Auton Robot (2008) 24: 435–449
DOI 10.1007/s10514-008-9089-4

Systems, control models, and codec for collaborative observation of remote environments with an autonomous networked robotic camera

Dezhen Song · Ni Qin · Ken Goldberg

Received: 4 July 2007 / Accepted: 24 January 2008 / Published online: 14 February 2008

Abstract Networked robotic cameras are becoming popular in remote observation applications such as natural observation, surveillance, and distance learning. Equipped with a high optical zoom lens and agile pan-tilt mechanisms, a networked robotic camera can cover a large region with various resolutions. The optimal selection of camera control parameters for competing observation requests and the on-demand delivery of video content for various spatiotemporal queries are two challenges in the design of such autonomous systems. For camera control, we introduce memoryless and temporal frame selection models that effectively enable collaborative control of the camera based on the competing inputs from *in-situ* sensors and users. For content delivery, we design a patch-based motion panorama representation and coding/decoding algorithms (codec) to allow efficient storage and computation. We present system architecture, frame selection models, user interface, and codec algorithms. We have implemented the system and extensively tested our design in real world applications including natural observation, public surveillance, distance learning, and building construction monitoring. Experiment results show that our frame selection models are robust and effective and our on-demand content delivery codec can satisfy a variety of spatiotemporal queries efficiently in terms of computation time communications bandwidth.

Keywords Frame selection · Panorama video · Networked cameras · Tele-robot

1 Introduction

Consider a high-resolution pan-tilt-zoom camera installed in a deep forest. Connected to the Internet through a long-range wireless network, the robotic camera allows scientists and/or the general public to observe nature remotely. Equipped with robotic pan-tilt actuation mechanisms and a high-zoom lens, the camera can cover a large region with very high spatial resolution and allows for observation at a distance without disturbing animals. For example, a Panasonic HCM 280A pan-tilt-zoom camera has a 22× motorized optical zoom, a 350° pan range, and a 120° tilt range. It can reach a spatial resolution of 500 megapixel per steradian at its highest zoom level. The full coverage of the viewable region is more than 3 giga-pixels if represented as a motion panorama.

Unlike a conventional webcam with a fixed lens, the combination of the robotic actuation and the networking capability for a networked robotic camera enables collaborative observation: allowing a group of sensors or users to simultaneously share control of the robotic camera. There are two new challenges in the collaborative observation:

This work was supported in part by the National Science Foundation under IIS-0643298, IIS-0534848/0535218, by Panasonic Research, by Intel Corporation, and by UC Berkeley's Center for Information Technology Research in the Interest of Society (CITRIS), and the Berkeley Center for New Media (BCNM).

D. Song (✉) · N. Qin
Computer Science Dept., Texas A&M University,
College Station, TX 77843, USA
e-mail: dzsong@cs.tamu.edu

N. Qin
e-mail: nqin@cs.tamu.edu

K. Goldberg
Depts. of IEOR and EECS, University of California, Berkeley,
CA 94720, USA
e-mail: goldberg@berkeley.edu

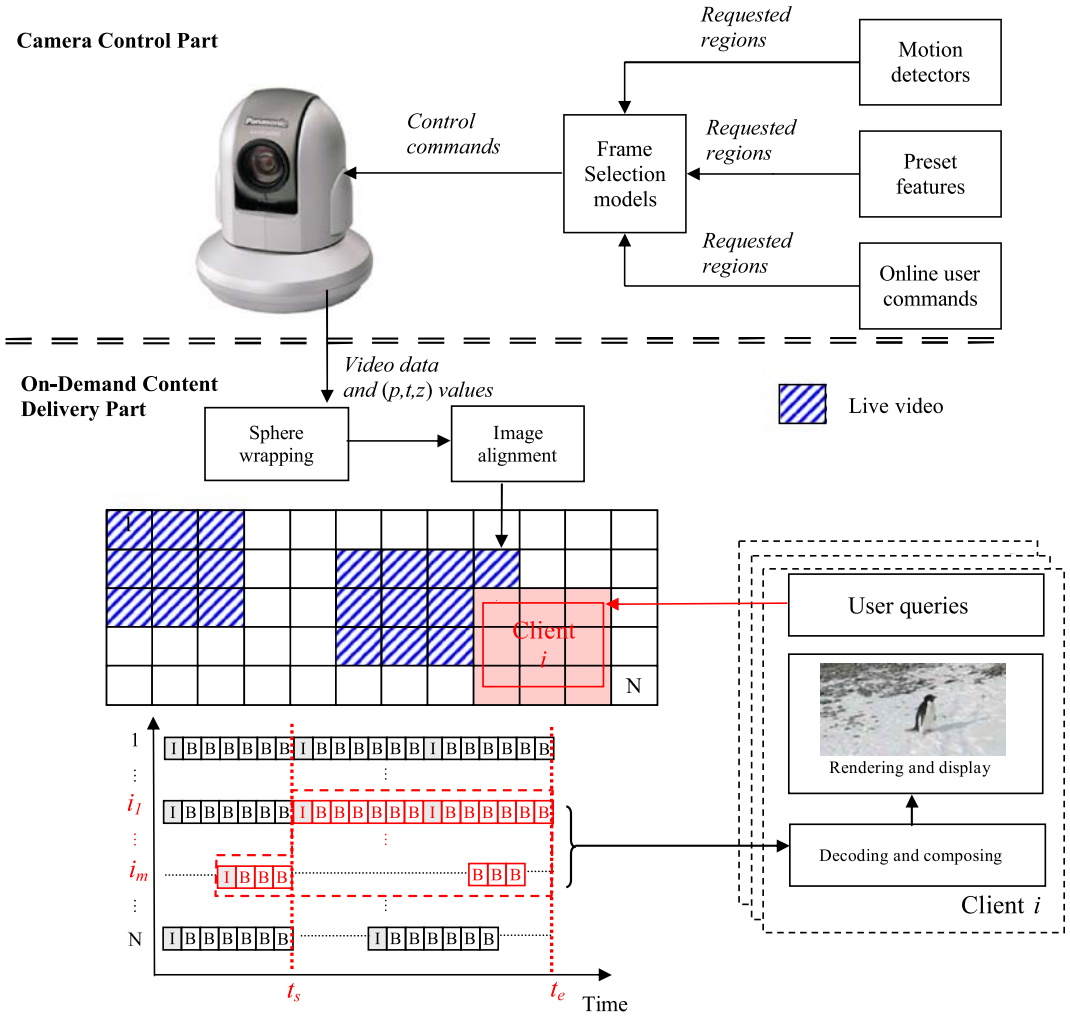


Fig. 1 System diagram. The system includes two parts divided by the *dashed line*. The *upper part* is camera control part. Requests from *in-situ* sensors such as motion detectors, preset features, and online users are fed into the frame selection models to generate camera control commands. The *lower part* is on-demand content delivery part. The *grid* in the figure represents a patch-based high-resolution panorama video system that allows multiple users to query different part of video concurrently. I's and B's indicate the I-frame and the B-frame used in MPEG-2 compression

- *Camera control problem:* Where should the camera be aimed? As illustrated in Fig. 1, the camera could have multiple simultaneous inputs including requests from *in-situ* sensors such as motion detectors, requests from predefined events such as tracking moving objects, and direct inputs from online users. It is necessary to design an effective frame selection model to coordinate this collaborative observation.
- *Content delivery problem:* How should spatio-temporal video segments be stored and delivered? There could be many concurrent online users who want to access the camera. Transmitting the full-sized ever-changing gigapixel panorama video to every user is unnecessary and expensive in terms of the bandwidth requirement. Each user usually wants to observe a different sub-region and time window of the panorama video. For example, an ornithologist may need to study video streams captured early in the morning and above the tree line where birds are most likely to be flying.

