and assumptions

The camera on a typical satellite orbits the Earth at a speed of more than 7km per second. As illustrated in Figure 4.2, a reflection mirror moves under pitch and roll control,

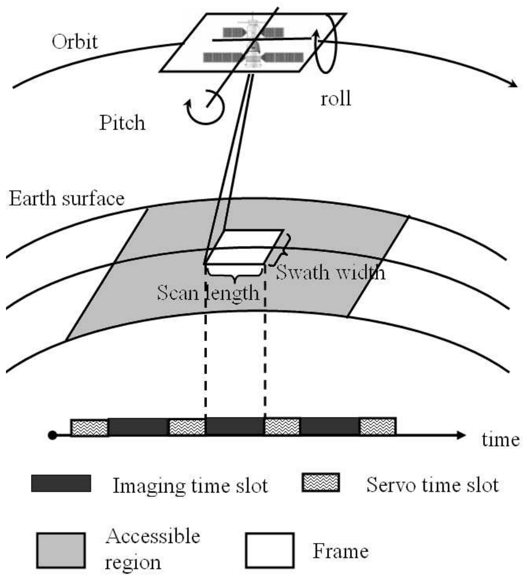


Figure 4.2: *Satellite with accessible region, and frame definition.*

which directs a rectangular region of the earth into the camera's field of view. As illustrated in the figure, the satellite's orbit is partitioned into time windows. During each window, the satellite camera can capture one rectangular image which we refer to as a frame. Since most satellites cannot perform yaw rotation, the frame is always aligned with satellite orbit direction.

We assume that the camera frame is a rectangle with fixed aspect ratio (4:3), ie its width is proportional to the resolution. A triple $c = [x, y, z]$ describes such a frame: $[x, y] \in R_a$ specifies the center point of the frame with respect to an accessible region R_a, and z specifies the resolution of the frame. The pair x, y determines the pitch and roll angles of the mirror. Setting $z = 10$ meters means a pixel in the image is equivalent to area of 10×10 square meters. A higher z-value means lower image resolution. The attainable resolution set is Z, so $z \in Z$. For example, a frame has a width that is 1000 times the resolution z and a length that is 1333 times the resolution z, then the area of the frame is $1000*1333\times z^2$. The width and the length of the frame are linear functions of the resolution, which are defined as $w(z)$ and $l(z)$ respectively.

For a given time window, we consider n requested viewing zones. The i^{th} request, $0 \leq i \leq n$, is a rectangle $r_i = [x_i, y_i, w_i, l_i, z_i, u_i]$, where $[x_i, y_i] \in R_a$ specifies center point with respect to the accessible region, w_i, l_i are the width and the length of the requested rectangle, z_i is the desired resolution, and u_i is the utility for the request, which describes how much the client is willing to pay for the requested view zone. This is also the maximum reward associated with this request. We assume that all requested viewing zones are iso-oriented rectangles (not necessarily with 4:3 aspect ratio) aligned with the satellite's orbit direction.

Given a set of n requested viewing zones, the objective is to compute a single frame c^* that yields maximum total reward. The solution space is

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

We consider two cases: 1) Z is a finite discrete set and 2) Z is a continuous set.

4.3.2 Reward Metric

Recall that r_i is the i^{th} requested viewing zone. The corresponding client has a utility u_i for this region. Define s_i as the reward from the i^{th} request. Let $c = [x, y, z]$ be a candidate camera frame. If the r_i is fully covered by c, i.e., $r_i \subseteq c$, and the desired resolution is obtained, i.e., $z_i \geq z$, then $s_i = u_i$. If the resolution requirement is satisfied but the coverage is partial, then the reward is discounted by a coverage ratio: $s_i = u_i \frac{Area(r_i \cap c)}{Area(r_i)}$. If

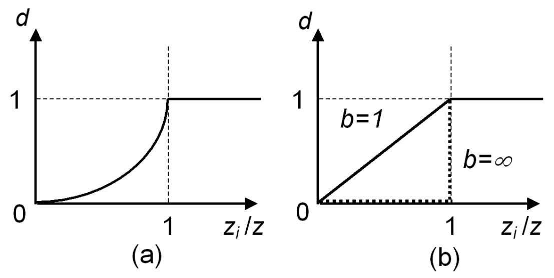


Figure 4.3: Resolution discount function.

$z_i < z$ (the resolution requirement is not satisfied) then the reward should be discounted by a resolution discount factor $d(z, z_i)$. Hence, $s_i = u_i \frac{Area(r_i \cap c)}{Area(r_i)} d(z, z_i)$. As illustrated in Figure 4.3(a), the resolution discount function $d(z, z_i)$ is a truncated function: $0 \leq d(z, z_i) \leq 1$. It is an increasing function of z_i/z because an image has more value as resolution increases. The resolution discount function we propose is

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

Let $Resolution(r_i) = z_i$ and $Resolution(c) = z$. We call this reward metric the Coverage-Resolution Ratio (CRR),

$$s_i(c) = u_i \frac{Area(r_i \cap c)}{Area(r_i)} \min\Big(\big(\frac{Resolution(r_i)}{Resolution(c)} \big)^b, 1 \Big) \tag{4.1}$$

The exponential discount factor b determines how resolution affects reward. Figure 4.3(b) shows two cases: $b = 1$ and $b = \infty$. The case is $b = \infty$ corresponds to a scenario in which users will not accept any image with a resolution that is lower than requested. We use the case $b = 1$ as default setting for numerical examples in the rest of the chapter.

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

$$s(c) = \sum_{i=1}^{n} s_i(r_i, c). \tag{4.2}$$

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

4.3.3 Properties of the CRR reward metric

The Coverage-Resolution Ratio s is non-smooth and piecewise linear in both x and y. For convenience we use $s(x, y, z)$ instead of $s(c)$ with $c = [x, y, z]$.

Nonsmoothness

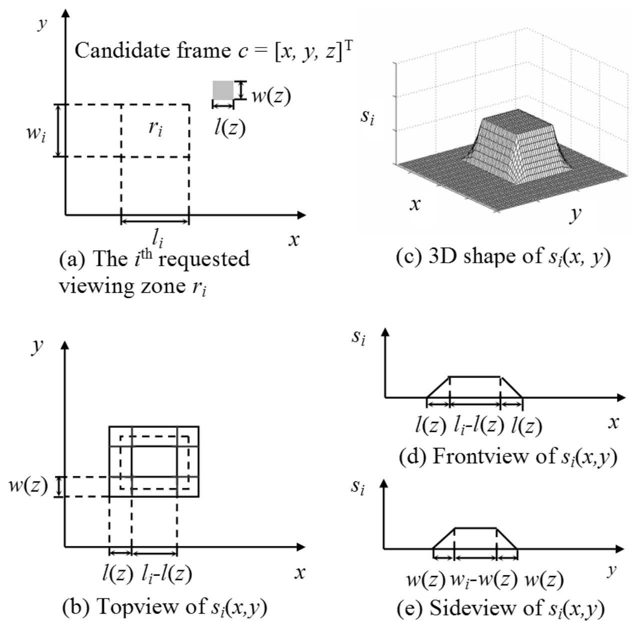


Figure 4.4: *The CRR reward function, $s_i(x, y)$, for a given candidate frame. Assume $z_i > z$, $l(z) \leq l_i$, and $w(z) \leq w_i$, we can move the candidate frame (gray rectangle) around the the r_i to observe how $s_i(x, y)$ changes. The function is plateau-like with a maximum height of $u_i Area(c \cap r_i)/Area(r_i)$. The function consists of 5 planar and 4 quadratic surfaces at the corners.*

Recall that $Area(r_i) = w_i l_i$, $z = Resolution(c)$, and $z_i = Resolution(r_i)$. To study the properties of the reward function, we first treat z as a constant: $Area(c \cap r_i) = p_i(x, y)$ is a function of (x, y). The objective function defined by Equation (4.1) becomes a function of the center point of the candidate frame,

$$s(x,y) = \sum_{i=1}^{n} \omega_i p_i(x,y) \tag{4.3}$$

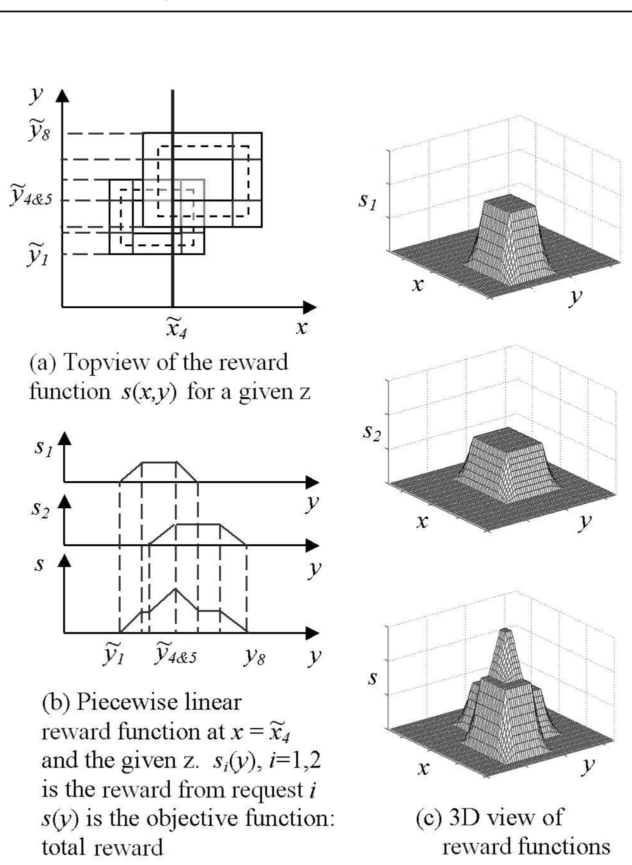
where

$$\omega_i = \frac{u_i}{w_i l_i} \min((z_i/z)^b, 1) \tag{4.4}$$

is a constant for each user. We know that $p_i(x,y)$ is the area of the intersection of the i^{th} requested viewing zone 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 4.4 shows the shape of $s_i(x,y)$ given $z < z_i$ and candidate frame c smaller than r_i. Note that s_i is non-differentiable with respect to x and y so we cannot use derivative-based approaches to solve this problem.

Piecewise linearity in x and y

Since all requested viewing zones and the candidate frame are iso-oriented rectangles, the shape of any intersection between them is also a rectangle with its edges parallel to either x axis or y axis. Thus the term $p_i(x,y)$ in Equation (4.3) is 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 reward function $s(x,y)$ is a linear combination of $p_i(x,y)$, $i = 1,...,n$, it has the same property. Figure 4.5 shows an example for a case with two requested viewing zones.



y
ỹ8
ỹ4&5
ỹ1
x̃4
x
(a) Topview of the reward function s(x,y) for a given z
s1
s2
s
y
ỹ1
ỹ4&5
y8
(b) Piecewise linear reward function at x = x̃4 and the given z. si(y), i=1,2 is the reward from request i s(y) is the objective function: total reward
(c) 3D view of reward functions

Figure 4.5: *The combined reward function $s(y)$ 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.*

4.3.4 Comparison with "similarity metrics"

Symmetric Difference (SD) and Intersection Over Union (IOU) are standard "similarity metrics" used in pattern recognition as a measure of how similar two shapes are [17, 15, 118]. In our case, for a requested viewing zone r_i and a candidate frame c, the SD metric would be:

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

The intersection-over-union metric would be

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

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

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

The key differences between CRR and these metrics are:

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

4.4 Algorithms

In this section we present algorithms for two versions of the Satellite Frame Selection problem. We start with a version in which the resolution is restricted to a set of discrete values. Subsection 4.4.2 describes an algorithm for this version of the problem. In Subsection 4.4.3 we allow the resolution to vary continuously. The algorithms make use of geometric events we call “plateau vertices.”

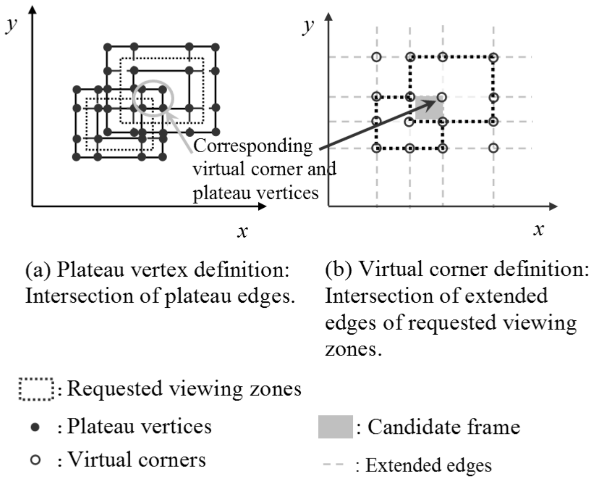

y
x
Corresponding virtual corner and plateau vertices
y
x
(a) Plateau vertex definition: Intersection of plateau edges.
(b) Virtual corner definition: Intersection of extended edges of requested viewing zones.
: Requested viewing zones
● : Plateau vertices
○ : Virtual corners
: Candidate frame
: Extended edges

Figure 4.6: *Illustration of relationship between "plateau vertices" and "virtual corners" with geometric interpretation for two requested viewing zones and a candidate frame with fixed resolution. Each candidate frame with its center located at a "plateau vertex" must have one its corner overlapped with a "virtual corner".*

4.4.1 Plateau vertices and virtual corners

Recall that the objective function for a given resolution and one requested viewing zone $s_i(x, y)$ is plateau-like as shown in Figure 4.4. The function consists of nine facets: one top plane, four side planes, and four quadratic surfaces at the corners. There are two vertical boundaries and two horizontal boundaries at the bottom (bounding the entire plateau), the same numbers of similar edges at the top (bounding the plateau's flat top), and eight boundaries separating side planes and corner quadratic surfaces (see Figure 4.6(a)).

For a fixed resolution z and n requested viewing zones, there are n plateaus. We define a *plateau vertex* as an intersection between any two boundaries, which includes both intersections of facet boundaries induced by a single plateau or by two distinct plateaus. Since all plateaus are iso-oriented, one of the boundaries is horizonal and the other is vertical. A plateau vertex can be represented by a three-dimensional vector $(\tilde{x}, \tilde{y}, z)$. For n

requested viewing zones and m fixed resolutions, there are $O(mn^2)$ plateau vertices. Figure 4.6(a) show an example of plateau vertices for two requested viewing zones.

As illustrated in Figure 4.6(b), we define a *virtual corner* as an intersection of two extended edges of original rectangular shaped requested viewing zones. For n requested viewing zones, there are $O(n^2)$ virtual corners as well. Since the definition of virtual corner does not depend on resolution, each virtual corner can be represented by a two-dimensional vector in (x, y) plane. Figure 4.6 shows the relationship between plateau vertices and virtual corners, which is also described by the following lemma,

Lemma 4 (Virtual Corner/Plateau Vertex Relation). A candidate frame with its center located at a plateau vertex must have one corner overlapping a virtual corner.

Proof. Recall that a plateau vertex is an intersection of one vertical plateau edge and one horizontal plateau edge. The plateau edge corresponds to the non-smooth part of the reward function, which occurs when an edge of the candidate frame overlaps with an extended edge of a requested viewing zone. If a candidate frame is centered at a plateau vertex then its edges must overlap with two orthogonal extended edges of some requested viewing zones, which forms a virtual corner by definition. □

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

- The notion of plateau vertex is only applicable to cases where the resolution of the candidate frame is discrete and finite; the notion of virtual corner applies to cases where the resolution is either discrete or continuous.

- Not all virtual corners have corresponding plateau vertices. For example, the top left virtual corner in Figure 4.6(b) has no corresponding plateau vertex. In fact, a virtual corner that does not overlap with any REAL edge of a requested viewing zone does not have a corresponding plateau vertex.

- For m discrete resolutions, a virtual corner has $O(m)$ corresponding plateau vertices. A virtual corner has at most four corresponding plateau vertices for a fixed resolution. The position of the virtual corner is invariant to resolution by definition.

Plateau vertices and virtual corners can help us to find the optimal solution for the optimization problem defined in Equation (4.2) for the discrete resolution case and the continuous resolution case respectively.

Lemma 5 (Virtual Corner Optimality Condition). At least one optimal frame has one corner overlapping with a virtual corner.

Proof. Let $c^* = [x^*, y^*, z^*]$ be an optimal solution. If we fix $z = z^*$, we get $s(x, y)$ as a summation of plateaus. As discussed earlier, for a fixed z and x, the objective function $s(y)$ is piecewise linear. So the optimum must be at a vertex $y = \tilde{y}$ such that $s(x^*, \tilde{y}, z^*) = s(x^*, y^*, z^*)$. We also know that the line $y = \tilde{y}$ in the (x, y) plane is one of the horizontal facet boundaries of the plateaus. Similarly, we can find another optimal frame $[\tilde{x}, \tilde{y}, z^*]$, where line $x = \tilde{x}$ is one of the vertical facet boundaries of the plateaus. Therefore, the optimal frame $[\tilde{x}, \tilde{y}, z^*]$ is centered at a plateau vertex $(\tilde{x}, \tilde{y})$ for a fixed resolution $z = z^*$. Apply Lemma 4, we know that the optimal frame $[\tilde{x}, \tilde{y}, z^*]$ must have one corner at one of the virtual corners. □

Using the Virtual Corner Optimality Condition (VCOC), we can restrict frames to share a corner with a virtual corner, thereby reduce the dimensionality of the problem. The VCOC is true no matter whether the resolution variable is discrete or continuous. However, it is more convenient to use plateau vertices when the resolution variable is discrete. The VCOC can be transformed to the following Plateau Vertex Optimality Condition,

Lemma 6 (Plateau Vertex Optimality Condition). When the resolution variable is discrete, at least one optimal frame is overlapping with a plateau vertex.

Proof. From the proof the Lemma 5, we know that we can find an equivalent optimal solution $[\tilde{x}, \tilde{y}, z^*]$ from a given optima solution $c^* = [x^*, y^*, z^*]$. We also know that $[\tilde{x}, \tilde{y}]$ are intersection of two facet boundaries. For the discrete resolution case, z^* has to be one of the discrete resolutions in the solution space. Then the point $[\tilde{x}, \tilde{y}, z^*]$ is one of the plateau vertices. □

4.4.2 Algorithms for discrete resolutions

A satellite camera may have a discrete set of m resolution levels. We can use this algorithm to find the best position and resolution parameters.

Brute force approach.

Based on the Lemma 6, we can solve the optimization problem by simply checking all combinations of resolution levels and corresponding plateau vertices. We evaluate the objective function for each of the $O(n^2)$ plateau vertices and repeat this for each of the m resolution levels. It takes $O(n)$ time to evaluate a candidate frame c. Therefore, the brute force algorithm runs in $O(n^3m)$.

Efficient traversal of plateau vertices.

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

Plateau Vertex Traversal (PVT) Algorithm

```
$s^* = 0$,                                                                  $O(1)$
Sort $\{y_i + .5w_i\}$, $i = 1, ...n$                                       $O(n \log n)$
Sort $\{y_i - .5w_i\}$, $i = 1, ...n$                                       $O(n \log n)$
For each resolution level $z$                    ($m$ in total)
  (i). /* Compute vertical extended plateau edges $\{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_{4n}\}$ */     $O(n)$
    For each requested viewing zone $i$, $i = 1, ..., n$,
      $\tilde{x}_{4i-3} = x_i - .5(w_i + w(z))$, $\tilde{x}_{4i-2} = x_i - .5(w_i - w(z))$,
      $\tilde{x}_{4i-1} = x_i + .5(w_i - w(z))$, $\tilde{x}_{4i} = x_i + .5(w_i + w(z))$,
    End For
  (ii). /* Compute the sorted sequence $\{\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_{4n}\}$*/,          $O(n)$
    For each $i$, $i = 1, ..., n$
      $\tilde{y}_{4i-3} = y_i - .5(w_i + w(z))$, $\tilde{y}_{4i-2} = y_i - .5(w_i - w(z))$,
      $\tilde{y}_{4i-1} = y_i + .5(w_i - w(z))$, $\tilde{y}_{4i} = y_i + .5(w_i + w(z))$,
    End For
    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). /*Solve 1D problems */
    For $x = \tilde{x}_i$, $i = 1, ..., 4n$,
      $s = \max_y \; s(\tilde{x}_i, y, z)$,
      if $s > s^*$ then $s^* = s, x^* = \tilde{x}_i$, $y^* = y, z^* = z$.
    End For
End For
Output $s^*$ as optimal objective function value and $(x^*, y^*, z^*)$ as optimal frame.
```

In step (iii) of the PVT algorithm, we traverse the vertical facet boundaries of the plateaus one by one. Figure 4.7 illustrates how it works using the example of two requested viewing zones. For each vertical edge, we find the maximum. Using Lemma 5, 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

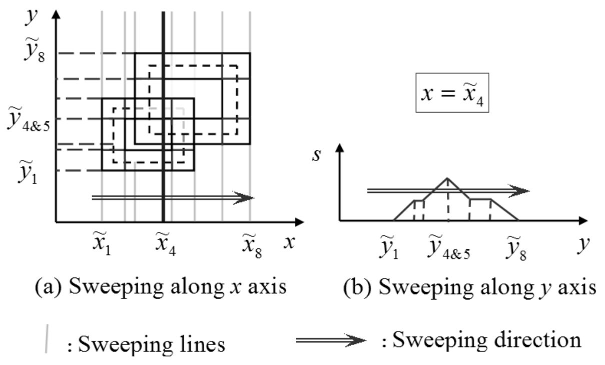
$$\max_{y} \ s(x, y, z)$$

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 4.5. 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 $O(n^2)$, proving the following theorem.

Theorem 6. We can solve the Satellite Frame Selection problem in time $O(n^2m)$ for n users and m resolution levels.

4.4.3 Algorithms for continuous resolution

If a satellite camera has a variable and continuous resolution, there are infinitely many "plateau vertices". The virtual corner optimality condition can reduce the 3D optimization problem to $O(n^2)$ 1D optimization problems with respect to variable z. We then show that each 1D optimization problem can be dissected into $O(n)$ piecewise polynomial



y
$\tilde{y}_8$
$\tilde{y}_{4\&5}$
$\tilde{y}_1$
$\tilde{x}_1$ $\tilde{x}_4$ $\tilde{x}_8$ x
(a) Sweeping along x axis
$x = \tilde{x}_4$
s
$\tilde{y}_1$ $\tilde{y}_{4\&5}$ $\tilde{y}_8$ y
(b) Sweeping along y axis
: Sweeping lines
: Sweeping direction

Figure 4.7: *An illustration of PVT algorithm using the example in figure 4.5. Figure (a) shows how we sweep along x axis to dissect the 2D optimization problem into $O(n)$ 1D optimization problems. Figure (b) shows how we solve the 1D optimization problem by traversal over the ordered vertices for $x = \tilde{x}_4$.*

functions, for each, we can find an optimal in time $O(n)$. Using incremental computation and a diagonal sweep, we show how to improve the running time to $O(n^3)$.

Basic Virtual Corner Algorithm (BVC)

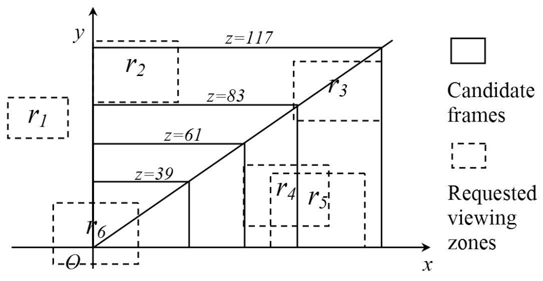


Figure 4.8: *An example of the 1D optimization problem with respect to z. In this example, we assume $l(z) = 4z$, $w(z) = 3z$, $b = 1$, and $u_i = 1$ for $i = 1, ..., n$.*

For n requested viewing zones, there are $O(n^2)$ virtual corners. The virtual corner optimality condition (Lemma 5) allows us to find the optimal frame by checking the candidate frames that have one of their corners overlapped with one of the virtual corners. This means that we can reduce the original 3D optimization problem in Equation (4.2) to

$O(n^2)$ 1D optimization problems. Define $p_i(z) = Area(r_i \cap c)$, $a_i = Area(r_i) = w_i l_i$, the 1D optimization problem is to find,

$$\max_z s(z) = \sum_{i=1}^{n} u_i(p_i(z)/a_i) \min((z_i/z)^b, 1) \tag{4.5}$$

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

To study the 1D maximization problem in Equation (4.5), consider a virtual corner. For simplicity, we assume that the virtual corner is at the origin. Moreover, we assume that the virtual corner coincides with the lower left corner of the candidate frame. (The virtual corner in Figure 4.8 is the intersection of the extensions of the left edge of r_2 and the bottom edge of r_5.) Placements in which one of the other three corners of the candidate frame coincides with the virtual corner are handled in a similar fashion. We may be able to eliminate some of the placements beforehand, but it reduces the computation by only a constant factor. Now, we gradually increase z and observe the value of $s(z)$: Figure 4.9 shows the function for the example in Figure 4.8.

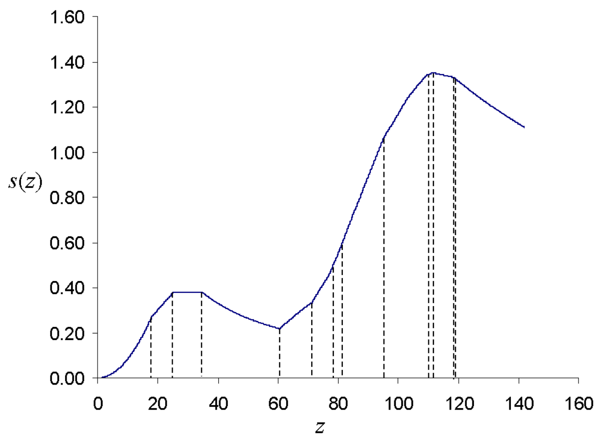


Figure 4.9: *Reward function for the example in figure 4.8 as a function of image resolution.*

a. Critical z Values and Intersection Topologies. The function $s(z)$ is a piecewise smooth function (see Figure 4.9), so derivative-based approaches cannot be used directly. We refer to a maximal z-interval on which $s(z)$ is smooth as a segment. We consider four questions that form the basis for our algorithms.

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

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

We start with question 1).

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

Let $Z_c(x_v, y_v)$ be the set of critical z values for virtual corner (x_v, y_v). From Equation (4.8), we see that the non-smoothness comes from the non-smoothness of either $\min((z_i/z)^b, 1)$ or $p_i(z)$. The critical z values that come from the former type form a subset $Z_c'(x_v, y_v)$, those of the latter type a subset $Z_c''(x_v, y_v)$. The former type is easy to deal with because it occurs at $z = z_i$, $i = 1, ..., n$. Therefore, $Z_c'(x_v, y_v) = \{z_i|\text{i=1,...,n}\}$, so $|Z_c'(x_v, y_v)| = n$. Note that $Z_c'(x_v, y_v)$ is the same for all virtual corners (x_v, y_v), so

$Z'_c(x_v, y_v) = Z'_c$.

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

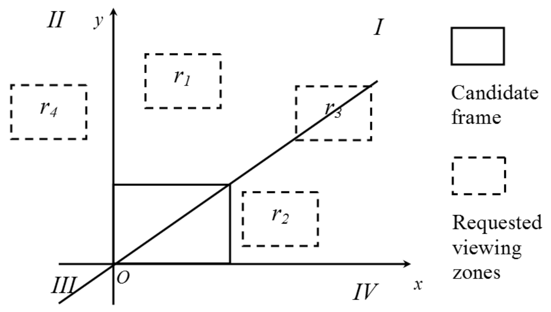


Figure 4.10: *Examples for "fundamental rectangles". In this figure, r_1 and r_2 are type (a) rectangles, r_3 is a type (b) rectangle, and r_4 is a type (o) rectangle.*

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

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

Figure 4.10 gives examples for these three classes of fundamental rectangles.

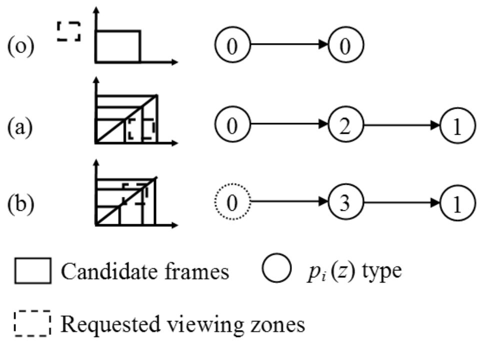


Figure 4.11: *Change of $p_i(z)$ for the three classes of requested viewing zones when z gradually increases from 0^+ to $+\infty$.*

As shown in Figure 4.11, as z increases,

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

The transitions correspond to critical z values.

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

b. Requested viewing zone decomposition. A requested viewing zone that is not a fundamental rectangle intersects at least one of following: the positive x-axis, the positive y-axis, and the extended diagonal of the expanding candidate frame. We treat the different intersection patterns and show that in each case the requested viewing zone can be decomposed into at most four fundamental rectangles (see also Figure 4.12).

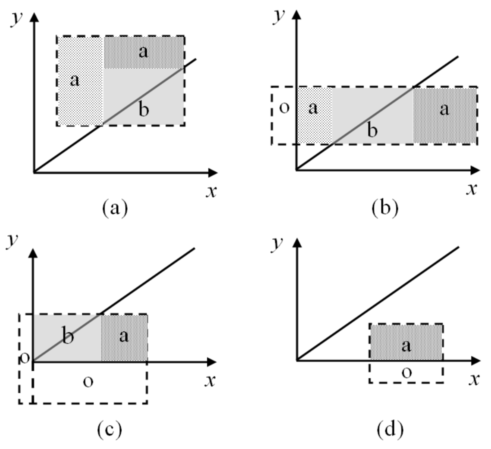


Figure 4.12: *Examples of four requested viewing zone decomposition cases.*

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

axes, then it can be decomposed into one class (a) rectangle, one class (b) rectangle, and two class (o) rectangles.

As we can see from figure 4.12, a decomposed requested viewing zone can yield at most three fundamental rectangles that are either class (a) or class (b). Every fundamental rectangle inherits the z_i value of the original request.

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

Lemma 7. For each virtual corner, the z-axis can be partitioned into $O(n)$ segments, on which $s(z)$ is smooth.

Lemma 7 answers our question 2) from the previous section.

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

Lemma 8. For each segment, $s(z)$ is a polynomial function with 6 coefficients g_0, g_1, g_2, g_3, g_4, and g_5,

$$s(z) = g_0 z^{-b} + g_1 z^{-b+1} + g_2 z^{-b+2} + g_3 + g_4 z + g_5 z^2. \tag{4.6}$$

Proof. For a virtual corner (x_v, y_v), let us assume the segment is defined by $[z', z'')$, where $z', z'' \in Z_c(x_v, y_v)$ are two adjacent critical z values. The n requested viewing zones have been classified and decomposed into $k = O(n)$ class (a) or (b) rectangles. We denote those rectangles as $\tilde{r}_i$, $i = 1, ..., k$. Let us define set $S' = \{i | z_i \leq z'\}$ and set $S'' = \{i | z_i \geq z''\}$. From the definition of critical z value, we know that $z_i \notin (z', z'')$ for $i = 1, ...n$ so that $S' \cup S'' = \{1, ..., k\}$ and $S' \cap S'' = \emptyset$. Therefore, Equation (4.5) becomes,

$$s(z) = \sum_{i \in S''} u_i p_i(z)/a_i + \sum_{i \in S'} u_i (p_i(z)/a_i)(z_i/z)^b \tag{4.7}$$

We also define S_j be the set of rectangles with type j intersection areas when $z \in [z', z'')$, for $j = 1, 2, 3$ respectively. Recall that $a_i = w_i l_i$ is a constant; we have

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

We can perform a similar transform for the second term of Equation (4.7).

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

Combining them, we get Equation (4.6). □

The proof of Lemma 8 shows that Equation (4.5) can be converted into Equation (4.6) in $O(n)$ time. The maximum of Equation (4.6) can be found in constant time.

Combining Lemma 7 and Lemma 8 yields the Basic Virtual Corner Algorithm.

Basic Virtual Corner (BVC) Algorithm

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

Theorem 7. The Basic Virtual Corner algorithm (BVC) solves the problem in $O(n^4)$ time.

Virtual Corner with Incremental Computing (VC-IC)

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

Virtual Corner with Incremental Computing (VC-IC)

```
Sort members of $Z_c(x_v, y_v)$                              $O(n \log n)$

Compute first polynomial coefficients                        $O(n)$

For each subsequent segment                                  $O(n)$

    Update polynomial coefficients                           $O(1)$

    Find maximum for the polynomial                          $O(1)$

End For
```

The VC-IC algorithm improves the running time:

Theorem 8. The Virtual Corner with Incremental Computing (VC-IC) algorithm solves the problem in $O(n^3 \log n)$.

Virtual Corner with Incremental Computing and Diagonal Sweeping (VC-IC-DS)

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

Each critical z value in $Z''_c(x_v, y_v)$ uniquely defines the position of the upper right corner of the candidate frame on its extended diagonal, which is called critical point in the figure 4.13(a). Each critical point corresponds to the point that the candidate frame start intersecting some requested viewing zone or the point that the intersection between the candidate frame and some requested viewing zone ends. This gives a geometric interpretation for those critical z values. Figure 4.13(a) shows a case with two requested viewing zones and five critical z values.

Let $Z_e''(x_v, y_v)$ be the set of the corresponding z values of the intersections between the extended diagonal and the extended edges, which is illustrated in Figure 4.13(b). $Z_e''(x_v, y_v)$ also depends on virtual corner (x_v, y_v). As shown in Figure 4.13(a) and Figure 4.13(b),

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

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

Figure 4.13(c) illustrates a nice property of the sorted sequence of points in $Z_e''(x_v, y_v)$. In the figure, we have an ordered sequence of intersected points at the extended diagonal that starts from the origin O. we number the point closest to the origin as point 1 and the second closest as point 2. As we gradually move the extended diagonal downward and observe what happens to the sorted sequence, we find that the order of the sorted sequence does not change until the diagonal line hits an intersection between two extended edges, which is a virtual corner by definition. Let us define this virtual corner be the adjacent virtual corner to the virtual corner at the origin. Point 1 and point 2 switch their order at the adjacent virtual corner (i.e. the gray rectangle in the figure 4.13(c)). This phenomenon shows that if we have a sorted sequence of the intersection points at a virtual corner, we can get the sorted sequence at an "adjacent virtual corner" in constant time.

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

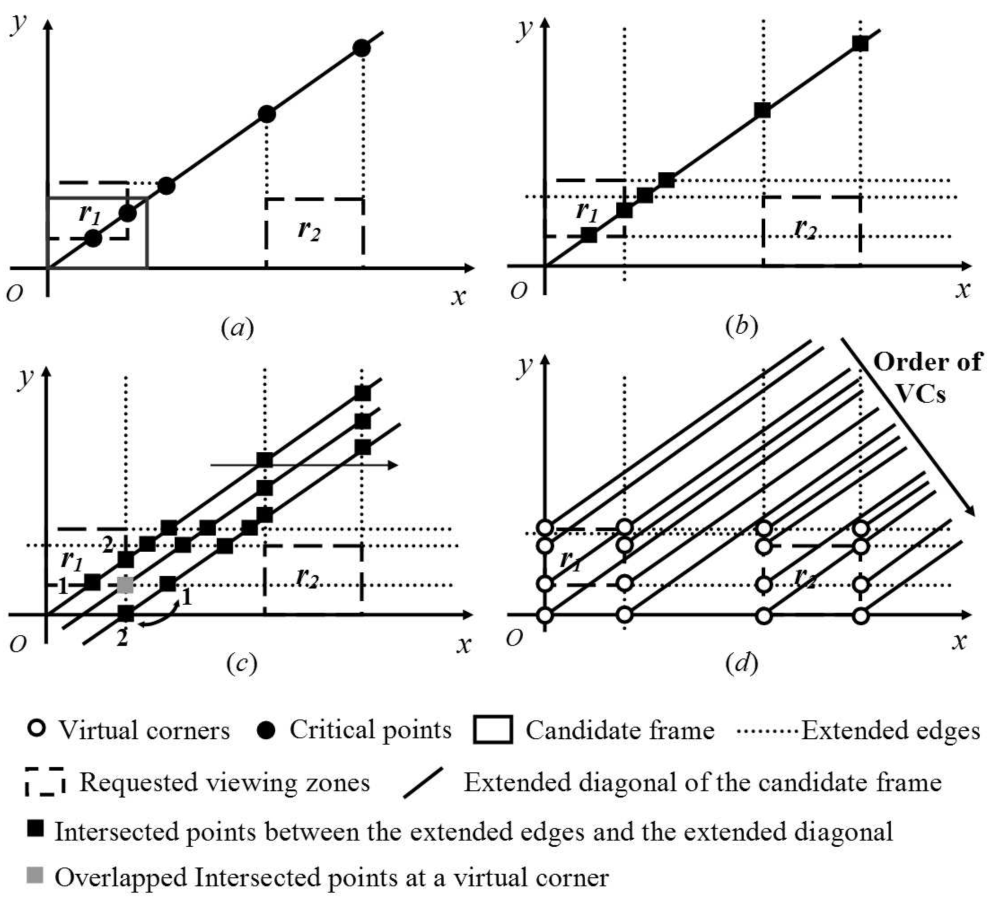

y
x
O
r1
r2
(a)
(b)
(c)
(d)
1
2
Order of VCs
○ Virtual corners
● Critical points
Candidate frame
Extended edges
Requested viewing zones
Extended diagonal of the candidate frame
■ Intersected points between the extended edges and the extended diagonal
■ Overlapped Intersected points at a virtual corner

Figure 4.13: *(a) $Z_c''(x_v, y_v)$ for a two requested viewing zone case, (b) $Z_e''(x_v, y_v)$ are set of intersection points between the extended diagonal of the candidate frame and the extended edges, (c) The two intersection points switch order only at a virtual corner formulated by the intersection of the two extended edges that generate the two intersection points, and (d) Sorting virtual corners in this order can reduce the sorting cost in the algorithm.*

the extended diagonal and an intercept at $+\infty$, we decrease the intercept and stop at each virtual corner. As shown in figure 4.13(d), we solve the sub problem for the virtual corner when the sweeping line stops. This yields the following VC-IC-DS algorithm.