As illustrated in Fig. 1, we address the two problems above by developing new systems, new camera control models, and new data representation/codec algorithms. For camera control problem, we propose memoryless and temporal frame selection models that can effectively combine multiple concurrent user and sensor inputs with the consideration of various authorities and emergency levels. The frame selection models capture the camera frame selection as a spatiotemporal resource allocation optimization problem with different weighting functions in its objective functions that can be computed efficiently.

For content delivery problem, we notice that both camera coverage and user queries have spatiotemporal constraints.

To efficiently organize the frames captured by the camera and satisfy various and concurrent user queries is the key to the problem. An analogy to this problem is the Google Earth[1] system where each user requests a view of different regions of the planet Earth. While the image-sharing aspect of Google Earth is similar to our system, the primary difference is that the satellite image database of the Google Earth system is relatively static and user queries do not involve the time dimension whereas our system has to be run in near real time and satisfy spatiotemporal queries. We propose a patch-based approach in a spherical coordinate system to organize data captured by cameras at the server end. Built on an existing video-streaming protocol, the patch-based approach allows efficient on-demand transmission of the queried content.

We have implemented the system and tested in both in-lab and field experiments. The five-year field tests in a variety of applications have shown that the new developments are effective and efficient in dealing with real world applications such as natural observation, building construction monitoring, distance learning, and surveillance. The rest of paper is organized as follows. After presenting the related work, we discuss the overall system design and user interface. Based on the system, we introduce our frame selection models and the on-demand content delivery part before reporting experiment results. We begin with the related work.

2 Related work

Our system builds on the existing work of networked teleoperation (Goldberg and Siegwart 2002) and panorama video systems (Benosman and Kang 2001).

Our system is designed to allow multiple online users to share access to robotic cameras. In the taxonomy proposed by Chong et al. (2000), these are Multiple Operator Single Robot (MOSR) systems or Multiple Operator Multiple Robot (MOMR) systems. An Internet-based MOSR system is described by McDonald, Cannon, and their colleagues (Cannon 1992; McDonald et al. 1997). In their work, several users assist in waste cleanup using Point-and-Direct (PAD) commands. Users point to cleanup locations in a shared image and a robot excavates each location in turn. Recent developments in MOSR systems can be found in (Goldberg and Chen 2001; Goldberg et al. 2003). In (Goldberg et al. 2003) Goldberg et al. propose the "Spatial Dynamic Voting" (SDV) interface. SDV collects, displays, and analyzes sets of spatial votes from multiple online operators at their Internet browsers using a Gaussian point clustering algorithm developed to guide the motion of a remote human "Tele-Actor". Existing work on MOSR and MOMR systems provides strategies to efficiently coordinate the control of the shared robot. Users are usually forced to share the same feedback from the robot. However, users may not be interested in the same event at the same time even when they access the system at the same time. This becomes more obvious when the shared robot is a robotic camera. Time and space of interests may vary for different online users. This paper is aimed to address this new problem.

Our work is directly related to panorama video systems because a panorama is a natural data representation to visualize data from pan-tilt-zoom cameras. Because of its capability to visualize a large region with high resolution, a panorama video system finds many applications such as videoconferencing (Foote and Kimber 2000), distance learning (Foote and Kimber 2001), remote environment visualization (Benosman and Kang 2001), and natural environment observation (Song and Goldberg 2005). The related work can be classified into two categories: panorama video generation and panorama video delivery.

There are many methods to generate a panorama video. A panorama can be generated using a single fixed camera with a wide-angle lens or parabolic mirrors (Baker and Nayar 1999; Nayar 1997; Xiong and Turkowski 1997; Ng et al. 2005). However, due to the fact that it can not distribute pixels evenly in the space and the resolution limitation imposed by the CCD sensor, it cannot generate high-quality video. A panorama video can also be generated by aligning videos from multiple cameras (Swaminathan and Nayar 2000; Foote and Kimber 2001). Although the design can provide complete coverage with live video streams, those systems require simultaneous transmission of multiple video streams and the bandwidth requirement is very high. Panorama video can also be built from registering a pre-recorded sequence of video frames (Irani et al. 1996; Trucco et al. 2000; Zhu et al. 1999; Agarwala et al. 2005) captured by a single rotating camera. However, only portions of the panorama contain live video data at any moment. Our system fits into this category as well. Argarwala et al.'s panoramic video texture (PVT) (Agarwala et al. 2005) and Rav-Acha et al.'s dynamosaics (Rav-Acha et al. 2005) are representative work in this category that constructs pseudo-live panorama video out of a single video sequence by alternating time-space correspondence. Bartoli et al. (2004) develop motion panoramas that extract moving objects first and then overlay the motion part on top of a static background panorama. We summarize the existing panoramic video systems in Table 1. In existing systems, a panorama video is always transmitted and fully reconstructed at the user end because panorama resolution is not a concern. However, when the resolution of the panorama is very high, on-demand transmission is necessary. Our development is the first system that tackles this problem.

Due to its flexibility in coverage and resolution, pan-tilt-zoom cameras have been widely used in surveillance applications. Our system can be viewed as a special case of

[1] http://earth.google.com

Table 1 A comparison of existing panoramic video systems

System	Camera	Band-width	Video output	Sample systems
Wide angle lens / mirrors	Single fixed	Low	Low quality live stream	Baker and Nayar 1999; Nayar 1997; Xiong and Turkowski 1997; Ng et al. 2005
Multiple camera panorama video	Multiple fixed	High	Live panoramic video	Swaminathan and Nayar 2000; Foote and Kimber 2001
Panoramic video texture	Single pan	High	Pseudo-live panorama video by changing video temporal display	Agarwala et al. 2005
Dynamosaics	Single pan	High	Pseudo-live panorama video by changing space-time volume	Rav-Acha et al. 2005
Motion panorama	Single	Low	Static panorama background overlaid with live moving objects trajectory	Irani et al. 1996; Bartoli et al. 2004
Our system	Pan-Tilt-Zoom	Low	Partial live panorama	This paper

the automated surveillance system proposed by Collins et al. (2000). They summarize the exist research in three categories including detection/tracking, human motion analysis, and activity analysis. our work complements the existing work by focusing on camera motion planning and content delivery using inputs from multiple users and sensors. As pointed out by Kansal et al. (2006), networked pan-tilt-zoom cameras can provide very high spatial resolution through controlled camera motion. Camera motion control in (Kansal et al. 2006) was primarily driven by detected activities in the scene and only concerns single inputs. Our work assumes known concurrent activities that have been detected by either online users or *in-situ* sensors and focuses on camera view selection instead.

Transmitting a panorama video is non-trivial. For a low resolution panorama video system, we can encode the whole panorama video and send it to clients. However it consumes too much bandwidth when the resolution of the panorama increases. Furthermore, it cannot deal with random spatiotemporal accesses. Irani et al. (Irani et al. 1995; Irani and Anandan 1998) propose mosaic-based compression. A static panorama background is first constructed out of the video sequence and then each video frame is compressed using the static panorama background as a reference. Furthermore, it detects and indexes the motion objects and provides content-based video indexing. Although they do not deal with on-demand transmission, their work inspires our paper. Ng et al. (2005) propose to partition the panorama into six vertical slices spatially and compress each sliced video sequence separately using MPEG-2. When a user requests the video of a part of the panorama video, only sliced video sequences that intersect with the user's requested area are transmitted. This method is among the first to consider on-demand queries.

Our work complements Ng et al.'s work from two aspects. First, Ng et al.'s work designs for multiple-camera systems that have omni-directional coverage. Due to their full coverage, there is no discontinuity spatiotemporally in the coverage of input image data. Our work is the first design for pan-tilt-zoom robotic camera, where the input image data coverage are usually not continuous in time for a given location. Due to this discontinuity and the random access nature of user queries, our system actually has three different types of outputs: (1) static panorama, (2) archived video, and (3) live video whereas Ng et al.'s design only has video data as output. To directly apply Ng et al.'s design into a PTZ camera will results in repeatedly encoding the data from the previous patches when there is no live coverage and dramatically reduce the efficiency. Second, Ng et al.'s work design divides the panoramic video into vertical tiles. This works great for cylindrical panorama, which is usually the case for multi-camera panorama video system. As shown in our new experiment results, this is not very efficient for a PTZ camera that has large tilt displacement. Its efficiency of encoding decreases as the camera tilt range increases. Furthermore, our work is the first to study how the different patch design would impact system performance such as bandwidth requirement and compression time.

Our lab has developed various algorithms for pan-tilt-zoom cameras. In (Song et al. 2003, 2006a), we develop

shared camera control algorithms to deal with competing requests from online users. We address the image alignment problem in a spherical coordinate system in (Qin et al. 2006). We also address the camera calibration problem in natural environments where steady calibration objects are not available (Song et al. 2006b). In this paper, we build on our system development in networked camera system in the past five years and propose systematic solutions for the camera control problem and the content delivery problem in networked robotic cameras.

3 System architecture

Figure 2 illustrates our system architecture. Any user with Internet connection can access our system. Users log on to our system and send their queries to a server. The server directly connects to the camera. The connection is most likely via a long distance wireless connection if the camera is deployed in a remote area. The system is not limited to a single camera. Multiple cameras can be used to increase coverage as long as their image frames can be projected into the same spherical panorama. Here we use a single camera to illustrate the idea. Since the camera cannot provide the concurrent coverage of the entire viewable region due to its limited field of view and the limited number of pixels in its CCD sensor, a selective coverage is needed. In our system, the camera parameters are controlled by the frame selection model which accepts inputs from preprogrammed patrolling sequence, *in-situ* sensors, and/or user requests as illustrated in Fig. 1.

The user interface consists of two parts: a static background panorama that covers the user query region and a video segment superimposed on top of the background panorama if there are video data collected for the queried time duration. Therefore, depending on user queries and camera configurations, the server may transmit different contents to a user such as a pre-stored video segment, a high-resolution static image with the timestamp closest to the request time window, or a live video from the camera.

On the other hand, users might use low-power devices such as PDAs or cell phones, which do not have the computation power to perform expensive image alignment and panorama construction computation. The server should perform as much computation in generating and delivering panorama video as possible. The server employs our evolving panorama to accomplish the task.

3.1 Evolving panorama

As illustrated in Fig. 3, we propose an *evolving panorama* model to deal with spatiotemporal camera frame inputs and user queries. The evolving panorama is not a panorama but a collection of individual frames with timestamped registration parameters. The registration parameters allow the frame to be registered as part of a virtual spherical panorama.

A panorama is usually constructed by projecting frames taken at different camera configurations into a common coordinate system, which is referred to as a composite panorama coordinate system. We choose a spherical coordinate system as the composite panorama coordinate system due to its relative small distortion if compared to a planar panorama composite coordinate system and large tilt coverage if compared to a cylindrical panorama composite coordinate system. In (Qin et al. 2006), we have shown that image alignment on a same spherical surface can be performed very efficiently because there exist projection invariants to allow the quick computation of registration parameters. Using a pre-calibrated camera, a point $q = (u, v)^T$ in a newly-arrived video frame F is projected to the point $\tilde{q} = (\tilde{u}, \tilde{v})^T$ in $\tilde{F}$ in the spherical coordinate system. The spherical coordinate system is centered at the lens optical center and has

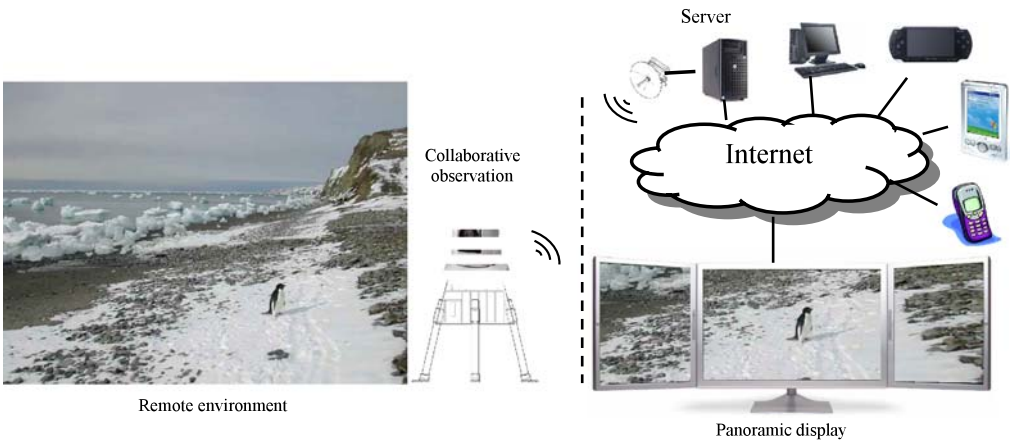


Fig. 2 System architecture and user interface

Fig. 3 Evolving panorama

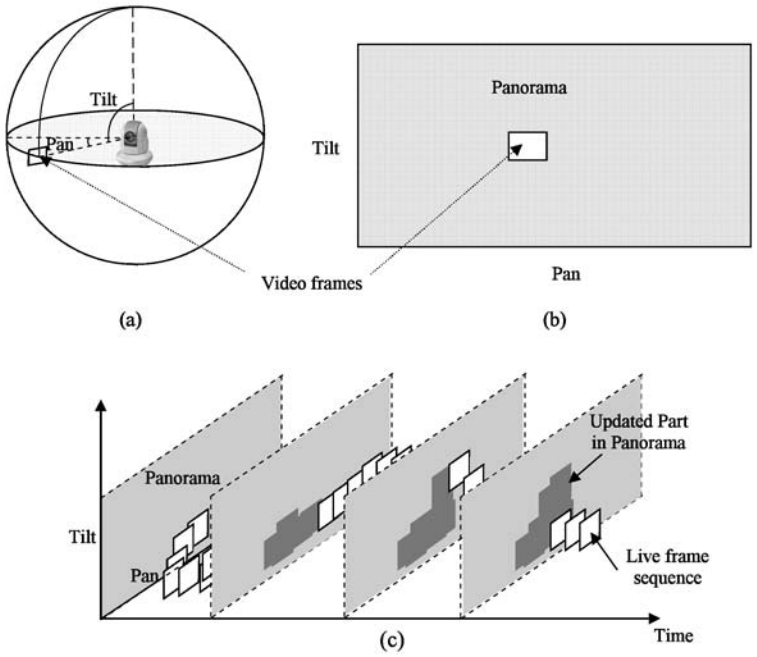
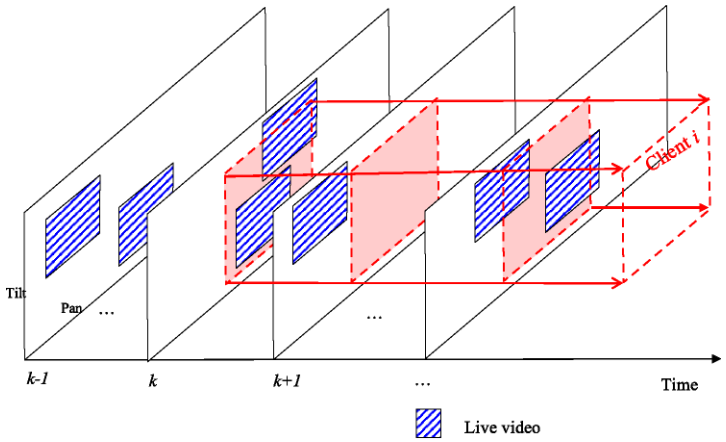