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

```
Sort $Z'_c$                                                  $O(n \log n)$
Sort virtual corners in sweeping order                       $O(n^2 \log n)$
Sort $Z''_e(x_v, y_v)$ for the first virtual corner          $O(n \log n)$
For each virtual corner $(x_v, y_v)$                         $O(n^2)$
    Update ordered set $Z''_e(x_v, y_v)$                     $O(1)$
    Get members of $Z''_c(x_v, y_v)$                         $O(n)$
    Merge $Z'_c$ and $Z''_c(x_v, y_v)$                       $O(n)$
    Run the sub routine in section 4.4.3.                    $O(n)$
End For
Report the maximum $s(c)$ and the corresponding $c^*$.
```

Theorem 9. The Virtual Corner with Incremental Computing and Diagonal Sweeping (VC-IC-DS) approach solves the problem in $O(n^3)$ time.

4.5 Results

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

Figure 4.14 shows the results for four different sets of inputs. As we can see from Figure 4.14(a) and (b), the optimal frame does not necessarily have one corner overlapping

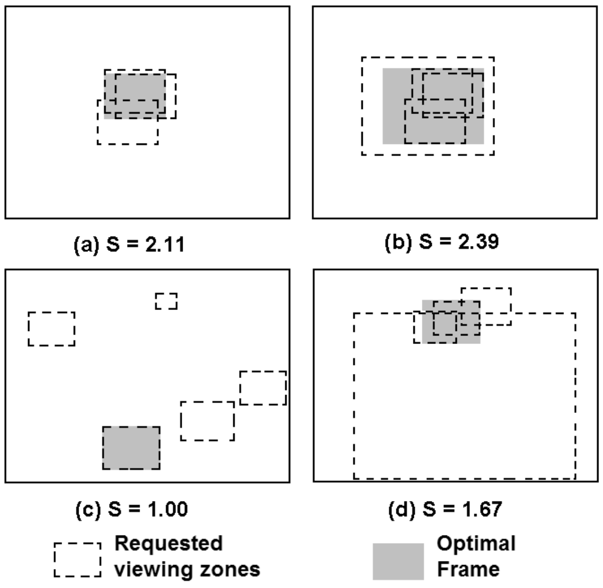


Figure 4.14: *Examples of computed optimal frames (shown in grey). We set* $b = 1$ *and* $u_i = 1$ *for all requests and use PVT Algorithm from section 4.4.2. We have* 10 *different resolution levels and set* $l(z) = 4z$ *and* $w(z) = 3z$.

with a corner of a requested viewing zone. However, one corner of the optimal frame does overlap with one of the virtual corners. Figure 4.14(b) has three requested viewing zones exactly the same as those in (a) and one more big requested viewing zone. It is interesting to see how the optimal frame changes after the big requested viewing zone joined in the system. Figure 4.14(c) shows that if all input rectangles fall far way from each other, the algorithm functions as a chooser selects one input rectangle with the highest utility value as the output. Figure 4.14(d) shows that a large requested viewing zone does not necessarily yield a large optimal frame. It worth mentioning that results depend on utility u_i, which functions as weight on each requested viewing zone. Those samples only illustrate cases that utility are same across or requested viewing zones.

Figure 4.15 shows a speed comparison for the two algorithms presented in Sub-

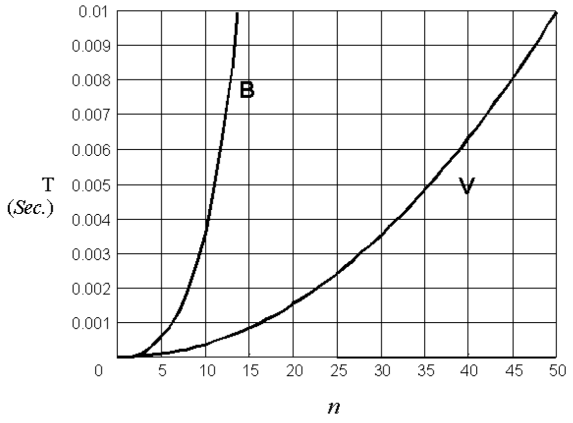


Figure 4.15: Speed comparison for the two algorithms from Subsection 4.4.2. Curve B refers to the brute-force algorithm; curve V refers to the PVT algorithm. Each data point is based on average of 10 runs with random inputs.

section 4.4.2 for a fixed resolution level ($m = 1$). It confirms what the theoretical analysis predicts; the plateau vertex traversal algorithm clearly outperforms the brute-force approach. We use random inputs for testing. The random inputs are generated in two steps. First, we generate four random points, which are uniformly distributed in R_a. The four points represent locations of interests, which are referred as seeds. For each seed, we use a random number to generate a radius of interest. Then we generate requested viewing zones. To generate a requested viewing zone, we create six random numbers. One of them is used to determine which seed the request will be associated with. Two of them will be used to generate the location of the center point of the request, which is located within the corresponding radius of the associated seed. The remaining three random numbers are used to generate width, length, and resolution for the request. In the test, we set utility to be 1 across all inputs. Each data point in Figure 4.15 is the average of ten runs.

Figure 4.16 shows the relationship between the optimal frame size and the choice of b value in the Coverage-Resolution Ratio. This demonstrates the tradeoff: large b leads

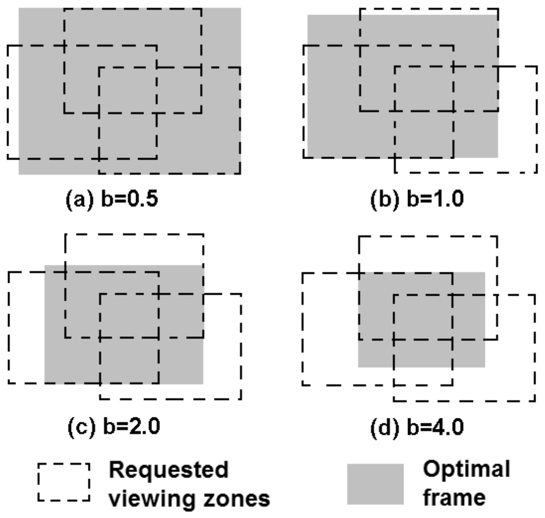


Figure 4.16: *Relationship between the optimal frame size and the choice of the b value in Coverage-Resolution Ratio metric in Subsection 4.3.2. Algorithm chosen and its settings are the same as the samples in figure 4.14.*

to large ouput frames. As $b \to 0^+$, the optimal frame becomes the smallest frame that contains all request viewing zones: $Area(c \cap r_i) = Area(r_i)\ \forall i$.

Figure 4.17 illustrates the speed difference between BVC, VC-IC, and VC-IC-DS algorithms. Each data point in Figure 4.17 is an average of 10 trials with different random inputs, where the same random inputs are used to test all three algorithms. The timing results are consistent with the theoretical analysis.

4.6 Conclusions and Future Work

To automate satellite camera control, this chapter introduces the Satellite Frame Selection problem: find the satellite camera frame parameters that maximize reward during each time window. We formalize the SFS problem based on a new reward metric that incorporates both image resolution and coverage. For a set of n client requests and a

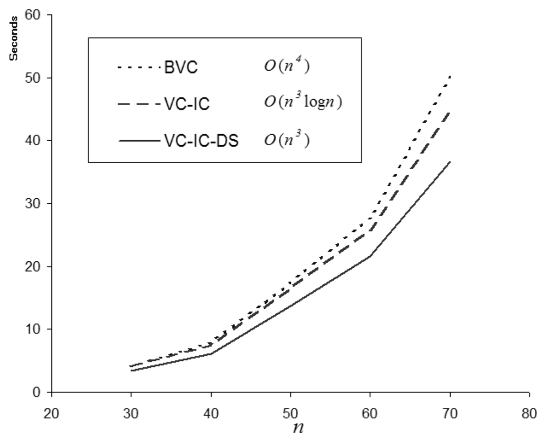


Figure 4.17: *Computation speed comparison for random inputs.*

satellite with m discrete resolution levels, we give an SFS algorithm that computes optimal frame parameters in $O(n^2m)$. For satellites with continuously variable resolution ($m = \infty$), we give an SFS algorithm that computes optimal frame parameters in $O(n^3)$. We have implemented all algorithms and compare computation speeds on randomized input sets.

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

In last three chapters, we have studied the Collaborative Teleoperation system, where the shared device is a Pan-Tilt-Zoom robotic camera. In Chapter 5, we introduce a different Collaborative Teleoperation system, where the shared resource is a human "Tele-Actor".

Chapter 5

The Tele-Actor System: Collaborative Teleoperation Using Networked Spatial Dynamic Voting

5.1 Introduction

Consider the following scenario: an instructor wants to take a class of students to visit a research lab, semiconductor plant, or archaeological site. Due to safety, security, and liability concerns, it isn't possible to arrange a class visit. Showing a pre-recorded video does not provide the excitement nor group dynamics of the live experience. In this chapter we describe a system that allows groups to collectively visit remote sites using client-server networks. Such "collaborative teleoperation" systems may be used for applications in education, journalism, and entertainment.

Remote-controlled machines and teleoperated robots have a long history [108]. Networks such as the Internet provide low-cost and widely-available interfaces that makes such resources accessible to the public. In almost all existing teleoperation systems, a single human remotely controls a single machine. We consider systems where a group of humans shares control of a single machine. In a taxonomy proposed by Tanie et al. [21], these are Multiple Operator Single Robot (MOSR) systems, in contrast to conventional Single Operator Single Robot (SOSR) systems.

In MOSR systems, inputs from many participants are combined to generate a single control stream. There can be benefits to collaboration: teamwork is a key element in education at all levels [103, 23, 22] and the group may be more reliable than a single (possibly malicious) participant [44].

As an alternative to a mobile robot, which can present problems in terms of mobility, dexterity, and power consumption, we propose the "Tele-Actor", a skilled human with cameras and microphones connected to a wireless digital network, who moves through the remote environment based on live feedback from online users.

We have implemented several versions of the system. Figure 5.1 shows a view of the "Spatial Dynamic Voting" (SDV) interface implemented for Internet browsers. Users are represented by "votels": square colored markers that are positioned by each user with a mouse click. This chapter presents system architecture, interface, and collaboration metrics

.

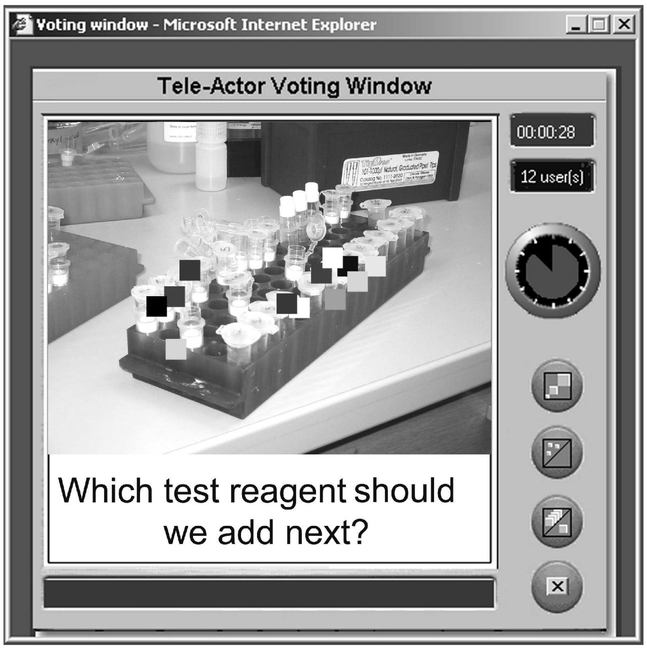

Voting window - Microsoft Internet Explorer
Tele-Actor Voting Window
00:00:28
12 user(s)
Which test reagent should we add next?

Figure 5.1: *The Spatial Dynamic Voting (SDV) interface as viewed by each user. In the remote environment, the Tele-Actor takes images with a digital camera which are transmitted over the network and displayed to all participants with a relevant question. With a mouseclick, each user places a color-coded marker (a "votel" or voting element) on the image. Users view the position of all votels and can change their votel positions based on the group's response. Votel positions are then processed to identify a "consensus region" in the voting image that is sent back to the Tele-Actor. In this manner, the group collaborates to guide the actions of the Tele-Actor.*

5.2 Related Work

Tanie, Matsuhira, Chong, et al. [21] proposed a useful taxonomy for teleoperation systems: Single Operator Single Robot (SOSR), Single Operator Multiple Robot (SOMR), and Multiple Operator Multiple Robot (MOMR).

Most networked robots are SOSR, where control is limited to one human operator at a time. Tanie et al. analyzed an MOMR system where each operator controls one robot arm and the robot arms have overlapping workspaces. They show that predictive displays and scaled rate control are effective in reducing pick-and-place task completion times that require cooperation from multiple arms [21].

In an MOMR project by Elhajj, Fukuda, Liu, Xi, and colleagues [30, 31], two remote human operators collaborate to achieve a shared goal such as maintaining a given force on an object held at one end by a mobile robot and by a multi-jointed robot at the other. The operators, distant from the robots and from each other, each control a different robot via force-feedback devices connected to the Internet. The authors show both theoretically and experimentally that event-based control allows the system to maintain stable synchronization between operators despite variable time-lag on the Internet.

MOMR models are also relevant to online collaborative games such as *Quake* and *The Sims Online*, where players remotely control individual avatars in a shared virtual environment.

In SOMR systems, one human operator controls multiple robots. A variant is No Operator Multiple Robot (NOMR) systems, sometimes called Collaborative or Cooperative Robotics, where groups of autonomous robots interact to solve an objective [3]. Recent results are reported in [28, 102, 99, 18].

A number of SOSR systems have been designed to facilitate remote interaction. Paulos and Canny's Personal Roving Presence (ProP) telerobots, built on blimp or wheeled platforms, were designed to facilitate remote social interaction with a single remote operator [95, 96]. Hamel [56] studied how networked robots can be useful in hazardous environments. Fong, Thorpe and colleagues study SOSR systems where collaboration occurs between a single operator and a mobile robot that is treated as a peer to the human and modeled as a noisy information source [36]. Related examples of SOSR "cobots" are analyzed in

[2, 78, 109, 11, 83, 82].

One precedent for an online MOSR system is described in McDonald, Cannon and colleagues [85]. For waste cleanup, several users to assist remotely using Point-and-Direct (PAD) commands [19]. Users point to cleanup locations in a shared image and a robot excavates each location in turn. In this Internet-based MOSR system, collaboration is serial but pipelined, with overlapping plan and execution phases. The authors demonstrate that such collaboration improves overall execution time but do not address conflict resolution between users.

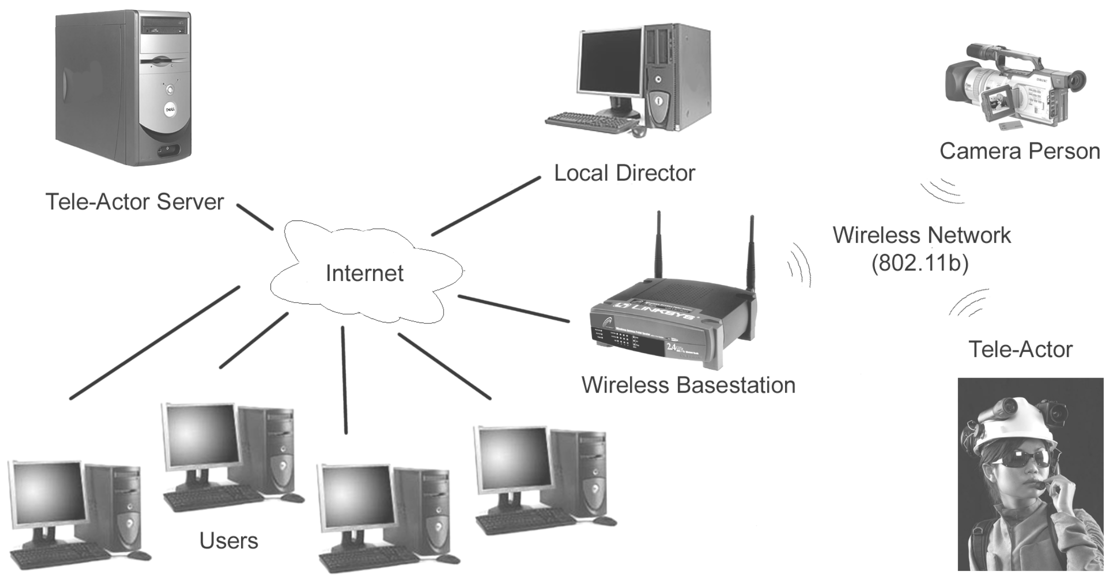


Figure 5.2: *System architecture. Participants on the Internet view and voting on a series of voting images. The human "Tele-Actor," with head-mounted wireless audio/video link, moves through the remote environment in response. The "Local Director" facilitates interaction by posting textual queries.*

Pirjanian studies how reliable robot behavior can be produced from an ensemble of independent processors [98]. Drawing on research in fault-tolerant software [74], Pirjanian considers systems with a number of homogenous processors sharing a common objective.

He considers a variety of voting schemes and shows that fault-tolerant *behavior fusion* can be optimized using plurality voting [13] but does not consider spatial voting models such as ours.

In [45], we present an Internet-based MOSR system that averaged multiple vector inputs to control the position of an industrial robot arm. We report experiments with maze-following that suggested that groups of humans may perform better than individuals in the presence of noise due to central limit effects.

In [44], we used finite automata to model collaborating users in a MOSR system such as Cinematrix, a commercial system [20] that allows large audiences to interact in a theater using plastic paddles. To model such systems, we averaged an ensemble of FA outputs to compute a single stream of incremental steps to control the motion of a point robot moving in the plane. We analyzed system performance with a uniform ensemble of well-behaved deterministic sources and then modeled malfunctioning sources that go silent or generate inverted control signals. We found that performance is surprisingly robust even when a sizeable fraction of sources malfunction.

Outside of robotics, the notion of MOSR is related to a very broad range of group activities including social psychology, voting, economics, market pricing, traffic flows, etc. ACM organizes annual conferences on Computer Supported Collaborative Learning (CSCL) and Computer Supported Cooperative Work (CSCW). Surveys of research in this broader context can be found in [87, 68, 101, 39, 6, 34, 48].

We note that the concept of human-mounted cameras with network connections is not novel: there is extensive literature on "wearable computer" systems [79, 80]. The focus of our research is on collaborative control. A preliminary report on the Tele-Actor system appeared in [50].

5.3 System Architecture

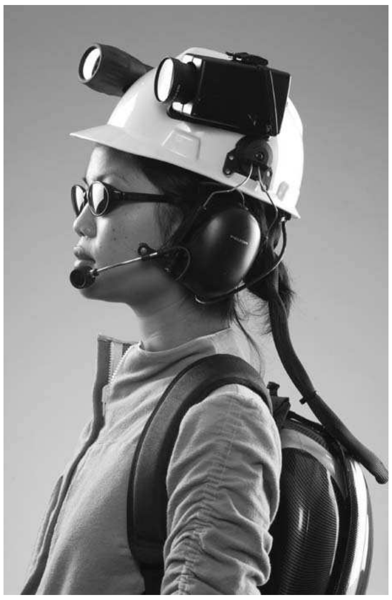


Figure 5.3: *The human Tele-Actor transmits images from the remote environment using the helmet-mounted video camera and responds to user votes. Helmet design by E. Paulos, C. Myers, and M. Fogarty.*

The Tele-Actor system architecture is illustrated in Figure 5.2. As the Tele-Actor moves through the environment, camera images are sent to the Tele-Actor server for distribution to users, who respond from their Internet browsers. User voting responses are collected at the Tele-Actor server, which updates java applets for all users and for the Tele-Actor in the field. The Tele-Actor carries a laptop which communicates to the Internet using the 2.4 GHz 802.11b wireless protocol. A camera-person with a second camera provides third person perspectives as needed. Using this architecture, the users, the Tele-

Actor Server, the Local Director, the camera-person, and the Tele-Actor communicate via the Internet.

5.4 SDV User Interface

We have developed a new graphical interface to facilitates interaction and collaboration among remote users. Figure 5.1 illustrates the “Spatial Dynamic Voting” (SDV) interface that is displayed on the browser of all active voters. Users register online to participate by selecting a votel color and submitting their email address to the Tele-Actor server, which stores this information in our database and sends back a password via email. The server also maintains a tutorial and an FAQ section to familiarize new users with how the systems works.

Using the SDV interface, voters participate in a series of 30-60 second voting images. Each voting image is a single image with a textual question. In the example from Figure 5.1, the Tele-Actor is visiting a biology lab. Voters click on their screens to position their votels. Using the HTTP protocol, these positions are sent back to the Tele-Actor server and appear in an updated voting image sent to all voters every 3-5 seconds. In this way voters can change their votes. When the voting cycle is completed, SDV analysis algorithms analyze the voting pattern to determine a consensus command that is sent to the Tele-Actor. The SDV interface differs from multiple-choice polling because it allows spatially and temporally continuous inputs.

To facilitate user training and asynchronous testing, the Tele-Actor system has two modes. In the offline mode, voting images are drawn from a prestored library. In the online mode, voting images are captured live by the Tele-Actor. Both offline and online modes have potential for collaborative education, testing, and training. In this chapter we focus on the online mode.

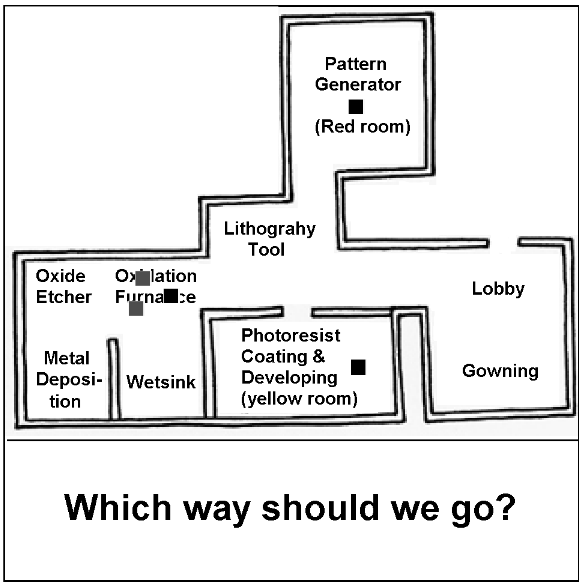


Figure 5.4: *Navigation Query. Participants indicate where they want the Tele-Actor to go.*

Figures 5.4, 5.5, 5.6, and 5.7 illustrate four types of SDV queries and their associated branching structures. In each case, the position of the majority of votels decides the outcome.

We tried including a live video broadcasting stream but found that due to bandwidth limitations, the resolution and frame rate is unacceptable for low-latency applications. Standard video broadcasting software requires about 20 seconds of buffered video data for compression, which introduces unacceptable delays for live visits. We are hoping this can be reduced in the future with faster networks such as Internet2.

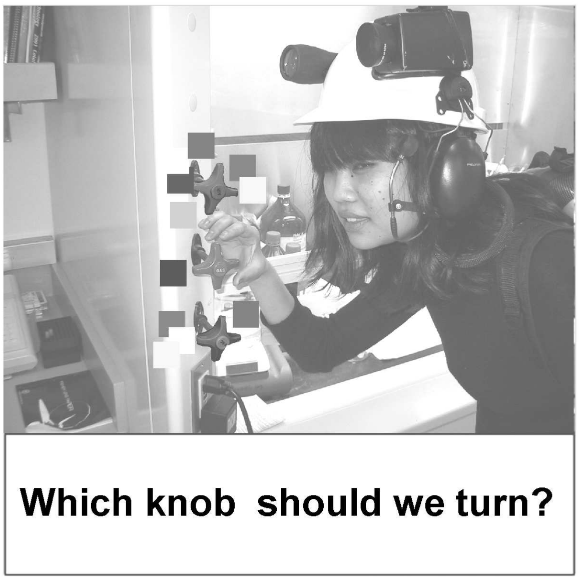


Figure 5.5: *Point Query. Participants point out a region of interest in the voting image.*

5.5 Hardware and Software

5.5.1 Version 3.0 (July 18, 2001)

The Tele-Actor webserver is an AMD K7 950Mhz PC with 1.2GB memory connected to a 100Mbs T3 line. The Local Base Station is a Dell Pentium III 600Mhz laptop with 64MB memory connected to a 10Mbs T1 line at the remote site. It has a USB video card, which captures video at 320×240 resolution.

We used the Swann MicroCam wireless video camera, model ALM-2452 [1]. It is 18x34x20 mm and weighs 20 grams, with a 9 volt battery as its power supply. It has a 2.4 GHz analog RF output at 10 mW and transmits line-of-sight up to 300 feet with a resolution of 380 horizontal lines.

[1]http://www.swann.com.au

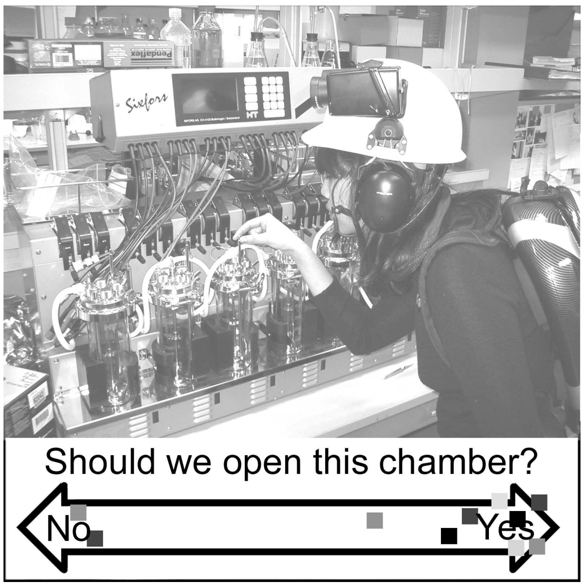


Figure 5.6: *Opinion Query. Votel position can be anywhere between extreme values to indicate degree of opinion.*

Custom software includes:

1. the client side SDV browser interface based on DHTML, (Screen shot is shown in Figure 5.9).

2. the Local Basestation image selection interface, (Screen shot is shown in figure 5.8), and

3. the Tele-Actor server.

During online mode, the Local Basestation, running Microsoft Windows 98, uses a custom C++ application to capture images with textual questions and transmit them to the Tele-Actor server for distribution.

During both online and offline modes, the Tele-Actor server uses custom C and C++ applications to maintain the database and communicate with the local base station

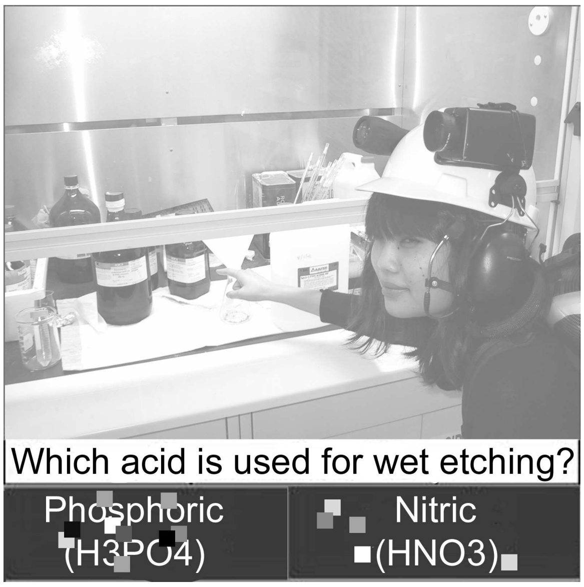


Figure 5.7: *Multiple-choice Query. A variation on the Point Query with a small number of explicit options.*

and with all active voters. The Tele-Actor server runs Redhat Linux 7.1 and the Apache web server 1.3.20. The Resin 2.0.1 Apache plug-in and Sun JDK 1.3.1 with Mysql database 3.23.36 provide java server pages to handle the user registration and data logging [2]. Custom software built on the graphic development toolkit GD 2.0.1 generates election images overlaid with current votel positions.

5.5.2 Version 9.0 (July 25, 2002)

A lot improvements had been done during the year after the lauch of the Tele-Actor syste. Some changes are

1. Scenario design: We found that it is necessary to have some scripts for action to meet the requirement of education. We usually have a few prepared small scenarios before

[2]http://www.caucho.com

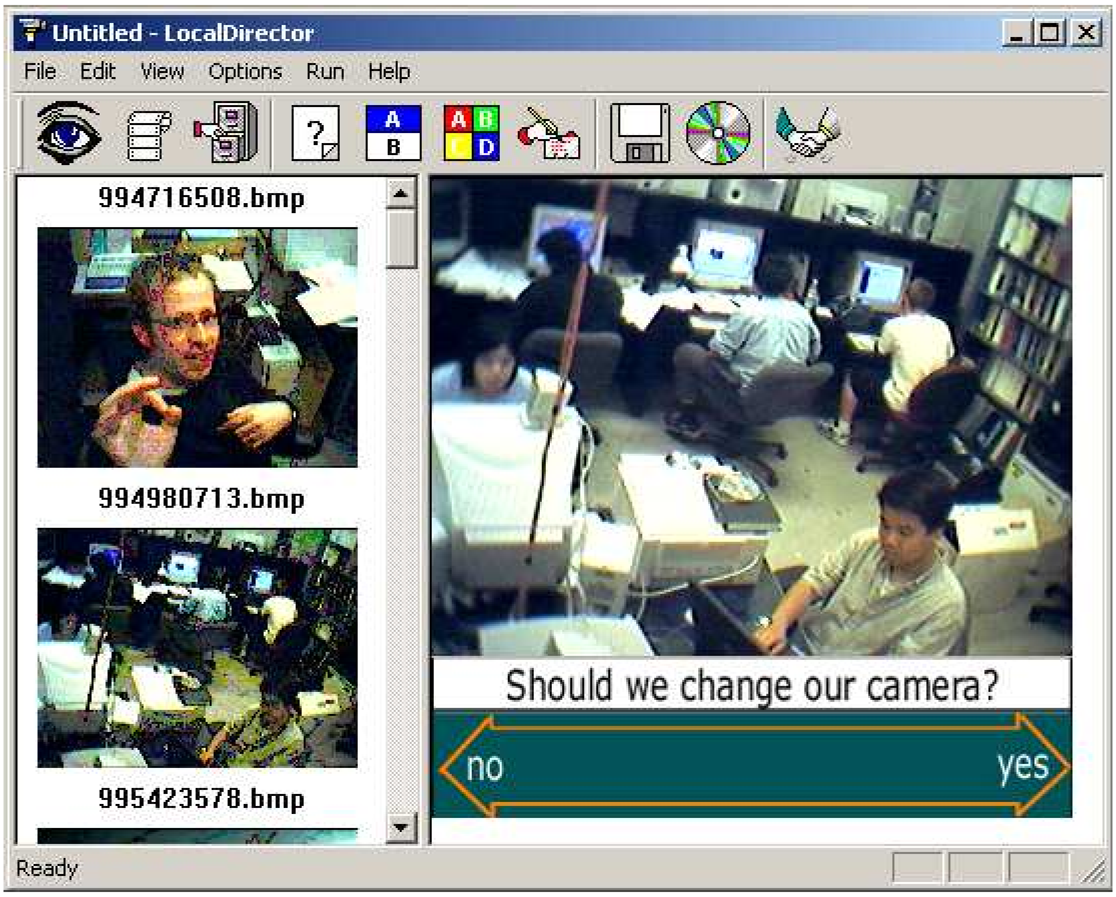
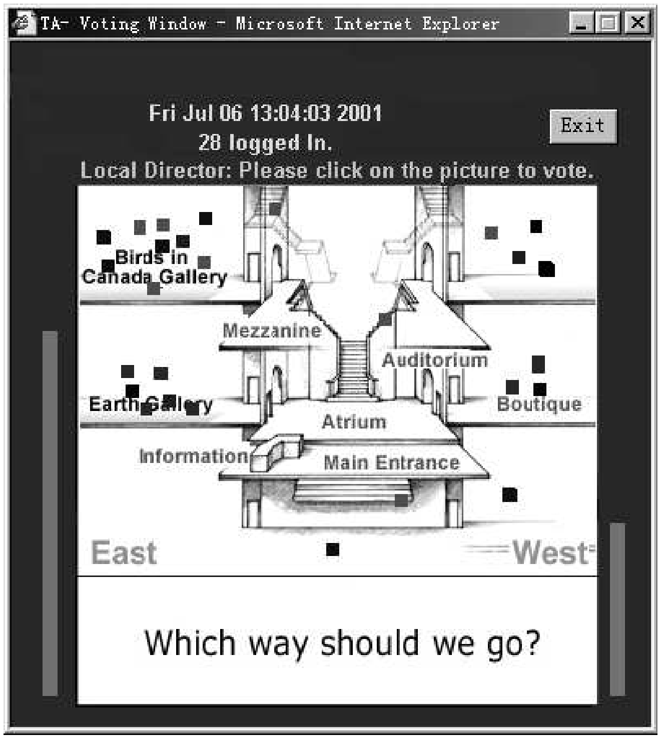


Figure 5.8: *Local Director Software. Local director selects a voting image and propose a voting question.*

the live event. Voting result will drive the event from one scenario to the other, which follows a treelike structure. Scenarios should be educational and interesting.