Fig. 4 The relationship between the evolving panorama and a user query. The *striped regions* indicate how the evolving panorama updates as camera frames arrive. The *shaded box* indicates the part of the data the user queries



its radius equal to focal length f of the lens. The spherical pre-projection that projects q to $\tilde{q}$ is,

$$\tilde{u} = \arctan\left(\frac{u}{f}\right), \tag{1a}$$

$$\tilde{v} = -\arctan\left(\frac{v}{\sqrt{u^2 + f^2}}\right). \tag{1b}$$

Each point $(\tilde{u}, \tilde{v})^T$ in $\tilde{F}$ is defined using local pan and tilt spherical coordinates with units of radians. This is a local spherical coordinate because it forces the camera's optical axis to overlap with vector $(\tilde{u} = 0, \tilde{v} = 0)$. The next step is to re-project the local spherical coordinate to a global spherical coordinate to obtain image registration parameters using image alignment. The concept of an evolving panorama builds on the fact that the panorama is continuously updated by the incoming camera frames as illustrated in Fig. 3(c). In fact, we do not store and build the whole panorama in order to avoid expensive computation.

Different users might have different spatiotemporal queries. We also need to understand the relationship between the evolving panorama and user queries.

3.2 Understanding user queries

For a high resolution panorama video, it is impractical to transmit the entire video sequence due to bandwidth limitations. The screen resolution of the display device also limits the resolution of the video. Additionally, a user might not be interested in the entire viewable region. As illustrated in Fig. 4, a typical user query can be viewed as a 3D rectangular query box in space and time. Define q_i as the ith query,

$$q_i = [u, v, w, h, t_s, t_e], \tag{2}$$

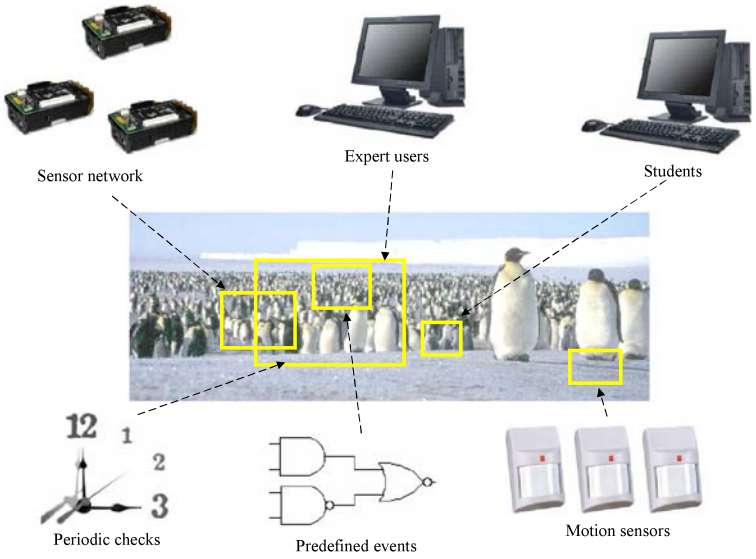


Fig. 5 Requested regions for a robotic camera in natural observation

where (u, v) defines the center position of the query rectangle on the panorama, w and h are width and height of the rectangle, and time interval $[t_s, t_e]$ defines the time window of the request. Figure 4 only illustrates a single user query. At any time k, there may be many different concurrent queries. Addressing the need of different and concurrent queries in content-delivery is one of the requirements for our system. When time interval $[t_s, t_e]$ of a user query intersects with current time, it means the user requests the right to control the camera and hence the user generates a requested region in the camera viewable region.

3.3 Understanding requested regions

At any given time, the camera might receive multiple concurrent requests from different sources as illustrated in Fig. 5. Each requested region is characterized by a rectangle

$$r_i = [u_i, v_i, w_i, h_i, z_i, \omega_i], \tag{3}$$

where variables u_i, v_i, w_i, and h_i define the rectangle in the same ways as u, v, w and h in (2), z_i refers to the desirable resolution, and ω_i is the weight associated with the ith requested region. Weight ω_i indicates that requests might have different emergency/importance levels determined by the interests of observation and user authority levels. For example, a request triggered by a motion detector is usually more important than a regular patrolling behavior in security surveillance. An expert in natural observation is usually granted with higher weight by the system that of a student. Since there is only one camera and usually more than one request, we need to choose camera configurations for the competing requests. Frame selection models are designed to serve this purpose.

4 Frame selection models

In our collaborative observation 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 the resolution of the frame, which corresponds to frame size and zoom level due to the fixed number of pixels in a camera CCD sensor. c defines a rectangular camera frame (the camera has a fixed aspect ratio of 4:3). User i requests a desired frame r_i as defined in (3). Given requests from n users, the system computes a single global frame c^* that will best satisfy the set of requests.

We define a Coverage-Resolution Ratio (CRR) metric for request "satisfaction" $s(c, r_i)$ based on how the request r_i compares with a candidate camera frame c,

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

The term $\frac{\text{Area}(r_i \cap c)}{\text{Area}(r_i)}$ characterizes coverage ratio. The resolution ratio $\min(z_i/z, 1)$ reaches its maximum of 1 if the candidate frame resolution is better than the requested resolution: $z \leq z_i$. Here we abuse the set intersection operator $\cap$ and $r_i \cap c$ refers to the overlapping region between r_i and c. Each of n users submits a request. Let

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

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.

An alternative frame selection model is based on the history of requests over multiple motion cycles. We extend (4) using a weighted sum of the request satisfaction. In this case total satisfaction is a function of time t:

$$s(c,t) = \sum_{i=1}^{n} \alpha_i(t) s_i(r_i(t), c(t)) \tag{5}$$

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

$$\alpha_i(t) = \sum_{k=0}^{t-1} \frac{u_i(k)}{2^{t-1-k}}, \tag{6}$$

which yields the recursive formulation:

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

If request i does not get satisfied by the camera frame computed during the current frame, its weight $\alpha_i(t)$, will increase over future motion cycles, eventually dominating the weights of other requests to satisfy its desired frame request. In this sense fairness is guaranteed over time. We name this frame selection model as *temporal frame selection* model.

Depending on camera characteristics and scene characteristics, different weighting functions and constraints can be used in (6) and in the optimization problem.

For example, if the servo speed of the camera is slow, then the camera traveling time should be considered. Due to motion blurring, we do not want keep images taken during travel. Hence we want to reduce camera travel time. One possible solution is to weight each request by the traveling time. Then (5) can be modified as,

$$\alpha_i(t) = \frac{1}{\|c_i(t) - c(t-1)\|_2}, \tag{7}$$

where, $\|\cdot\|_2$ is the L_2 distance, $c(t-1)$ is the current camera position which is generated at the last period $t-1$, and $c_i(t) = \arg\max_c s_i(r_i(t), c)$ is the camera frame satisfying $r_i(t)$ if request $r_i(t)$ is the only request to the camera. Weight function $\alpha_i(t)$ is inversely proportional to the traveling distance $\|c_i(t) - c(t)\|_2$. This extension will create a "lazy frame selection" model that do not like to change the camera setting significantly. This weighting is not used in our tests, because the Panasonic HCM 280 camera rotates at 120 degrees per second. The traveling time usually a fraction of a second is negligible if compared with the time for refocusing, the time for re-adjusting iris, and the time for stabilizing the camera, which usually takes more than a second.