2. New java-based interface: We have encountered a number of compatibility problems when we use DHTML as the main programming tool to build SDV interface. DHTML also has very limited functionality. We decided to switch to java based interface since version 4.0. The java-based interface allows us to animate the votel movement and employ more flexible voting intervals. A screen shot of new interface is shown in figure



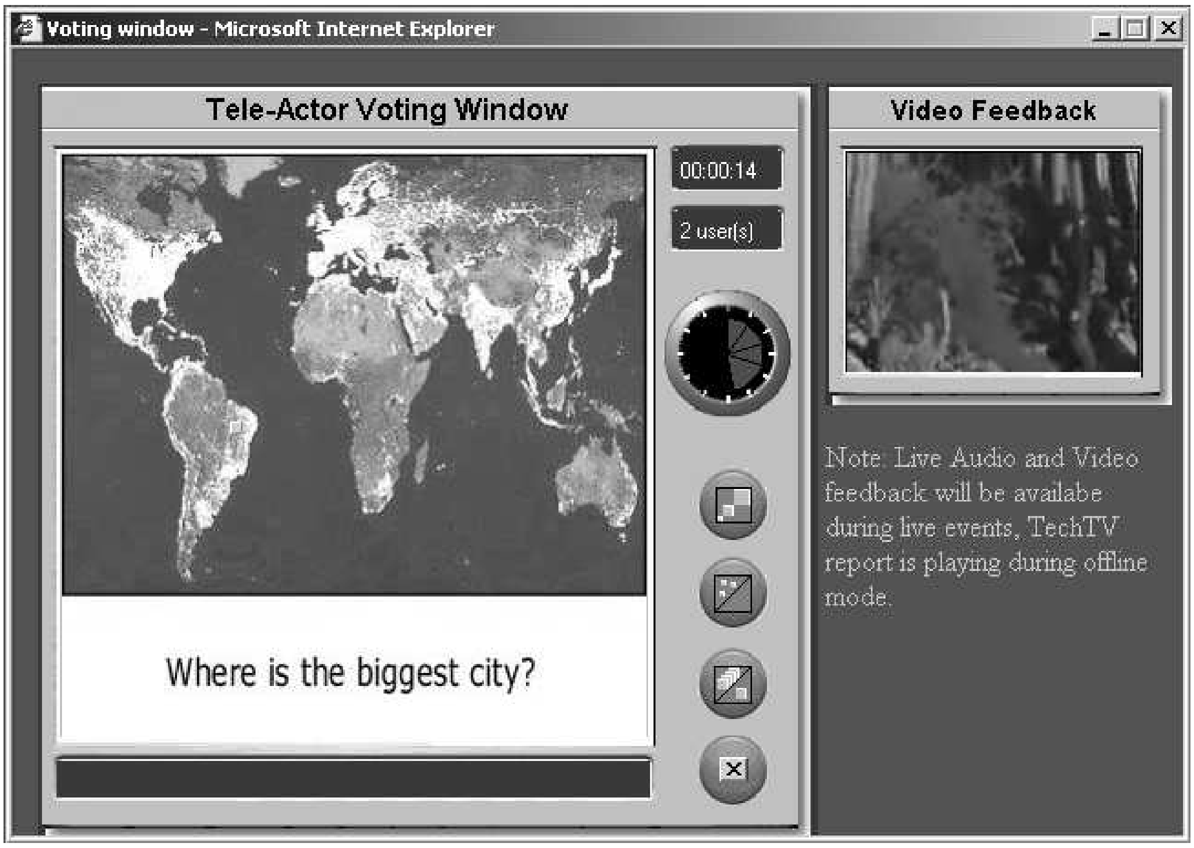
Fri Jul 06 13:04:03 2001
28 logged In.
Local Director: Please click on the picture to vote.

Which way should we go?

Figure 5.9: *DHTML-based SDV interface in version 3.0*

5.10.

3. Improved voting image quality: We found that the image quality provided by analog wireless camera is not satisfying if illumination condition is not good, which is a common problem for most indoor environments. We used Sony camcorder with night vision functionality to solve the problem. As shown in figure 5.2, we also used a dedicated cameraman to shoot the video. Sony camcorder has local video display, which lets cameraman know the image quality immediately. Image capturing task has been shifted to cameraman instead of local director. A shutter-like device has been added to the system to allow cameraman to capture the precious moment. Cameraman also


Voting window - Microsoft Internet Explorer
Tele-Actor Voting Window
Video Feedback
00:00:14
2 user(s)
Where is the biggest city?

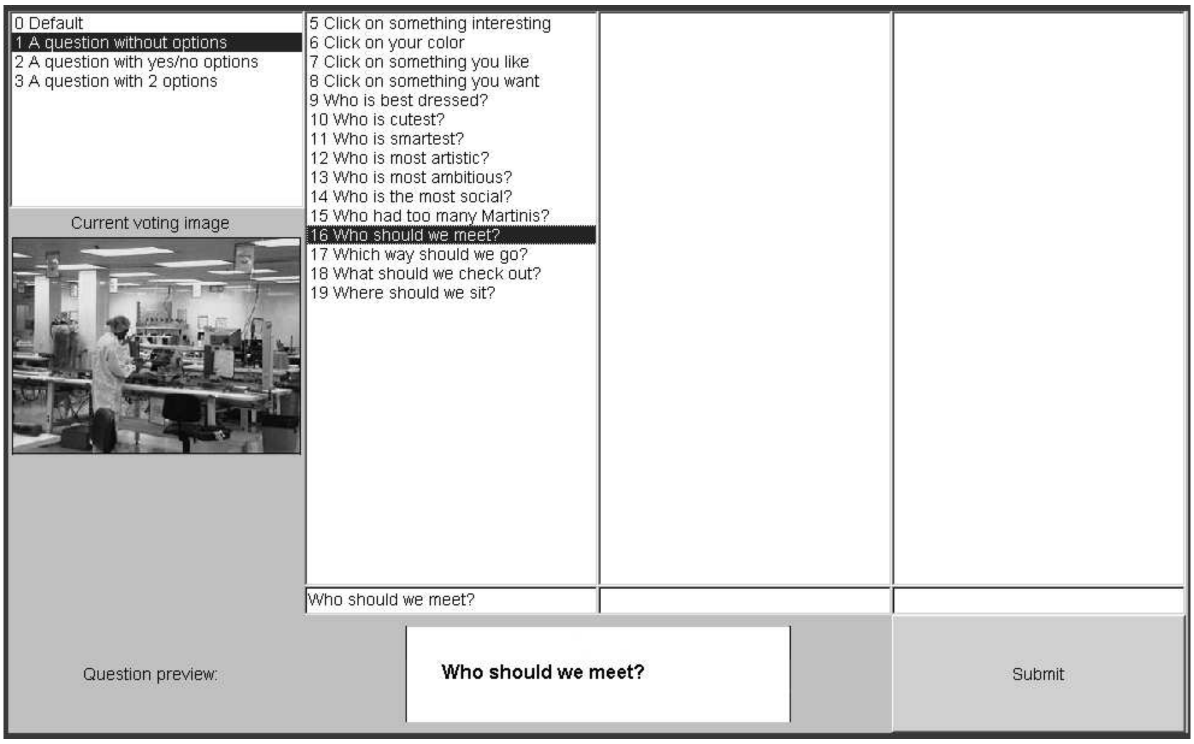
Note: Live Audio and Video feedback will be availabe during live events, TechTV report is playing during offline mode.

Figure 5.10: *Java-based SDV interface in Version 9.0*

has a small LCD display mounted on his elbow to show the current voting information to facilitate view planning.

4. Facilitate voting question formulation: Local director only focuses on posting voting questions. A new java applet supported by C-based CGI script has been developed to replace the software shown in figure 5.8. Some candidate questions are pre-stored in database to reduce the time delay caused by voting question formulation. The screen shot of the local director applet is shown in figure 5.11.

5. Video/Audio broadcasting: Starting from Version 8.0, we provide remote users with


0 Default
1 A question without options
2 A question with yes/no options
3 A question with 2 options

5 Click on something interesting
6 Click on your color
7 Click on something you like
8 Click on something you want
9 Who is best dressed?
10 Who is cutest?
11 Who is smartest?
12 Who is most artistic?
13 Who is most ambitious?
14 Who is the most social?
15 Who had too many Martinis?
16 Who should we meet?
17 Which way should we go?
18 What should we check out?
19 Where should we sit?

Who should we meet?

Question preview: **Who should we meet?**

Figure 5.11: *Java applet for local director*

streaming video/audion feedback. As shown in the figure 5.2, a new video server has been added to the system. Figure 5.10 also shows that a small video window has been added to top-right corner of the screen. We have compared several kinds of video streaming technologies and selected Microsoft Media Encoder as our broadcasting software.

6. New wireless technology: In Version 3.0, we used analog 2.4 GHz wireless camera to transmit image from event field to local base station. We soon learned that analog 2.4 GHz based camera is vulnerable to interference and has very limited image quality. Also, the distance between local base station and event field is limited by the range

of the analog signal. We then switched to 802.11b protocol based wireless ethernet, which is usually hooked up to Internet. We digitize the image locally and send it to local base station using TCP/IP. Based on the new approach, the local base station can be located at anywhere on the Internet. Also the camera selection becomes more flexible because we decoupled the camera and wireless communication.

7. New hardware design: The primary Tele-Actor is carrying a 600 Mhz SONY picture book laptop with 128MB memory connected to a 11Mbs 802.11b wiress LAN at the remote site. It has a USB video card, which captures video at 320 × 240 resolution. The cameraperson has a PIII 750Mhz SONY VAIO latop with 256Mb memory with similar USB video capture device. The laptops direct their video displays to hand-mounted TVs to provide updates on voting patterns. Figure 5.2 show that the primary

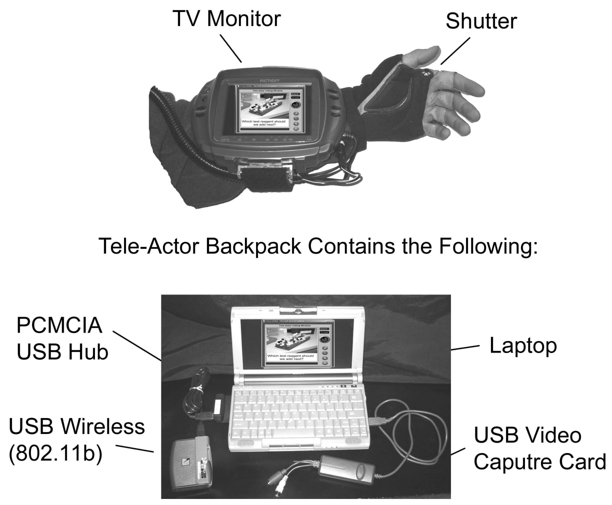


Figure 5.12: *Hardware configuration for the camera-person. The hardware configuration of the Tele-Actor is similar but has a helmet-mounted camera.*

Tele-Actor has a Canon camera mounted on her helmet. Figure 5.12 shows that the camera-person has a Sony camcorder with night vision capability, which provides very high quality image and video stream. Both of them are equipped with a shutter-like device to allow them to capture the precious moment in the live event.

5.6 Problem Definition and Algorithms

Users express responses by clicking on the voting image to spatially indicate a preferred object or direction in the field of view. As an alternative to semantic analysis of the voting image, we consider votels as spatial distributions and identify preferred "consensus" regions in the image. We then use these regions to define two metrics for individual and group performance in terms of leadership and collaboration.

5.6.1 Problem Definition

Voter Interest Functions

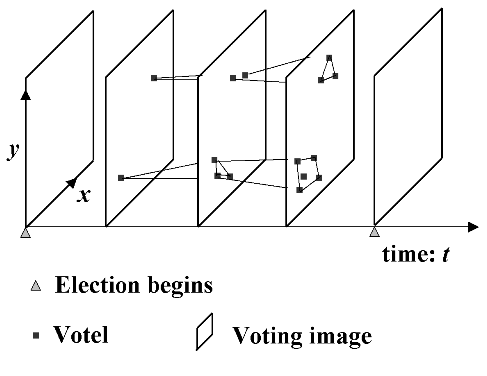


Figure 5.13: *Evolution of voting image as votels arrive.*

Consider the kth voting image. The server receives a response from user i in the form of an (x, y) mouseclick on image k at time t. We define the corresponding *votel*:

$v_{ik}(t) = [x_{ik}(t), y_{ik}(t)]$.

Consensus Regions

Votels are usually clustered around some regions in a voting image. We refer to these regions as *consensus regions*:

$$S_k = \{C_{1k}, C_{2k}, ..., C_{mk}\}.$$

Since there are n voters, $m \leq n$.

Given $V_k = \{v_{ik}(T)\}$, $i = 1, ..., n$, we can compute the consensus regions S_k.

One approach is to use existing methods: cluster analysis[16, 65, 116] and convex hull generation. After votels are classified into groups, we can compute the convex hull of each group with 3 or more votels and treat each convex polygon as a consensus region. In the rest of the section, we analyze voting patterns in terms of goals and collaboration based on known consensus regions $\{C_{1k}, C_{2k}, ..., C_{mk}\}$.

5.6.2 Ensemble Consensus Region

Given S_k, V_k, the *ensemble consensus region* is a region with the most votels. Let

$$I_k(i,j) = \begin{cases} 1 & \text{if } [x_{ik}(T), y_{ik}(T)] \in C_{jk} \\ 0 & \text{otherwise} \end{cases}$$

The count

$$n_{kj} = \sum_{i=1}^{n} I_k(i,j)$$

is the number of votels inside consensus region j of voting image k. Breaking ties arbitrarily, let C_k^*, the ensemble consensus region, be any C_{jk} with max n_{kj}.

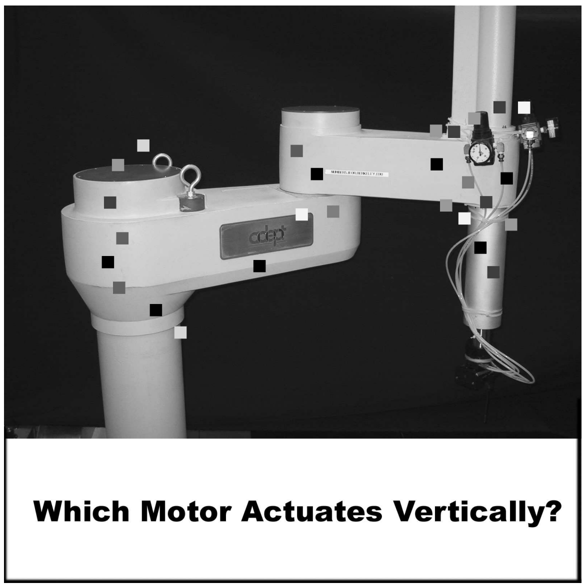


Figure 5.14: *Voting image of an industrial robot arm with 27 votels.*

C_{jk}	Interval	Width	#Votes	D_{kj}
1	[52, 94]	42	8	2.26
2	[139, 180]	51	5	1.16
3	[236, 288]	52	14	3.19
Overall	–	145	27	2.21

Table 5.1: *SDV analysis of Voting Image from Figure 5.14. Intervals and widths are in pixels.*

A consensus region can be projected onto a line in the voting image plane to obtain a consensus *interval.* Table 5.6.2 summarizes votel analysis for the votels shown in Figure 5.14, where consensus regions are projected onto the x axis to obtain three consensus intervals. Consensus interval 3, with the most votels, is the ensemble consensus interval.

5.6.3 Collaboration Metric

To what degree are voters collaborating? We define a measure of collaboration based on the density of votels in each consensus region. For consensus region j in voting image k, define the votel density ratio as:

$$D_{kj} = \frac{d_{kj}}{d_k} = \frac{\frac{n_{kj}}{a_{kj}}}{\frac{N_k}{A}} = \frac{n_{kj}}{N_k} \, (\frac{A}{a_{kj}})$$

where d_{kj} is the votel density (votes per unit area) for consensus region j, d_k is the overall average votel density for the voting image k, n_{kj} is number of votels in consensus region j, a_{kj} is the area or width of the consensus region j, N_k is the total number of votes and A is the area of the voting image. This metric is proportional to the ratio n_{kj}/N_k and inversely proportional to the area of the consensus region. The metric is high when many votes are concentrated in a small consensus region and low when votes are uniformly spread among multiple consensus regions. We can also compute an overall collaboration level for voting image k:

$$D_k = \frac{\sum n_{kj}}{\sum a_{kj}} \frac{A}{N_k} = A / \sum a_{kj}$$

which can measure of how focused the votels are.

Table 2 gives results for another voting image. The collaboration measure for each consensus region is given in the last column of Tables 1 and 2. In table 5.6.3, the data suggests that users are collaborating in a focused manner to vote for consensus interval 2 even though it has fewer votes than consensus interval 3.

C_{jk}	Interval	Width	#Votes	D_{kj}
1	[44, 84]	40	10	2.35
2	[141, 168]	27	6	3.32
3	[223, 283]	60	16	2.51
Overall	–	127	32	2.37

Table 5.2: *SDV Analysis for another voting image.*

5.7 Online Field Tests

We performed a half-hour field test on 8 Nov 2001, with 25 7th-grade students from Dolores Huerta Middle School, and used the Tele-Actor to visit the UC Berkeley Microlab to learn how microchips are made. The Microlab is located at fourth floor of Cory Hall, UC Berkeley. Microchip fabrication needs a clean room environment and there are hazardous materials being used in fabrication process as well. It is usually difficult to arrange a field trip for students to such environment. Our Tele-Actor, who is well-trained and aware of safety and security issues, is directed by the 7th Grader students to explore the lab.

A second field test was conducted on July 25th, 2002, with 26 9th grade-students visiting a Biotechnology lab at the Lawrence Berkeley National Laboratory. This was part of the Robot-Clone-Human high-school curriculum project involving UC Berkeley's Alpha Lab, the Interactive University Project, and San Francisco Unified School District. This project is developing a teaching module geared for high school biology students to learn about what biotechnology is, how robots work, and how robots are used in biotech.

The third field test was conducted on Oct 23th, 2002 with 23 students from a UC Berkeley Mechanical Engineering graduate course, who used the Tele-Actor interface to visit the UC Berkeley Microlab to learn details about the wafer manufacturing process.

5.8 Conclusions

This chapter describes a networked teleoperation system that allows groups of participants to collaboratively explore remote environments. We propose two innovations: the SDV, a networked interface for collecting spatial inputs from many simulataneous users, and the "Tele-Actor," a skilled human with cameras and microphones who navigates and performs actions in the remote environment based on this input. We presented system architecture, interface, experiments, and metric that accesses the collaboration level.

Collaborative teleoperation systems will benefit from advances in broadband Internet, local wireless digital standards (802.11x), video teleconferencing standards, and Gigahertz processing capabilities at both client and server. We are working on efficient algorithms for consensus identification and "scoring" to motivate user interaction. We will perform a series of field tests with different user groups and different remote environments.

In related research, we are developing collaborative teleoperation systems where the shared resource is a machine such as a robotic pan-tilt-zoom camera. Our goal is systems that are viable for very large groups (1000 person and up), allowing collective exploration over networks such as Internet2 and interactive television.

To experiment with the latest version of our system, please visit: www.tele-actor.net.

Chapter 6

Algorithms for The Tele-Actor: Unsupervised Scoring for Scalable Internet-Based Collaborative Teleoperation

6.1 Introduction

In Tele-Actor field tests with students ranging from middle and high-school we discovered that students would often become passive, simply watching the video without participating. Instructors asked for a mechanism to quantify student engagment for grading purposes. We realized we could address both issues by introducing a scoring metric that would continually assess students and provide a competitive element to the interactions.