The existing frame selection model allows partial coverage of user requests. This works for applications such as observation and surveillance. However, complete coverage of the object would be required for some applications. For example, a partial coverage of the penguin in Fig. 5 would be no value to the request. Thus, we need to augment our optimization problem by introducing request integrity constraints such as $\text{Area}(r_i \cap c) \in \{\emptyset, r_i\}$ for served users.

These frame optimization problems can be solved with exact algorithms (Song et al. 2006a) or fast approximation algorithms (Song et al. 2003; Song and Goldberg 2007). The frame selection models address the camera control problem. However, user interests are not necessarily to be focused on the current camera frame. Delivering contents to users on-demand is the second challenge. The on-demand delivery builds on server-side data representation: how the evolving panorama in Fig. 3 is encoded.

5 On-demand content delivery

We propose a patch-based panorama video data representation. This data representation allows us to partition the image space and allows partial update and partial retrieval. Built on the data representation, we then present a frame insertion algorithm and a user query algorithm. To illustrate the idea, we build our algorithms based on the MPEG-2 streaming protocol, which is the most popular protocol that can be decoded by a majority of client devices. However, the design can be easily extended to more recent protocols such as the MPEG-4 family for better compression and quality.

5.1 Patch-based evolving panorama video representation

We partition the panorama video into patches and encode each patch individually using MPEG-2 algorithms. Figure 6 shows a snapshot of the patch-based panorama at time k.

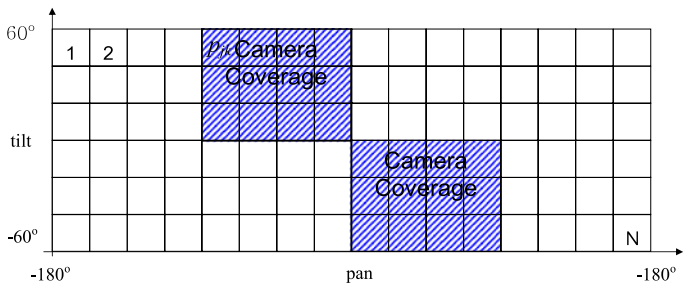


Fig. 6 A snapshot of the evolving panorama video at time k. The striped regions are current camera coverage. Note: this figure assumes there are two cameras

Only a subset of patches contain live video data because cameras cannot provide full coverage of the entire viewable region at a high-zoom setting. The panorama snapshot is a mixture of live patches and static patches. Let us define the jth patch as p_j, $j = 1, \ldots, N$ for a total of N patches. Each patch contains a set of video data $p_j = \{p_{jk} \mid k = 1, \ldots, \infty\}$ across the time dimension. Define F_k as the camera coverage in the viewable region at time k, which refers to the striped region in Fig. 6. If p_j intersects with F_k, p_{jk} contains live video data at time k. Otherwise, p_{jk} is empty and does not need to be stored. To summarize this, the whole patch-based evolving panorama video $\mathbb{P}_t$ at time t is a collection of live patches p_{jk}s,

$$\mathbb{P}_t = \{p_{jk} \mid j = 1, \ldots, N,\ k = 1, \ldots, t,\ p_{jk} \cap F_k \neq \emptyset\}. \quad (8)$$

5.2 Frame insertion algorithm

When a new video frame F_t arrives at time t, we need to update $\mathbb{P}_{t-1}$ to get $\mathbb{P}_t$,

$$\mathbb{P}_t = \mathbb{P}_{t-1} \cup \{p_{jt} \mid j \in \{1, \ldots, N\}, p_{jt} \cap F_t \neq \emptyset\}. \quad (9)$$

Implementing (9) on the server is nontrivial. As illustrated in Fig. 1, for raw video frame F_t, its extrinsic camera parameters are first estimated by aligning with previous frames. The alignment process is performed on the spherical surface coordinate system. Next, we project the frame F_t onto the composite panorama spherical coordinate system. For each patch p_j intersecting with F_t, we encode it individually. We use an MPEG-2 encoder for patch encoding in our implementation. As with any MPEG-2 encoders, the size boundary for the number of frames inside one group of pictures (GOP) is predefined. Each GOP contains one I frame and the rest of the frames are either P frames or B frames. The size of the GOP should not be too large for quick random temporal video retrieval. Each patch holds its own GOP buffer. If the patch p_j intersects the current frame F_t, the updated patch data are inserted into patch video sequence P_j's GOP buffer. Whenever the GOP buffer reaches its size limit, we encode it using the standard MPEG-2. Since only a partial area of the panorama contains live video data at a certain time range and the number of the frames inside the GOP is predefined, the patch video data p_{jk} inside one patch video segment are not necessarily continuous in the time dimension. We summarize the patch-based evolving panorama video encoding Algorithm 1.

5.3 User query algorithm

At time t, the system receives the ith user query $q_i = [u, v, w, h, t_s, t_e]$. To satisfy the request, we need to send the following data to the user at time t,

$$q_i \cap \mathbb{P}_t = \{p_{jk} \mid j \in \{1, \ldots, N\}, k \in [t_s, t_e], p_{jk} \cap q_i \neq \emptyset, p_{jk} \neq \emptyset\}. \quad (10)$$

Remark 1 It is worth mentioning that q_i is different from r_i defined in (3). r_i refers to the user request to current camera configuration which only involves his/her desired camera control commands. Therefore, it does not have length in time dimension. However, q_i is user i's query to existing panoramic video database. There is a time window associated with q_i. q_i can be used as r_i only if the time window of q_i contains current time t, $t \in [t_s, t_e]$.

We implement this query as follows: for each p_j we keep track of its start position and the timestamp of I frames in a lookup table, which is used for random spatiotemporal video access. After receiving r_i, the streaming server first locates the nearest I frame with respect to t_s and t_e. If the streaming server identifies there is no live data in patch p_j in the requested time range, no additional video data is transmitted for patch p_j. This procedure can be summarized as Algorithm 2.

The decoding procedure at the client side is the standard MPEG-2 decoding. It is worth mentioning that the output of the system is not always a video segment. As illustrated in Fig. 4, a user-requested region does not overlap with camera coverage at time $k + 1$. It is possible that a user request

Algorithm 1: Frame insertion algorithm

```
input  : F_t
output: Updated evolving panorama video
wrap F_t onto the spherical surface;
estimate F_t's registration parameters by aligning it
with previous frames;
project F_t onto the sphere panorama surface;
for each p_j and p_j ∩ F_t ≠ ∅ do
    insert p_jt into p_j's GOP buffer;
for each p_j, j = 1, ..., N do
    if p_j's GOP buffer is full then
        encode patch video segment;
        store patch video segment start position and
        time data into lookup table;
        reset GOP buffer for incoming data;
```

Algorithm 2: User query algorithm

```
input  : r_i
output: r_i ∩ ℙ in MPEG-2 format
Identify patch set
S = {p_j | j ∈ {1, ..., N}, p_j ∩ r_i ≠ ∅};
for each p_j ∈ S do
    find the nearest I frame p_jb earlier or equal to t_s;
    find the nearest I frame p_jc later or equal to t_e;
    transmit the patch segments between p_jb and p_jc;
```

might not intersect with any camera frames for the entire query time window $[t_s, t_e]$. For this situation, this algorithm will output an I-frame that is closest to $[t_s, t_e]$. Therefore, it sends a static image closest to the request. If the user request happens to be overlapped with current live camera coverage, the user receives live video. This algorithm allows three types of outputs: a pre-stored video, a live video, and a static image.