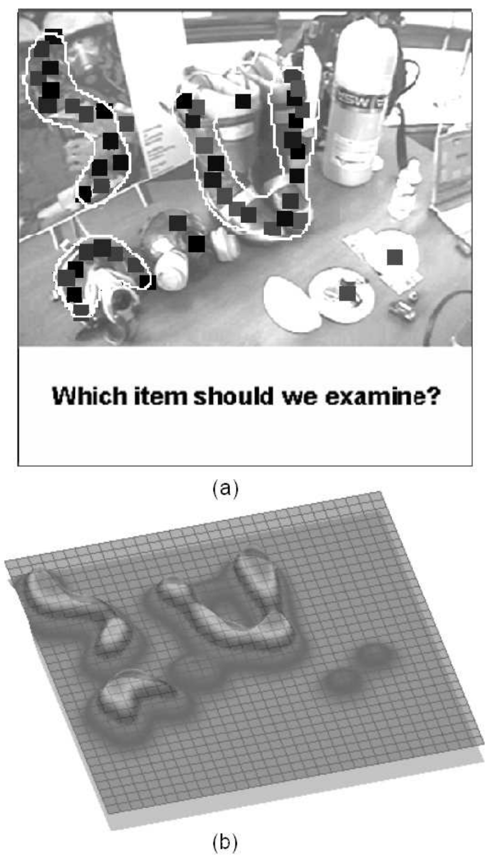


Figure 6.1: *Above, (a) illustrates a sample view from our scalable user interface, where spatial inputs from users are represented with colored square markers, in this case indicating points of interest in a live video image. The unsupervised scoring metric assigns a scalar value to each input. Below, (b) illustrates the aggregate (consensus) density distribution that is used to automatically compute scores based on vote arrival time.*

This chapter describes an unsupervised scoring metric and algorithms for rapidly computing it. The metric is "unsupervised" in the sense that it does not rely on a human expert to continuously evaluate performance. Instead, performance is based on "leadership": how quickly users anticipate the decision of the majority.

As illustrated in Figure 6.1, the unsupervised scoring metric is based on clustering of user inputs. For n users, computing scores runs in $O(n)$ time. This chapter presents problem formulation, distributed algorithms, and experiment results.

6.2 Related Work

Networked robots, controllable over networks such as 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. There are now dozens of Internet robots online, a book from MIT Press [49], and an IEEE Technical Committee on Internet and Online Robots. See [63, 104, 66, 70, 88, 57, 93, 77, 83, 82] examples of recent projects.

Tanie, Matsuhira, Chong, et al. [21] proposed a useful taxonomy for teleoperation systems: Single Operator Single Robot (SOSR), Single Operator Multiple Robot (SOMR), and Multiple Operator Multiple Robot (MOMR).

Most networked robots are SOSR, where control is limited to one human operator at a time. Tanie et al. analyzed an MOMR system where each operator controls one robot arm and the robot arms have overlapping workspaces. They show that predictive displays and scaled rate control are effective in reducing pick-and-place task completion times that require cooperation from multiple arms [21].

In an MOMR project by Elhajj, Fukuda, Liu, Xi, and colleagues [30, 31], two remote human operators collaborate to achieve a shared goal such as maintaining a given force on an object held at one end by a mobile robot and by a multi-jointed robot at the other. MOMR models are also relevant to online collaborative games such as *Quake* and *The Sims Online*, where players remotely control individual avatars in a shared virtual environment.

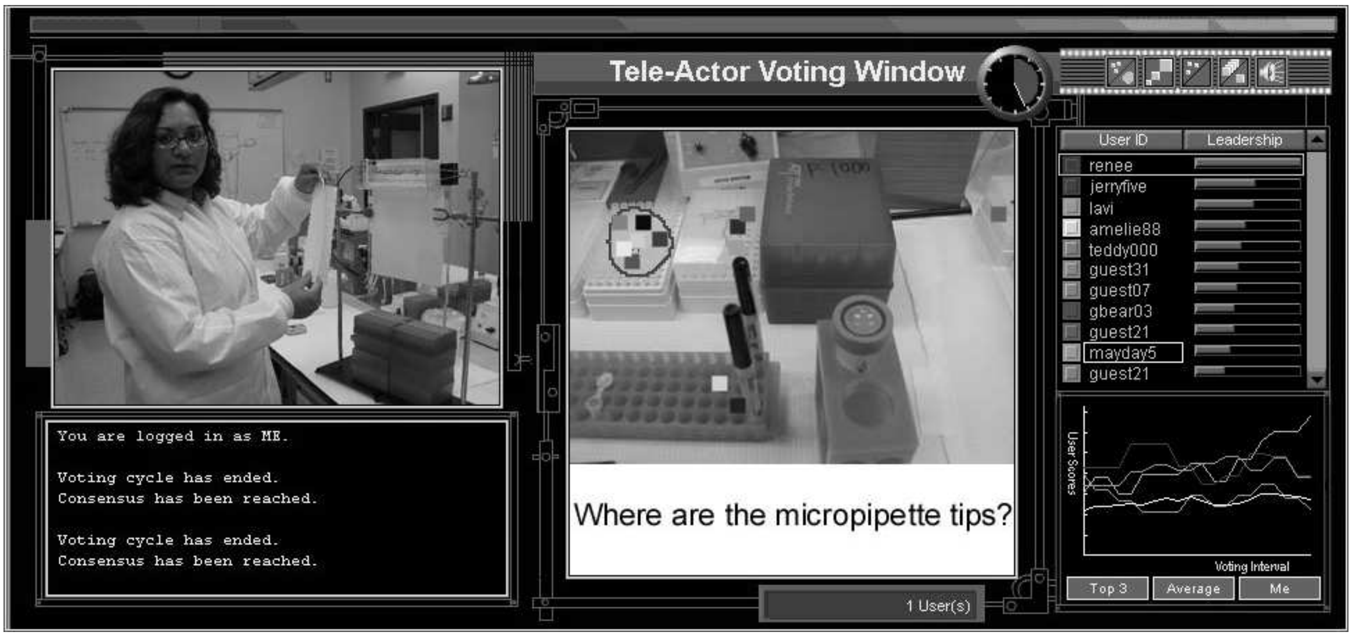
One precedent for an online MOSR system is described in McDonald, Cannon and colleagues [85]. For waste cleanup, several users to assist remotely using Point-and-Direct (PAD) commands [19]. Users point to cleanup locations in a shared image and a robot excavates each location in turn. In this Internet-based MOSR system, collaboration is serial but pipelined, with overlapping plan and execution phases. The authors demonstrate that such collaboration improves overall execution time but do not address conflict resolution between users.

In [45, 44], Goldberg, Chen, et al present an Internet-based MOSR system that averaged multiple vector inputs to control the position of an industrial robot arm. In [110, 111], we present another MOSR system that allow a group of users to simultaneously share control of a single robotic camera. We formulate MOSR problem as resource allocation problem and develop algorithms for camera control.

Outside of robotics, the notion of MOSR is related to a very broad range of group activities including compute gaming, education, social psychology, voting, etc. Although there currently are no standard "best practices" for the design of scoring systems in the computer games industry, some tentative efforts recently have been made to identify the key features of an effective scoring and ranking model. GamaSutra, the flagship professional resource for digital game designers from around the world, has proposed four characteristics of a successful scoring system. It should be reproducible (consistent), comprehensive (incorporating all significant aspects of game play), sufficiently detailed (so that players understand how it is calculated), and well-balanced (allowing for players to stay competitive even when playing from a disadvantage) [72]. Kriemeier suggests that a failure to address each of these four principles will negatively affect user participation and motivation. Proposed customizations to further increase player motivation in a system with these four standard features include introducing non-zero sum (cooperative) scoring elements and a score decay algorithm that will penalize players for a lower rate of participation in the game. The Tele-Actor unsupervised scoring model aims to incorporate these features, and to investigate their value and effectiveness from a user-perspective.

Two different areas in educational psychology investigate the usefulness of games in learning. The first addresses the element of "fun" in games as a powerful motivational tool [43, 60]. In particular, cooperative play models are thought to provide particularly powerful evidence of games as tools of engagement; as Sutton-Smith notes [113], children and young adults are so motivated to be accepted in such play, they make sacrifices of egocentricity for membership in the group, a claim that may be tested by the unsupervised scoring model.

Csikszentmihalyi's theory of "flow" [24], a confidence-building state in which a participant becomes absorbed in and highly effective at a particular activity, argues for the importance of feedback as the primary criterion for achieving flow. According to Csikszentmihalyi, flow creates a sense of motivation that is intrinsic to the activity, rather than relying on extrinsic rewards (prizes, grades, acclaim), and intrinsic motivation is ultimately more important for long-term success in any activity. For example: Gee [40] argues that students today are used to experiencing flow and achieving a sense of mastery in their at-home game play, but not in school. Many educators are considering adopting more game-like scenarios for learning as a way to incorporate students' proclivity for structured play.



```
You are logged in as ME.

Voting cycle has ended.
Consensus has been reached.

Voting cycle has ended.
Consensus has been reached.
```

Where are the micropipette tips?

Figure 6.2: *The Spatial Dynamic Voting (SDV) interface as viewed in the browser by each online voter. Low framerate live video (approx 1 fps) is displayed in the left window. Users vote on spatial preferences using their mouse to position a small marker (votel) in the middle (voting) window, over either a prestored or live image. Users view the position of all votels and can change their votel positions based on group dynamics. As described below, votel positions are processed to assign scores to each user. The list of active voters, ranked by score is displayed in the right window. The lower right window displays a plot of voter score, overall average, and scores for the top three voters.*

Group decision-making is sometimes modelled as an n-person (multi-player) co-operative game [120, 4]. We can view the unsupervised scoring system as a nonzero-sum multi-player game. in which users compete and cooperate to increase their individual leadership scores. The open-endedness of the n-person, non-zero sum scenario makes its outcome the most theoretically challenging to predict in game theory. The data collected on user behavior in the form of leadership scores may be useful in evaluating a variety of predictive models for n-person games and the associated likelihood and structures of cooperative behavior in real-life [5, 84, 107].

A related question is, how do players learn game strategies and adapt their game play behavior accordingly over time? The study of adaptive learning models in games attempts to develop a theory of "game cognition" that explains why people do not always discover or follow optimal strategies in game play [81]. Erev and Roth [33], for instance, argue for the importance of the presence or absence of reinforcement and feedback within a repetitive game structure as the most important factor in predicting player action. By providing feedback in the form of a leadership score and simultaneously tracking changes in scores over time, we can quantify how well scores correlate with group performance.

6.3 Problem Definition

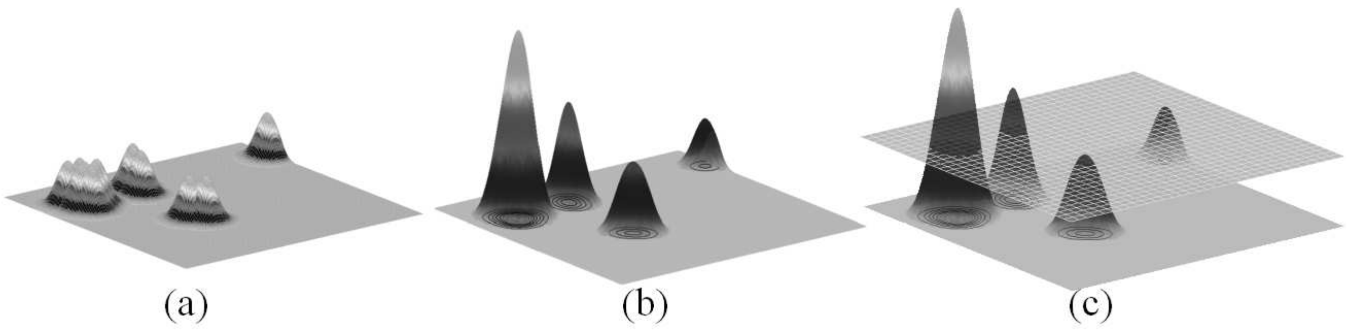


Figure 6.3: *An example of voter interest functions, the corresponding majority interest function, and an illustration of consensus region generation for the voting image in Figure 6.2.*

Figure 6.2 illustrates the user interface.

In this section, we propose an unsupervised scoring metric based spatial distributions of votes.

6.3.1 Inputs and assumptions

Consider the kth voting image. The server receives a response from user i in the form of an (x, y) mouseclick on image k at time t. We define the corresponding *votel*: $v_{ik}(t) = [x_{ik}(t), y_{ik}(t)]$.

Each votel represents a user's response ("vote") to the voting image. We model such responses with a *voter interest function*, a density function based on the bivariate normal distribution:

$$f_{ik}(x, y) \sim N(v_{ik}(t), \Sigma_{ik}(t))$$

where $v_{ik}(t)$ is the mean vector and $\Sigma_{ik}(t)$ is a 2×2 variance matrix, such that,

$$\iint_{\sigma} f_{ik}(x, y)\ dx\ dy \ = \ 1$$

where σ is the area of the voting image. Since σ is a bounded 2D region, the voter interest function is a truncated bivariate normal density function with mean at $v_{ik}(t)$ as illustrated in Figure 6.3(a).

Majority Interest Function

When voting on image k ends at stopping time T, the last votel received from each of n active voters determines V_k, a set of n votels. We define the *ensemble interest function* for voting image k as the normalized sum of these voter interest functions.

$$f_k(x, y) = \frac{1}{n} \sum_{i=1}^{n} f_{ik}(x, y).$$

Consensus Regions

As illustrated in figure 6.3(c), we can extract spatial regions by cutting the ensemble interest function using a horizontal plane at a height proportional to the overall volume. Let z_k be the cutting threshold. The z_k value satisfies the condition that the ratio between the partial volume of the ensemble interest function above the horizontal plane and the total volume of the ensemble interest function is constant r. We use a value of 0.10 (10% of the volume lies above the plane). The cutting plane defines an iso-density contour in the ensemble interest function that defines a set of one or more closed subsets of the voting image,

$$S_k = \{(x, y) | f_k(x, y) \geq z_k\}.$$

As illustrated in Figure 6.3(c), we refer to these subsets as *consensus regions*:

$$S_k = \{C_{1k}, C_{2k}, ..., C_{lk}\}.$$

Since there are n voters, $l \leq n$ is number of consensus regions.

Majority Consensus Region

Given S_k, V_k, the *majority consensus region* is the region with the most votels (breaking ties arbitrarily). Let

$$I_k(i,j) = \begin{cases} 1 & \text{if } [x_{ik}(T), y_{ik}(T)] \in C_{jk} \\ 0 & \text{otherwise} \end{cases}$$

The count

$$n_{kj} = \sum_{i=1}^{n} I_k(i,j)$$

is the number of votels inside consensus region j of voting image k. Breaking ties arbitrarily, let C^*_k, the majority consensus region, be any C_{jk} with max n_{kj}.

6.3.2 Unsupervised Scoring Metric

We measure individual performance in terms of "leadership". By definition, a "leader" anticipates the choices of the group. In our context, a leader is an individual who votes early in a position consistent with the majority consensus region. Define I_s is an outcome index for voter i and voting image s:

$$I_{s,i} = \begin{cases} 1 & \text{if } [x_{i,s}(T_s), y_{i,s}(T_s)] \in C^*_s \\ 0 & \text{otherwise} \end{cases}$$

Define $t_{s,i}$ as the duration of the time that voter i's votel stays in the majority interest region for the s^{th} voting image, T_s is the total voting time for voting image s. Therefore term $\frac{T_s - t_{s,i}}{T_s} I_{s,i}$ characterizes how well the voter anticipated the majority consensus region for the voting image s.

To smooth out rapid changes in user scores, we pass the term to the following low pass filter to get a stabilized "Leadership score":

$$L_{k+1,i} = (1-\alpha)L_{k,i} + \alpha \frac{T_k - t_{k,i}}{T_k} I_{k,i}$$

where the initial value $L_{0,i} = 0$ for each voter i. The value of filter factor α is set to 0.1 in our experiments to allow smooth fluctuation in user scores.

Note that this scoring metric depends only on the spatio-temporal pattern of votes and does not require a human expert.

6.4 Distributed Algorithm

To compute the unsupervised score for each user, we start by maintaining the ensemble interest function with a grid. We partition the voting image into 160 $\times$ 160 regular cells. For each voter interest function $f_{ik}(x, y)$, we discretize it into a 2D array with respect to the same lattice resolution. Depending on the variance of the Gaussian function and accuracy threshold, each element of the 2D array only contains constant number of non-zero entries. Therefore, to compute the ensemble interest function for n votes, we add those n 2D arrays into the cells. This operation takes $O(n)$. Figure 3(b) shows the shape of the ensemble interest function for the voter interest function in 3(a).

As illustrated in figure 3(c), we can extract spatial regions by cutting the ensemble interest function using a horizontal cutting plane. The next step is to compute the height of the cutting plane z_k for the given volume ratio r. Define $r(z)$, $z > 0$ be the volume ratio between the partial volume of the majority interest function above the cutting plane with height z and the total volume of the ensemble interest function. As z increases, the horizontal plane rises. Hence $r(z)$ is a monotonic decreasing function of z. A binary search can find z_k in $O(log(1/\varepsilon))$ steps with error ε. Since we need to compute the partial volume for each step, which takes at most $O(m)$ for m cells in the grid. Therefore, the complexity for computing the threshold is $O(log(1/\varepsilon)m)$. If we consider ε as a constant, then it is $O(m)$.

The second step is to perform the threshold for the ensemble function and find connected components, each of which forms a consensus region. The connected components algorithm makes two passes through the 2D array processing a row of cells at a time, from left to right. During the first pass, each cell is assigned the minimum label of its neighbors or if none exists, a new label is assigned. If neighboring cells have different labels, those labels are considered equivalent and entered into an equivalence table. The second pass uses the equivalence table to assign the minimum equivalent label to each non-zero entry. Since, this step also takes $O(m)$ time, the total computation time for computing the consensus regions for a given ensemble interest function is $O(m)$.

To determine the majority consensus region, we need to count the number of votels inside each consensus region. For this purpose, each cell maintains a votel count that is updated during votel insertions. In a single pass, we can sum the number of votels in the cells belonging to each consensus region. Thus, this step takes $O(m)$ time.

If we add computation time of algorithms for ensemble interest function, consensus regions, majority consensus regions, and leadership score together, the total computation time is $O(n + m)$.