6 Experiments and results

The system has been tested extensively during its 5-year development. We start the experiment with in-lab tests and then deploy the system for a variety of applications such as public surveillance, building construction monitoring, and natural observation. To ensure the robustness, we have tested both frame selection models and on-demand content delivery using different hardware configurations.

6.1 Hardware

We have tested our system using three types of Pan-Tilt-Zoom cameras as illustrated in Fig. 7. Table 2 lists the specifications of the three cameras. Among those parameters, pan range, tilt range, and lens Horizontal Field Of View (HFOV) determine the overall coverage of the panorama. Image resolution, size of CCD sensor, and focus length are used to establish coordinate projection model between the image coordinate system and the composite panorama coordinate system.

6.2 Experiment with frame section models

Figure 8 shows four examples with the memoryless frame selection model. Note that the optimal frame grows in image (b) after a large requested region is added. In Fig. 8(c), two more requested regions entered the system. Since they can not compete with the central group of requested regions, the optimal frame remains unchanged. Figure 8(d) shows a case with all but two requested regions disjoint, the algorithm selects a frame that covers the two overlapping requested regions. Figure 8 also illustrates that some requests can be starved indefinitely.

(a) Canon VCC3 (b) Canon VCC4 (c) Panasonic HCM 280

Fig. 7 The pan-tilt-zoom cameras tested with our system

Figure 9 shows four examples with the temporal frame selection model, where frame selection is based on request satisfaction over multiple motion cycles. A sequence of 4 motion cycles is illustrated with the same set of requested regions. Note that with this model, the camera frame changes to balance overall satisfaction over time.

During the in-lab test for frame selection models, the system went online in June of 2002 with the camera installed in Alpha Lab, UC Berkeley from June 8, 2002 to February 2003 as shown in the previous figures. An illustration of the total requested frames is shown in Fig. 10.

Figure 10(a) displays all 4822 requested regions from online users 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. $0 \le g(x, y) \le 255$, $r_j : 1 \le j \le 4822$ be the jth requested regions, 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 $g_{max} = \arg\max_{(x,y)} g(x, y)$,

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

We compute $g(x, y)$ for each point in the panorama and generate Fig. 10(b). As shown in the figure, the most popular region is the center of the camera workspace, looking at the Adept robot arm in the lab, where one our colleague was often performing robot calibration tests.

6.3 Experiment with on-demand content delivery

The server used in the test is a Dell Dimension DX with a 3.2 GHz Pentium dual-core processor and 2 GB RAM. The video camera is a Panasonic HCM 280a. We have implemented our algorithms using Visual C++ in Microsoft Visual

Table 2 A comparison of technical specifications of the 3 Pan-Tilt-Zoom cameras tested in our system

Camera	Pan	Tilt	Zoom	Focus length	HFOV
VCC3	−90°–(+90°)	−30°–(+25°)	10×	4.2–42 mm	4°–46°
VCC4	−100°–(+100°)	−30°–(+90°)	16×	4–64 mm	3°–47.5°
HCM 280	−175°–(+175°)	0°–(−120°)	21×	3.8–79.8 mm	2.6°–51°

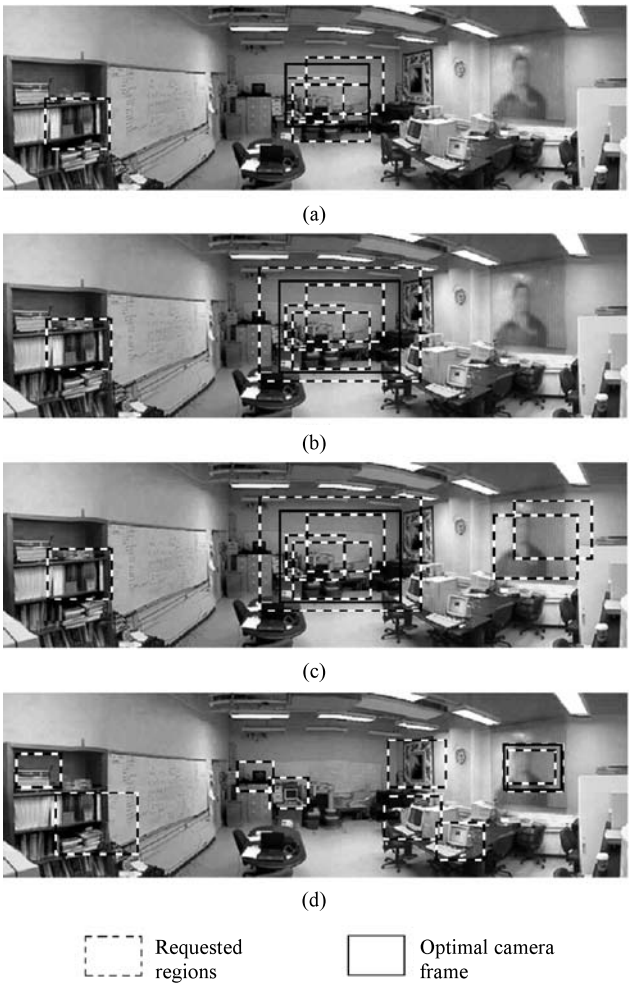


Fig. 8 Examples using memoryless frame selection model defined by (4). Four different sets of requested regions and the corresponding optimal frame are displayed. Note that the resulting frame is very different than what would be determined by simple averaging, and that some requests never get satisfied

Studio 2003.NET and adopted the MPEG-2 encoder and decoder source code developed by the MPEG Software Simulation Group.

We have conducted experiments using data from field tests. As illustrated in Fig. 11, we have deployed our camera in two testing fields including a construction site at UC Berkeley and a pond in Central Park, College Station, Texas. We have collected data at both sites. For the construction site, data cover a duration from Feb. 10, 2005 to Jun. 2, 2005. The camera has been controlled by both online users and a pre-programmed patrolling sequence. Data collected in the park cover the experiment duration of Aug. 24, 2005 to Aug. 31, 2005. The construction site provides an urban environment setting while tests in the park provide a natural environment setting.

The data for each trial consist of 609 image frames captured at a resolution of 640×480. For a frame rate of 25 frames per second, the data represent 24 seconds of recording by the HCM 280a. The overall raw RGB data file size is 536 megabytes for the 24-bit color depth used in the experiment. The constructed panorama has an overall resolution of 2742×909 after cropping the uneven edges. The panorama size is much smaller than what the camera can provide (i.e. giga-pixel level). Since our tests involve speed tests, a large image file will involve an excessive mixture of RAM and disk operations, which could bias the speed test results. Using a smaller data set can minimize disk-seeking operations and reveal the real difference in computation speed.