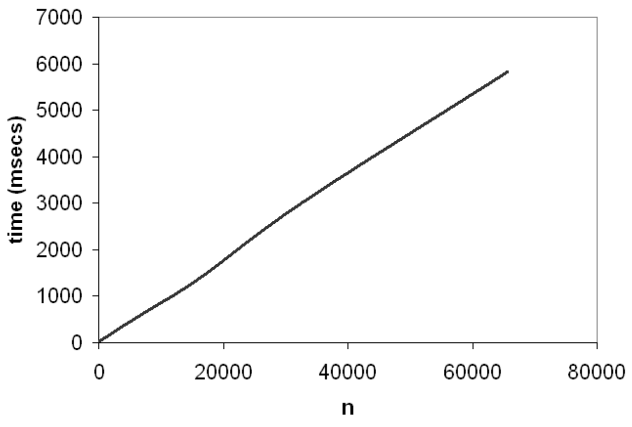


Figure 6.4: *Processing time for computing the unsupervised scoring metric as a function of the number of voters based on trials using random voter positions.*

The algorithm has been implemented and runs on the client side using Java. We tested the algorithm using a 750Mhz PC Laptop with 256Mb RAM. Figure 6.4 illustrates the linear scalability of the algorithm in terms of number of voters. The Tele-Actor server is primarily responsible for distributing voting data to all users. Each client sends the user's voting data and receives updated voting data from the Tele-Actor server every 1.0 seconds. For every server update, at most n new voter interest functions will need to be inserted. Using the updated interest functions, the consensus regions and the majority consensus region are recomputed. When a voting cycle ends, each voter computes only the voter's new leadership score, thus distributing the scoring calculations among the clients.

6.5 The “Tele-Twister” Application

To understand how the unsupervised scoring metric works with groups of participants and to test software reliability, we developed a collaborative teleoperation system based on a popular party game. The result is a sequence of multi-player non-zero sum games embedded inside a sequence of two-player zero sum games.

Twister was the first board game where human bodies are the board pieces. In this classic game, human players interact over a large horizontal playing board. Players sequentially place their hands and feet on colored circular targets chosen randomly (eg, Left foot: GREEN). The challenge for players is to maintain placement of hands and feet in increasingly difficult combinations without falling over.

In our version, Tele-Twister, there are two human players called “twisters”. One twister is dressed in red, the other in blue, as illustrated in figure 6.5. Remote participants (”voters”), download the Java applet and are assigned to one of two teams, red or blue. In

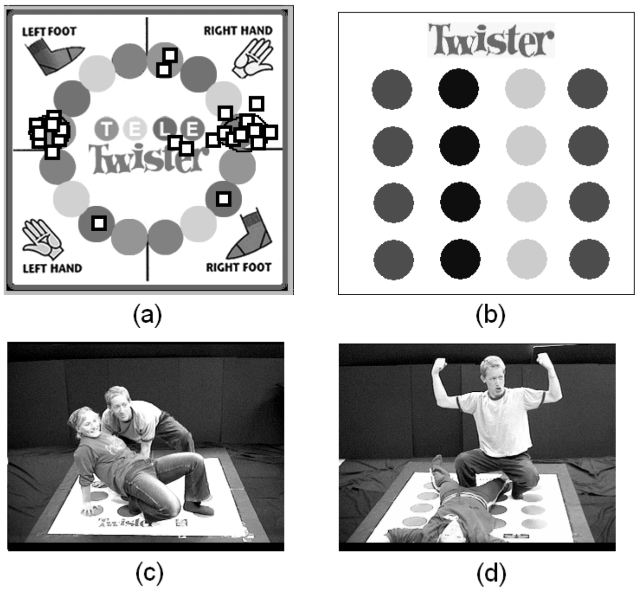


Figure 6.5: *Tele-Twister: a collaborative teleoperation system where online voters direct the movements of human players.*

Tele-Twister, random target selection is replaced by the teams, who view game status using the low framerate video and vote using the interface to collectively decide on the targets and compete to win: having their opponent fall over first.

A typical voting pattern is illustrated in Figure 6.5(a). Votel positions for all online voters are updated every second and displayed at each voter's browser. Consensus regions are computed and updated continuously by each browser. voters are free to change their votel position at any time during the voting cycle, but when it ends, the majority cluster determines the next move for the human twisters.

Tele-Twister is thus strategic: the red team chooses targets that are easy for the red twister and difficult for the blue twister, and vice versa for the blue team. Tele-Twister encourages active collaboration within teams and competition between teams. In this context, the unsupervised scoring metric rewards active participation and collaboration (voters who don’t vote or are outside the majority region at the end of a voting cycle receive a zero score).

Figure 6.5(b) shows the board layout, with 16 circles: red, blue, yellow, and green. Since August 12th, 2003 we have conducted public ”field tests” on Fridays, from 12-1pm Pacific Time. These field tests attract 10-25 online participants, who vote in alternating 30 second voting periods until one twister falls, ending the round. Typically, each round continues for 10-20 voting periods, so we conduct 4-5 rounds during each field test. Figure 6.5(c) and (d) are two snapshots from a typical round.

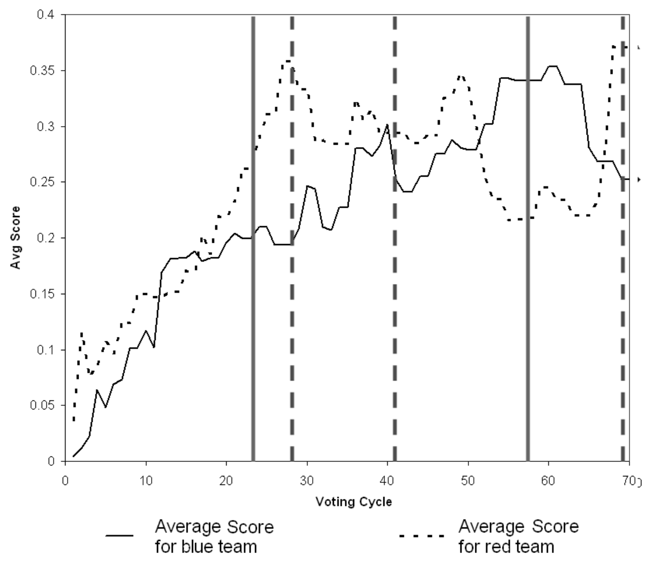


Figure 6.6: *Plot of unsupervised scoring metric from the Sept 26 2003 field test with two teams of 21 online voters, for five rounds of the Tele-Twister game. The figure plots average score for members of each team during sequential voting cycles. Vertical bars indicate the end of a round and which team wins. A solid vertical line indicates that the blue team won that round and a dashed vertical line indicates that the red team won.*

Figure 6.6 shows scoring data from the field test on Friday September 26th, 2003. All players begin the field test with a score of zero, so average score per team climbs during the initial voting cycles: the blue team wins the first round after approx 23 voting cycles.

How do user scores correlate with task performance (winning the round)? Note that in the four subsequent rounds, the team with the highest average score consistently wins the round: the red team wins rounds 2,3 and 5 and the blue team wins round 4. During round 4, members of the red team had difficulty agreeing on the appropriate next move, reducing the score of many red voters and in turn the average red team score. The lack of consensus (ie. "split votes") resulted in a loss during that round.

A team has higher average scores when the team collaborates, reaching consensus faster. This does not always correlate with success: it can lead to short-term snap decisions that may appear strong but are strategically weak.

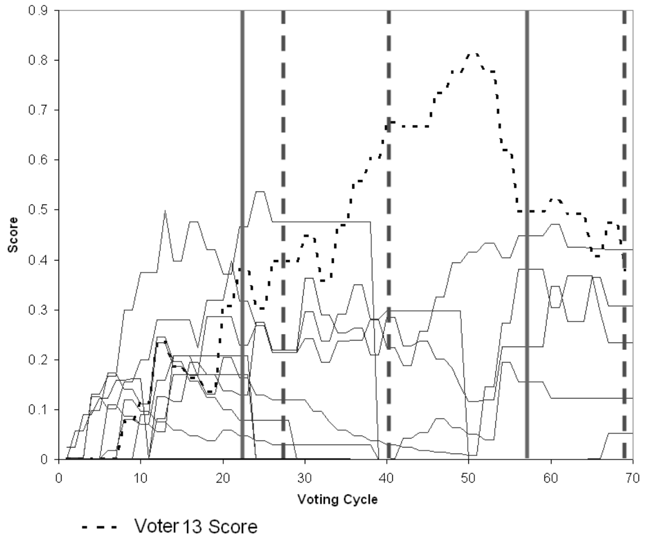


Figure 6.7: *From the same Sept 26 2003 field test, plot of unsupervised scoring metric for seven individual voters from the Blue team.*

Figure 6.7 plots individual voter scores from the same field test. Note that the score of voter 13 is consistently higher. In this case voter 13 is a member of our lab who has played the game during many previous rounds and has developed skill at picking the next moves. Other players follow his moves, resulting in a high score.

6.6 Future Work

This chapter paper describes an unsupervised scoring metric for collaborative teleoperation that encourages active participation and collaboration, and a distributed algorithm for automatically computing it. To understand how the scoring metric works with groups of participants and to test software reliability, we developed a collaborative teleoperation system based on a sequence of multi-player non-zero sum games embedded inside a sequence of two-player zero sum games. Initial results suggest that the metric encourages active participation and correlates reasonably with task performance.

Chapter 7

Conclusions and Future Work

7.1 Contributions

7.1.1 Challenges Identified in CT Systems

Collaborative teleoperation (CT) systems allow many users to simultaneously share control of a single remote physical resource such as a mobile camera or a human explorer over Internet. In CT, the challenges are how to design effective systems and how to compute consensus commands for the shared device. Using a robotic camera and a human explorer as the shared devices, I have studied both systems and algorithms for CT.

7.1.2 Formulation of CT Problems and Metrics

To compute consensus commands, I formulate the problem as an optimization problem: maximize total user satisfaction levels/reward by choosing the optimal control command. The satisfaction level/reward is defined as a metric function of users' inputs and the current control command.

User inputs for the Co-Opticon project are isooriented and congruent planar rectangles. Output, which is used to control the shared device, is a rectangle ensuring that total satisfaction levels are maximized. User inputs for the Tele-Actor project are planar points. Output of the Tele-Actor project is a closed region describing the most interesting region in the voting image.

These two problems for CT systems can be seen as instances of a class of problems described by: given n requests from a parameterized family of objects and an objective function, choose an optimal set of k representatives from a (possibly different) family of objects, where $k < n$. The objective function depends on definition of satisfaction/reward metric function.

For the Co-Opticon system and its extension in the satellite frame selection, the metric function depends on how the camera frame resembles the user requests with respect to location, shape, and resolution. The traditional similarity metrics such as Intersection Over Union (IOU) and Symmetric Difference (SD) in pattern recognition literature only measure location difference and shape difference but can not measure resolution difference. We propose Coverage-Resolution (CR) metric, which captures the location difference, the shape difference, and the resolution difference. The CR metrics are a not a single metric but a parameterized family of metrics. In addition to that, the CR metrics possess a piecewise linearity property, which actually speeds up the algorithms in comparison to the SD or IOU metric. In case that the shape is coupling with the resolution, i.e. a large frame means lower resolution, the CR metric can be reduced to a simple special format: Intersection

No.	System	Type	Zoom	Solution	Complexity
1	Co-Opticon	Centralized	m levels	Exact	$O(mn^2)$
2	Co-Opticon	Distributed	m levels	Exact	Server: $O(mn)$ Client: $O(n)$
3	Satellite	Centralized	Continuous	Exact	$O(n^3)$
4	Co-Opticon	Centralized	Continuous	Approximation	$O((n+1/\epsilon^3)\log^2 n)$
5	Co-Opticon	Distributed	Continuous	Approximation	Sever: $O(n)$ Client $O(1/\epsilon^3)$
6	Tele-Actor	Distributed	-	Approximation	Server: $O(n)$ Client $O(n)$

Table 7.1: *Algorithms developed for CT problems. Recall that n is number of requests.*

Over Maximum (IOM) metric.

7.1.3 Algorithms

I apply knowledge of computational geometry and optimization theory to address the two instances of the CT problems: the Frame Selection problem in the Co-Opticon system and the decision/scoring problem in the Tele-Actor system.

As illustrated in table 7.1.3, for the Frame Selection Problem in Co-Option system, since the speed is a critical issue, I developed both exact and approximation algorithms, the best of which runs in $O((n+1/\epsilon^3)\log^2 n)$ for n requests and approximation bound ϵ. For the Satellite Frame Selection problem, I developed an exact algorithm that runs in $O(n^3)$. For the decision/scoring problem in the Tele-Actor system, I developed Gaussian-based clustering approach and develop linear time approximation algorithms.

7.1.4 System Development and Experiments

I have implemented both systems and they have been extensively field tested with students and online users. We use cutting edge technologies such as high resolution networked cameras, broadband videoconferencing, and wireless networking to enhance collaborative teleoperation and its applications in education, security, entertainment, and journalism.

System Development

I have used C/C++, java, PERL, javascript, SQL to code the Tele-Actor system and the Co-Opticon system. The total amount of coding is more than 200K lines. About 60% of the code is written in C/C++, 30% of the code is written in Java. The system development involves all main stream OSs including Mac, Linux, and Windows. The scope of the coding covers knowledge of Video Streaming, Operating System, Network Communications, Computer Architecture, and Software Engineering. The compiler used includes GNU C++, Microsoft Visual C++, J2SE, and Active Perl. I have also spent a lot time on improving system scalability, reliability, and security when optimizing my development.

Experiments

The Co-Opticon systems was pre-launched inside the Alpha Lab, UC Berkeley in fall of 2002. After 6 months of testing, it was deployed at Evans Hall in Berkeley campus in summer of 2003. The system has been running 24 hours a day and 7 days a week since then. The system has never crashed.

The Tele-Actor system was launched in summer of 2001. Since then, the Tele-Actor visited many places including schools, semi-conductor manufacturing fabrication facilities, biology laboratories, the grand opening of a new building in Berkeley campus, San Francisco Exploratorium, Pasadena Art Center, and the Fifth Annual Webby Awards event. The primary users of the system are online users and students. The students who experienced with the system range from 7th grade high school students to graduate students from Berkeley. We learned important lessons from those experiments, such as network traffic jamming, network security problems, interference of wireless communication, administrative problems, and coordination of teammates from different backgrounds.

Internet Videostreaming for Teleoperation

In both CT systems that we have developed, we need to deliver the representation of the remote environment in video format to online users. We have tested mainstream software packages including Quick time, RealVideo Codec, Microsoft Media Encoder, Cuseeme, and Microsoft Netmeeting. One important experience we have learned from the experiments is that the state of art videostreaming technology can not satisfy the requirements from vision-based teleoperation systems.

In teleoperation, real time video is used as feedback information, which imposes different Quality of Service requirements in comparison to videostreaming for media contents. To deliver video over the Internet, we need to consider video compression standards and network transmission protocols. There is a tradeoff between framerate and resolution for a given bandwidth. There is also a tradeoff between compression ratio and computation

Standards	Buffering Time	Framerate
MJPEG	Negligible	Low
MPEG2	Noticeable	Moderate
MPEG4	Long (8∼10secs)	Highest
H.263+	<300 msecs	High

Table 7.2: *A comparison of existing videostreaming standards for a given resolution of* 320×240.

time for a given CPU. The computation time includes both CPU time and data-buffering time. Video compression is a very computationaly intensive task. A long computation period will introduce latency and significantly impair the tele-operation performance. We can use hardware to cut down the CPU time but not the data-buffering time. There are many standards and protocols available but are just variations of MJPEG, MPEG2, MPEG4, and H.26x family. We compare those standards in the following Table 7.1.4.

From teleoperation point of view, the buffering time determines the latency and the framerate determines the responsiveness of the system. An ideal videostream should have both high framerate and low buffering time. But if both can not be achieved at the same time, low latency is preferred. From Table 7.1.4, H.263+ outperforms other competitors. However, since H.263+ has to use random ports on UDP to transmit video signals, which is blocked by most of today's firewalls, it affects the scope of deployment.

Another interesting observation is that all of those standards try to rebuild a true and complete representation of the field of view. However, it might not be necessary for a teleoperation task. Sometimes, a high level abstraction is sufficient. For example, when a mobile robot is avoiding an moving obstacle, all the robot needs to know is the speed and bounding box of the moving object instead of knowledge that whether this object is human or other robots. We might want to control the level of details in video perception and transmission. This actually imposes a interesting problem: we need a new videostreaming standard that serves for teleoperation.

7.2 Future Work

7.2.1 Big Picture

The research on Collaborative Teleoperation is still at its infancy. The Collaborative Teleoperation can be viewed as an application of collaborative decision making process for a tele-robot over the Internet. As discussed in previous chapters, we have tested voting and optimization as the collaborative decision making strategies using two kinds of shared devices. However, there are other group decision strategies available, i.e. auction-based schemes, that may worth trying. Game theory based approach can also be applied here. Some collaboration methods can be viewed as cooperative game while others can be viewed as non-cooperative game. There also may be a hybrid of both as we discussed in Chapter 6.

Further research on how to select an optimal collaborative teleoperation strategy and how to classify a family of robotic device that share the same collaboration strategy is necessary. The research can not be done without extensive experimenting with different robotic devices and different group-making strategies.

7.2.2 Extensions of Frame Selection Problems

4-D 1-Frame Problem

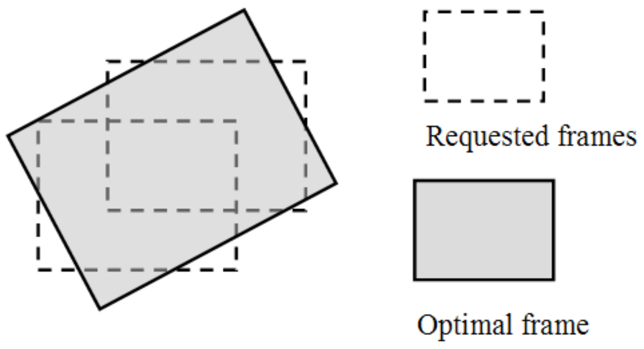


Figure 7.1: *The Optimal Frame is not necessarily an iso-oriented rectangle.*

In the Chapter 3 and Chapter 4, we assume that the camera frames are iso-oriented rectangles. However, this may not be adequate if the camera can rotate along its optical center axis. For example, it becomes more and more common for modern imaging satellites to perform such rotations. As illustrated in figure 7.1, this introduce a 4th degree of freedom in the camera control in addition to the pan, tilt, and zoom motions. This problem has not be addressed in the thesis. It is unclear how to get the exact optimal solution at this moment. Our initial study show that we may be able to give a bounded approximation solution by modifying our algorithms in Chapter 3.

3D 1-Frame Problem with Dwelling Time

One frequently asked problem is that some requested frames may never get satisfied if the request is located in a non-popular region. We touch briefly on the problem in Chapter

2 by using memory-based approaches. However, a better approach is to modify our metric to plan the camera frame with respect to frame shape, location, and the dwelling time.

$K-$**frame/**$k-$**camera Problem**

Another natural extension of the problem is what if we can take more frames. These frames can be either taken from the same camera from the same view point or simultaneously taken from multiple cameras. We name the former as $k-$frame problem and the later as $k-$camera problem. These problems is analogy to the $P-$center problem in facility location problem. The initial conjecture is that $k-$frame problem is NP-complete. We need to work on effective approximation algorithms and bounds. The difficulty of the $k-$camera problem also depends on if the cameras are homogenous or not and if the camera views overlap or not. If the k cameras have no overlap in viewable region, this problem can be reduced to k single frame selection problems. To attack $k-$frame and $k-$camera problems, we may also need to generalize our metrics.

1-Telescope Frame Selection Problem

Telescope is also an application that we are interested in applying our Collaborative Teleoperation ideas. First of all, high resolution telescopes are very valuable and scarce devices and need to be shared by multiple researchers. In the astronomy settings, the requests of telescope come in format of point coordinates. The output is the k frames of the telescope. Since there is no zoom adjustment in telescope, this looks like a simplified version of the k-frame problem. An intuitive solution is just to select k disjoint frames to cover as much point requests as possible.

However, we are not just to maximize the number of requests that covered by k frames. There are several facts complicate the problem. First of all, since the Earth is rotating, the time window that we can cover a point request depends on its location. Secondly, we may be able to cover a point request at different time. But the quality of the image depends on when we take the image. It depends on how far the light from the star travels inside the Earth atmosphere. The quality of the image is the best when the telescope points to straight upright direction. How to make sure we get high quality images for high priority requests is interesting, too. The third reason is that we need to consider telescope travelling time and its dome travelling time in planning. The last reason is that due to the changing weather condition, we need to keep in mind that an global optimal solution may not be the optimal under the uncertainty.

k-Telescope Frame Selection Problem

Similar to the $k-$camera problem, we may also have more than one telescope around the world. Coordinating these telescopes is also an interesting problem. Depending on goal of the coordination, different frame selection strategies should be used. For example, if the goal is to avoid repetitive observations, then we should partition the space with respect to telescope location and observation conditions. If the goal is to perform a global unstopped tracking on certain event happening in outer space, we may need to behavior completely different when we coordinate telescopes. These telescopes come from different locations with different parameters, which may also complicate the problem.

7.2.3 Another Viewpoint on Future Work

A CT system includes both human side and machine side. In human side, we need to design systems that motivate people to collaborate to improve group decision quality. Incentive compatibility between individual participants and the group performance is an important issue in system design. In machine side, we need to develop innovative strategies that make good use of the machine to improve reward, productivity, and usage. In our Tele-Actor system, we tried to address the human side problem. In our Co-Opticon system, we tried to address the machine side problem. We are also interested in how to combine both human side and machine side in consideration in new CT system: How to design an incentive compatible system for human operators and also optimize the robot utilization at the same time? We are developping a new system, which is built on the Co-Opticon system, will try to apply those strategies.

7.3 Conclusions

There are very exciting rich set of problems in the Collaborative Teleoperation field. From a school bus to the robot on the Mars, we share robotic resource quite often in our world. In robotics research, we spend a lot time on making the robotic device function properly but do not pay enough attention on how to efficiently share control of the device. In our practices on CT system, we feel that there is a clear call on efficiency and human incentive compatibility, which may not be unique to us and means more attention is needed for those issues in future robotics research.

Bibliography

[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] H. Arai, T. Takubo, Y. Hayashibara, and K Tanie. Human-robot cooperative manipulation using a virtual nonholonomic constraint. In *IEEE International Conference on Robotics and Automation*, April 2000.

[3] R. Arkin. Cooperation without Communication: Multiagent Schema-based Robot Navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[4] R.J. Aumann and J.H. Dreze. Cooperative Games with Coalition Structures. *International Journal of Game Theory*, 3:217–237, 1974.

[5] R.J. Aumann and M. Maschler. The Bargaining Set of Cooperative Games. pages 443–476, 1964.

[6] R. Baecker. *Readings in Groupware and Computer Supported Cooperative Work: Assisting Human-Human Collaboration.* Morgan Kaufmann, 1992.

[7] R. D. Ballard. A last long look at Titanic. *National Geographic*, 170(6), December 1986.

[8] A. Bejczy, G. Bekey, R. Taylor, and S. Rovetta. A Research Methodology for Tele-Surgery with Time Delays. In *First International Symposium on Medical Robotics and Computer Assisted Surgery*, Sept 1994.

[9] A. K. Bejczy. Sensors, Controls, and Man-Machine Interface for Advanced Teleoperation. *Science*, 208(4450), 1980.

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

[11] C. Bernard, H. Kang, S. K. Sigh, and J.T. Wen. Robotic system for collaborative control in minally invaive surgery. *Industrial Robot: An international Journal*, 26(6):476–484, 1999.

[12] B.G. Brooks BG and G. T. McKee. The Visual Acts model for automated camera placement during teleoperation. In *IEEE International International Conference on Robotics and Automation (ICRA 2001), Korea*, 5 2001.

[13] D. M. Blough and G. F.Sullivan. A Comparison of Voting Strategies for Fault-Tolerant Distributed Systems. In *9th Symposium on Reliable Distributed Systems*, 1990.

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

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

[16] A.J. Broder. Strategies for Efficient Incremental Nearest Neighbor Search. *PATREC: Pattern Recognition, Pergamon Press*, 23:171–178, 1990.

[17] A.Z. Broder. On the Resemblance and Containment of Documents. In *Proceedings of Compression and Complexity of SEQUENCES 1997, Positano, Italy, IEEE Comput. Soc.*, pages 21–29, March 1998.

[18] Z. Butler, A. Rizzi, and R. Hollis. Cooperative Coverage of Rectilinear Environments. In *IEEE International Conference on Robotics and Automation*, April 2000.

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

[20] R. Carpenter and L. Carpenter. www.cinematrix.com.

[21] 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.

[22] P. Coppin and W. Whittaker et al. EventScope: Amplifying Human Knowledge and

Experience Via Intelligent Robotic Systems and Information Interaction. In *9th IEEE International Workshop on Robot and Human Interactive Communication (Ro-man)*, 2000.

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

[24] M. Csikszentmihalyi. *Beyond Boredom and Anxiety.* Jossey-Bass Publishers, 1982.

[25] Barney Dalton and Ken Taylor. A Framework for Internet Robotics. In *IEEE International Conference On Intelligent Robots and Systems (IROS): Workshop on Web Robots*, Victoria, Canada, 1998.

[26] D. Desmet, P. Avasare, P. Coene, S. Decneut, F. Hendrickx, T. Marescaux, J-Y. Mignolet, R. Pasko, P. Schaumont, and D. Verkest. *Design of Cam-E-leon, a run-time reconfigurable Web camera.* Springer-Verlag, Berlin, Germany, 2002.

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

[28] B. Donald, L. Gariepy, and D. Rus. Distributed manipulation of multiple objects using ropes. In *IEEE International Conference on Robotics and Automation*, April 2000.

[29] M. Lemaitre E. Bensana and G. Verfaillie. Benchmark Problems : Earth Observation Satellite Management. *CONSTRAINTS: An International Journal*, (3):293–299, 1999.

[30] I. Elhaji, J. Tan, N. Xi, W. Fung, Y. Liu, T. Kaga, Y. Hasegawa, and T. Fukuda. Multi-site Internet-based Cooperative Control of Robotic Operations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[31] I. Elhajj, J. Tan, N. Xi, W.K. Fung, Y.H. Liu, T. Kaga, Y. Hasegawa, and T. Fukuda. Multi-site Internet-based tele-cooperation. *Integrated Computer-Aided Engineering*, 9(2):117–27, 2002.

[32] D. Eppstein. Fast Construciton of Planar Two-Centers. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 131–138, January 1997.

[33] I. Erev and A. E. Roth. Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria. *American Economic Review*, 88(4):848–81, September 1998.

[34] H. Kuzuoka et al. GestureMan: A Mobile Robot that Embodies a Remote Instructor's Actions. In *ACM Conference on Computer Supported Collaborative Learning*, 2000.

[35] R. Fisher, P. Vanouse, R. Dannenberg, and J. Christensen. Audience Interactivity: A Case Study in Three Perspectives. 1996.

[36] T. Fong, C. Thorpe, and C. Baur. Collaboration, Dialogue, and Human-robot interation. In *10th International Symposium of Robotics Research*, 2001.

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

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

[39] L. Gasser. Multi-Agent Systems Infrastructure Definitions, Needs, and Prospects. *Proceedings of the Workshop on Scalable MAS Infrastructure, Barcelona Spain*, 2000.

[40] J. P. Gee. High Score Education. *Wired*, 11(5), May 2003.

[41] Matthew Gertz, David Stewart, and Pradeep Khosla. A Human-Machine Interface for Distributed Virtual Laboratories. *IEEE Robotics and Automation Magazine*, December 1994.

[42] Raymond Goertz and R. Thompson. Electronically Controlled Manipulator. *Nucleonics*, 1954.

[43] E. Goffman. *Strategic Interaction.* University of Pennsylvania Press, 1969.

[44] 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.

[45] 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.

[46] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and Jeff Wiegley. Beyond the Web: Manipulating the Physical World via the WWW. *Computer Net-*

works and ISDN Systems Journal, 28(1), December 1995. Archives can be viewed at http://www.usc.edu/dept/raiders/.

[47] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and Jeff Wiegley. Robot Teleoperation via WWW. In *International Conference on Robotics and Automation*. IEEE, May 1995.

[48] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2), July 2001.

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

[50] 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.

[51] 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.

[52] 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.

[53] R. Grossi and G. F. Italiano. Revised version of “Efficient Cross-Trees for External Memory”. Technical Report TR-00-16, Oct. 2000.

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

[55] D. Halperin, M. Sharir, and K. Goldberg. The 2-Center Problem with Obstacles. *Journal of Algorithms*, 32:109–134, January 2002.

[56] W.R. Hamel, P. Murray, and R.L. Kress. Internet-based robotics and remote systems in hazardous environments: review and projections. *Advanced Robotics*, 16(5):399–413, 2002.

[57] K. Han, Y. Kim, J. Kim, and S.Hsia. Internet control of personal robot between KAIST and UC Davis. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[58] 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.

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

[60] W. Hartmann and B. Rollett. Positive interaction in the elementary school. pages 195–202, 1994.

[61] **http://heatex.mit.edu/**.

[62] **http://telerobot.mech.uwa.edu.au/**.

[63] H. Hu, L. Yu, P. W. Tsui, and Q. Zhou. Internet-based Robotic Systems for Teleoperation. *Assemby Automation*, 21(2):143–151, May 2001.

[64] J-Y. Mignolet J-Y, S. Vernalde, and M. Engels. A low-power reconfigurable Internet appliance camera. *Revue Hf. Electronique, Telecommunications*, (4):4–11, 2000.

[65] A. Jain, P. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on PAMI*, 22(1):4–37, 2000.

[66] S. Jia, Y. Hada, G. Ye, and K. Takase. Distributed telecare robotic systems using CORBA as a communication architecture. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

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

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

[69] T. Kaga, S. Nojima, E. Nanma, M. Tamura, and H. Nakagane. Internet camera system. *Matsushita Technical Journal*, 44(5):66–73, 1998.

[70] J. Kim, B. Choi, S. Park, K.Kim, and S. Ko. Remote control system using real-time MPEG-4 streaming technology for mobile robot. In *IEEE International Conference on Consumer Electronics*, 2002.

[71] 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.

[72] B. Kreimeier. Rising from the Ranks:Rating for Multiplayer Games. http://www.gamasutra.com/features/20000209/kreimeier_pfv.htm.

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

[74] Y. W. Leung. Maximum Likelhood Voting for Fault-Tolerant Software with Finite Output Space. *IEEE Transactions on Reliability*, 44(3), 1995.

[75] 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 *(CBS 2002), the IEEE Computer Society Bioinformatics Conference*, 2000.

[76] 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.

[77] R.C. Luo and T. M. Chen. Development of a Multibehavior-Based Mobile Robot for Remote Supervisory Control Through the Internet. *IEEE/ASME Transactions on Mechatronics*, 5(4):376–385, 2000.

[78] K. Lynch and C. Liu. Designing Motion Guides for Ergonomic Collaborative Manipulation. In *IEEE International Conference on Robotics and Automation*, April 2000.

[79] S. Mann. Wearable computing: A first step toward personal imaging. *computer*, 30, 1997.

[80] S. Mann. Humanistic Intelligence: WearComp as a new framework for Intelligent Signal Processing. *IEEE Proceedings*, 86(11):2123–2151, November 1998.

[81] R. Marimon and E. McGrattan. On Adaptive Learning in Strategic Games. 1995.

[82] R. Marin, P.J. Sanz, and J.S. Sanchez. Object recognition and incremental learning algorithms for a web-based telerobotic system. In *IEEE International Conference on Consumer Electronics*, 2002.

[83] R. Marin, P.J. Sanz, and J.S. Sanchez. A very high level interface to teleoperate a robot via Web including augmented reality. In *IEEE International Conference on Consumer Electronics*, 2002.

[84] A. Mas-Collell. An Equivalence Theorem for a Bargaining Set. *Journal of Math and Economics*, (18):129–139, 1989.

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

[86] N. Megiddo and K.J. Supowit. On the Complexity of Some Common Geometric Location Problems. *SIAM Journal on Computing*, 13:182–196, February 1984.

[87] D. G. Meyers, editor. *Social Psychology.* McGraw-Hill, 1996.

[88] T. Mirfakhrai and S. Payandeh. A delay prediction approach for teleoperation over the Internet. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[89] J. Mitchell and B. Borchers. A comparison of branch and bound and outer approximation methods for 0-1 MINLPs. *Computers and Operations Research*, 24:699–701, 1997.

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

[91] R. S. Mosher. Industrial Manipulators. *Scientific American*, 211(4), 1964.

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

[93] L. Ngai, W.S. Newman, and V. Liberatore. An experiment in Internet-based, human-assisted robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[94] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Dover Publications, Inc. NY, 1998.

[95] E. Paulos and J. Canny. Ubiquitous Tele-embodiment: Applications and Implications. *International Journal of Human-Computer Studies/Knowledge Acquisition*, 1997. Special Issue on Innovative Applications of the World Wide Web.

[96] E. Paulos and J. Canny. Designing Personal Tele-embodiment. In *IEEE International Conference on Robotics and Automation (ICRA)*, Detroit, MI, 1999.

[97] M. Perry and D. Agarwal. Remote control for videoconferencing. In *Proceedings of 2000 Information Resources Management Association International Conference. Anchorage, AK, USA*, 2000.

[98] P. Pirjanian. *Multiple Objective Action Selection and Behavior Fusion Using Voting.* PhD thesis, Aalbord University, 1998.

[99] P. Pirjanian and M. Mataric. Multi-Robot Target Acquisition Using Multiple Objective Behavior Coordination. In *IEEE International Conference on Robotics and Automation*, April 2000.

[100] C. Pollak and H. Hutter. A webcam as recording device for light microscopes. *Journal of Computer-Assisted Microscopy*, 10(4):179–83, 1998.

[101] S. Rasmussen and N.L. Johnson. Self-Organization in and around the Internet. In *Proceedings of 6th Santa Fe Chaos in Manufacturing Conference*, April 1998.

[102] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot collaboration for robust exploration. In *IEEE International Conference on Robotics and Automation*, April 2000.

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

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

[105] T. Sato, J. Ichikawa, M. Mitsuishi, and Y. Hatamura. A New Micro-Teleoperation System Employing a Hand-Held Force Feedback Pencil. In *International Conference on Robotics and Automation.* IEEE, May 1994.

[106] D. Schmid, B. Maule, and I. Roth. Performance teletests for industrial robots by the Internet. In *First IFAC-Conference on Telematics Applications in Automation and Robotics TA 2001. Weingarten, Germany. IFAC*, 7 2001.

[107] P. Shenoy. On Coalition Formation: A Game-Theoretical Approach. *International Journal of Game Theory*, 8:133–164, 1979.

[108] Thomas B. Sheridan. *Telerobotics, Automation, and Human Supervisory Control.* MIT Press, 1992.

[109] R. Siegwart and P. Saucy. Interacting mobile robots on the web. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[110] 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.

[111] D. Song, A. Pashkevich, and K. Goldberg. ShareCam Part II: Approximate and Distributed Algorithms for a Collaboratively Controlled Robotic Webcam. In *IEEE/RSJ International Conference on Intelligent Robots (IROS)*, Nov. 2003.

[112] D. Song, A. F. van der Stappen, and K. Goldberg. Exact and Distributed Algorithms for Collaborative Camera Control. In *The Workshop on Algorithmic Foundations of Robotics*, Dec. 2002.

[113] B. Sutton-Smith. *The Ambiguity of Play.* Harvard University Press, 1997.

[114] N. Tesla. Method of and Apparatus for Controlling Mechanism of Moving Vessels or Vehicles. *http://www.pbs.org/tesla/res/613809.html*, 1898.

[115] R. Tomovic. On Man-Machine Control. *Automatica*, 5, 1969.

[116] G. T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.

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

[118] R. C. Veltkamp and M. Hagedoorn. Shape Similarity Measures, Properties, and Constructions. In *Advances in Visual Information Systems, 4th International Conference, VISUAL 2000, Lyon, France, November 2-4, 2000, Proceedings VISUAL*, volume 1929, pages 467–476. Springer, 2000.

[119] D. Verkest, D. Desmet, P. Avasare, P. Coene, S. Decneut, F. Hendrickx, T. Marescaux,

J-Y. Mignolet, R. Pasko, and P. Schaumont. Design of a secure, intelligent, and reconfigurable Web cam using a C based system design flow. In *Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, IEEE, Piscataway, NJ, USA*, volume 1, pages 463–467, 2001.

[120] J. von Neumann and O. Morgenstern. *The Theory of Games and Economic Behavior.* Princeton University Press, 1953.

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

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

[123] Weixiong Zhang. Depth-First Branch-and-Bound versus Local Search: A Case Study. In *AAAI/IAAI*, pages 930–935, 2000.

[124] X. Zhang, N. Navab, and S-P Liou. E-commerce direct marketing using augmented reality. In *Proceedings of International Conference on Multimedia and Expo. New York, NY, USA.*, 7 2000.