In the first test, we are interested in testing how much storage savings we can gain from the design and how much computation time is needed to achieve the gain. During all the tests, we set the MPEG-2 quantization level to 50 without a rate limit. Therefore, we can compare the size of the video file data at the same video quality at different patch size settings. The last row in Table 3 actually encodes the entire panorama video at once without using patches, which is used as the benchmarking case. In this case, we update and generate a full panorama for each arriving camera frame. Then the full panorama is added into the GOP for encoding. The file size in Table 3 is displayed in units of kilobytes. Smaller file size means less storage and is preferable. It is interesting to see that patch-based approach has significant savings in storage. This is expected because our system does not encode the un-updated part of the panorama

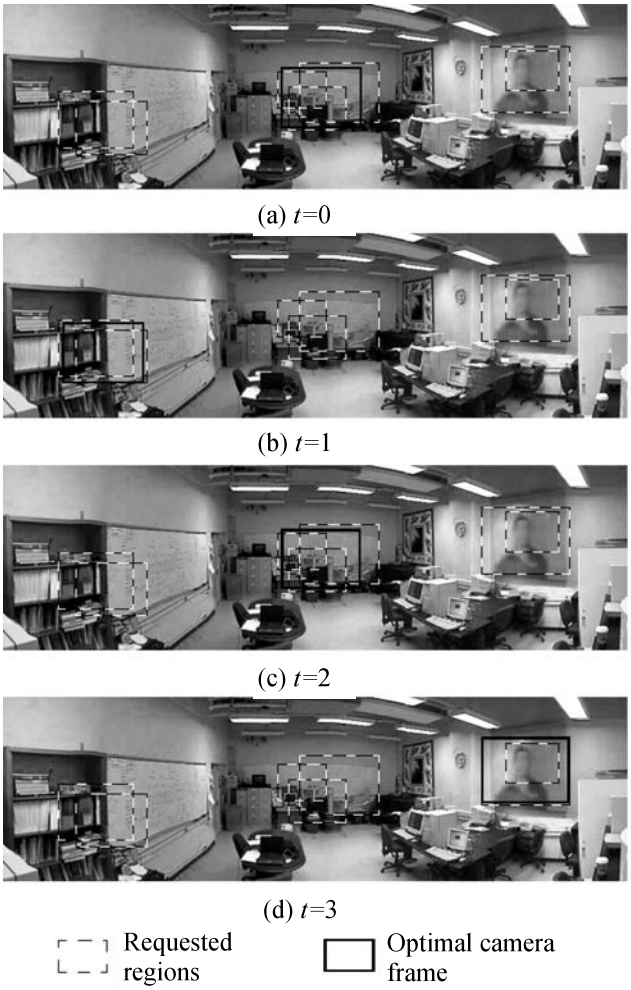


Fig. 9 Examples with the temporal frame selection Model defined by (5). The set of requested region is held constant, but weights evolve so that the camera frame changes to facilitate "fairness"

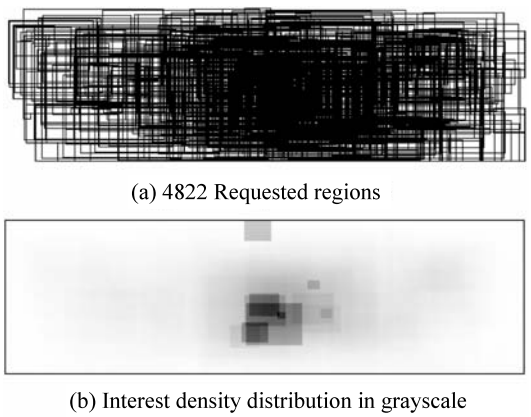


Fig. 10 Data of requested regions from June 8, 2002 to February 6, 2003

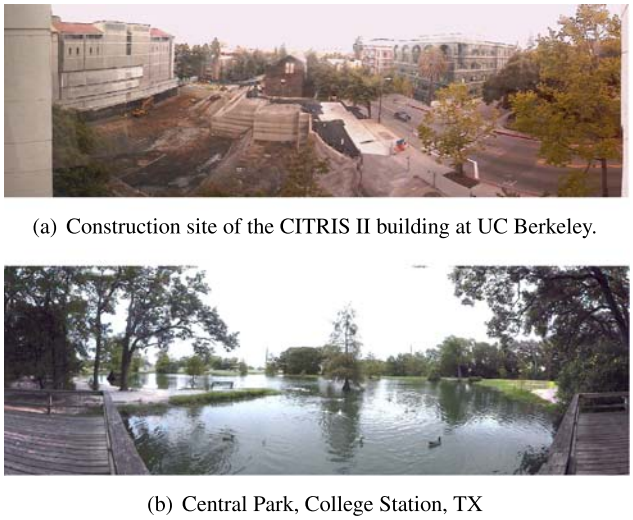


Fig. 11 Experiment sites

Table 3 Storage and computation speed versus different patch sizes. The 6th row divides the panorama into vertical tiles using Ng et al.'s method in (Ng et al. 2005). The 7th row does not divide the panorama video at all which transmits the whole panorama video at once

	Patch size	#Patches	File size (kb)	Speed
1	96×96	290	8044	6.9×
2	128×96	220	8191	6.4×
3	256×192	55	8871	5.0×
4	320×240	36	9965	3.8×
5	480×320	18	11099	3.1×
6	458×909	6	16688	2.4×
7	2742×909	1	22163	1×

as opposed to the benchmarking case which repeatedly encodes the un-updated region. The speed column compares the computation speed under the various patch size settings with the benchmarking case. As shown in Table 3, encoding the entire panorama in the benchmarking case takes more time than that of the patch-based approach. The computation speed gets faster as the patch size reduces. This can be explained by two reasons (1) less data: we do not repeatedly encode the un-updated region and (2) smaller problem space: the block matching problem space is much smaller for a smaller patch size in the MPEG-2 encoding. The 6th row in Table 3 encodes the panorama using the vertical division method in (Ng et al. 2005). Actually, the method in (Ng et al. 2005) encodes the entire panorama because it was designed for multiple-camera systems. To ensure a fair comparison, we improve it using by only coding the updated part of the panorama. Even though it is not as efficient as square pathes.

In the second test, we are interested in studying how much bandwidth is needed for a normal user query. We as-

sume that user has a screen resolution of 800 × 600. Therefore, the request follows the same size. We know that the bandwidth requirement depends on how many patches the request intersects with. We study two cases including the best-case scenario and the worst-case scenario. The best-case scenario refers to the case that the request intersects with the least number of patches. The worst-case scenario is the opposite. Table 4 summarizes the test results. Again, the last row of the table is the benchmarking case that encodes the entire panorama whenever there is a new frame entering the system. The 6th row refers to the modified (Ng et al. 2005) method. As expected, a smaller patch size is preferred because it requires less bandwidth.

6.4 Field tests

Our system has been deployed in 8 different sites for a variety of applications including building construction monitoring, public surveillance, distance learning, and natural observation. Table 5 summarizes the deployment history. Due to space limit, we cannot detail each deployment. We have recorded over 300,000 users in those deployments. The robot cameras have executed more than 52 million commands.

During the field tests, both frame selection models have been tested. We have collected feedback from online users during all the tests. Although inconclusive, it is interesting to see that a majority of users would prefer the memoryless frame selection model if there are less concurrent activities in camera coverage region. For example, users in construction monitoring applications usually observe a single activity over a relatively long period of time and they prefer the memoryless frame selection model. This is due to the fact that memoryless frame selection model tends to observe a fixed location if the requests do not change often. When they are multiple current interesting activities, users tend to be more satisfied with the temporal frame selection model. In applications such as natural observation, users usually prefer to shift attention more frequently in between several current activities. Hence the temporal frame selection model is more desirable. We conjecture that this phenomenon also relates to our design of limiting one request per user. In natural observation cases, the user might want to observe several locations. Although he/she cannot enter more frames, he/she tends to rely on other users to select other interesting locations and hope the system can observe all those locations in long run. However, the memoryless frame model would always select the most popular region and cannot patrol the camera to other interesting locations. Hence, it is understandable that users would prefer the temporal frame selection model in this case.

Table 4 Bandwidth for a user query versus different patch sizes. Same as those in Table 3, the 6th row and the 7th row give results for comparison to Ng et al's method in (Ng et al. 2005) and the naive benchmarking method that transmits the whole panorama video, respectively

	Patch size	Worst case (kbps)	Best case (kbps)
1	96 × 96	739.7	582.5
2	128 × 96	794.3	608.1
3	256 × 192	1344.1	860.2
4	320 × 240	1476.3	830.4
5	480 × 320	1849.8	822.1
6	458 × 909	3708.4	927.1
7	2742 × 909	7387.7	7387.7

7 Conclusion and future work

In this paper, we present frame selection models and on-demand content delivery for collaborative observation with a networked robotic camera. We present system architecture, frame selection models, user interface, and codec algorithms. We introduce memoryless and temporal frame selection models that effectively enable collaborative real time control of the camera based on the competing inputs from *in-situ* sensors and users. We design a patch-based motion panorama representation and coding/decoding algorithms (codec) to allow efficient storage and on-demand content delivery. We have implemented the system and extensively

Table 5 A summary of field tests

	Application	Duration	Location
1	In-lab test	Jun. 2002–Feb. 2003	Alpha Lab, UC Berkeley
2	Construction monitoring	Jun. 2003–Jun. 2005	Stanley Hall Construction site, UC Berkeley
3	Public surveillance	Sep. 2004	Sproul Plaza, UC Berkeley
4	Construction monitoring	Oct. 2004–Aug. 2005	CITRIS II Construction site, UC Berkeley
5	Natural observation	Aug. 2005	Central Park, College Station, TX
6	Marine life observation	Nov. 2005–Jan. 2006	Richardson Bay Audubon Sanctuary, CA
7	Distance learning	May 2006	PolyPEDAL Lab, UC Berkeley
8	Bird watching	Apr. 2007–present	Sutro Forest, CA

tested our design in real world applications including natural observation, public surveillance, and building construction monitoring. Experiments suggest that our frame selection models are efficient and that our on-demand video delivery codec can satisfy a variety of spatiotemporal queries efficiently in terms of computation time communications bandwidth.

In the future, we will extend our frame selection models to multiple cameras. Efficient algorithms will be developed to address the p-camera problem. Camera motion scheduling and path planning problem is also an interesting problem to investigate. There is also an important human-robot interaction/human-computer interaction problem such as how different frame selection models would impact user behavior or system performance. We will work with colleagues in HCI to address those problems using formal user studies. More quantitative analysis will also be conducted regarding this problem. We will also improve and implement our on-demand content delivery with more powerful streaming protocols such as MPEG4 family. Another interesting extension is to allow users to enter knowledge-based description instead of a fixed rectangular input. This is of a great value for many applications because users often do not know where the interesting objects are in the camera coverage region but they do know what objects they are interested in. This will generate a set of new research problems: how to optimize the camera configuration with respect to a set of abstractions.

Acknowledgement Thanks to D. Volz and J. Liu for their insightful discussions. Thanks are given to H. Lee, Y. Xu, C. Kim, B. Green, and H. Wang for their contributions to Networked Robots Lab in Department of Computer Science, Texas A&M University. Thanks also to Jeremy Schiff, Ted Schmidt, Andrew Dahl, and other students in the Automation Sciences Lab at UC Berkeley.

References

Agarwala, A., Zheng, C., Pal, C., Agrawala, M., Cohen, M., Curless, B., Salesin, D., & Szeliski, R. (2005). Panoramic video textures. In *ACM transactions on graphics (proceedings of SIGGRAPH 2005)*, Los Angeles, CA, July 2005.

Baker, S., & Nayar, S. K. (1999). A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, *35*(2), 175–196.

Bartoli, A., Dalal, N., & Horaud, R. (2004). Motion panoramas. *Computer Animation and Virtual Worlds*, *15*, 501–517.

Benosman, R., & Kang, S. B. (2001). *Panoramic vision*. New York: Springer.

Cannon, D. J. (1992). *Point-and-direct telerobotics: object level strategic supervisory control in unstructured interactive human-machine system environments*. PhD thesis, Stanford Mechanical Engineering, June 1992.

Chong, N., Kotoku, T., Ohba, K., Komoriya, K., Matsuhira, N., & Tanie, K. (2000). Remote coordinated controls in multiple telerobot cooperation. In *IEEE international conference on robotics and automation* (Vol. 4, pp. 3138–3343), April 2000.

Collins, R., Lipton, A., & Kanade, T. (2000). Introduction to special section on video surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 745–746.

Foote, J., & Kimber, D. (2000). FlyCam: practical panoramic video and automatic camera control. In *IEEE international conference on multimedia and expo, ICME 2000* (Vol. 3, pp. 1419–1422), New York, NY, July 2000.

Foote, J., & Kimber, D. (2001). Enhancing distance learning with panoramic video. In *Proceedings of the 34th Hawaii international conference on system sciences* (pp. 1–7).

Goldberg, K., & Chen, B. (2001). Collaborative control of robot motion: robustness to error. In *International conference on intelligent robots and systems (IROS)* (Vol. 2, pp. 655–660), October 2001.

Goldberg, K., & Siegwart, R. (Eds.). (2002). *Beyond webcams: an introduction to online robots*. Cambridge: MIT Press.

Goldberg, K., Song, D., & Levandowski, A. (2003). Collaborative teleoperation using networked spatial dynamic voting. *Proceedings of the IEEE*, *91*(3), 430–439.

Irani, M., & Anandan, P. (1998). Video indexing based on mosaic representations. *Proceedings of the IEEE*, *86*, 905–921.

Irani, M., Hsu, S., & Anandan, P. (1995). Video compression using mosaic representations. *Signal Processing: Image Communication*, *7*, 529–552.

Irani, M., Anandan, P., Bergen, J., Kumar, R., & Hsu, S. (1996). Efficient representations of video sequences and their applications. *Signal Processing: Image Communication*, *8*.

Kansal, A., Kaiser, W., Pottie, G., Srivastava, M., & Sukhatme, G. (2006). Virtual high-resolution for sensor networks. In *ACM SenSys*, Boulder, CO, November 1–3, 2006.

McDonald, M., Small, D., Graves, C., & Cannon, D. (1997). Virtual collaborative control to improve intelligent robotic system efficiency and quality. In *IEEE international conference on robotics and automation* (Vol. 1, pp. 418–424), April 1997.

Nayar, S. K. (1997). Catadioptric omnidirectional camera. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition* (pp. 482–488), San Juan, Puerto Rico.

Ng, K. T., Chan, S. C., & Shum, H. Y. (2005). Data compression and transmission aspects of panoramic videos. *IEEE Transactions on Circuits and Systems for Video Technology*, *15*(1), 82–95.

Qin, N., Song, D., & Goldberg, K. (2006). Aligning windows of live video from an imprecise pan-tilt-zoom robotic camera into a remote panoramic display. In *IEEE international conference on robotics and automation (ICRA)*, Orlando, FL, May 2006.

Rav-Acha, A., Pritch, Y., Lischinski, D., & Peleg, S. (2005). Dynamosaics: video mosaics with non-chronological time. In *IEEE computer society conference on computer vision and pattern recognition*.

Song, D., & Goldberg, K. (2005). Networked robotic cameras for collaborative observation of natural environments. In *The 12th international symposium of robotics research (ISRR)*, San Francisco, CA, October 2005.

Song, D., & Goldberg, K. (2007). Approximate algorithms for a collaboratively controlled robotic camera. *IEEE Transactions on Robotics*, *23*(5), 1061–1070.

Song, D., Pashkevich, A., & Goldberg, K. (2003). ShareCam part II: approximate and distributed algorithms for a collaboratively controlled robotic webcam. In *IEEE/RSJ international conference on intelligent robots (IROS)* (Vol. 2, pp. 1087–1093), Las Vegas, NV, October 2003.

Song, D., van der Stappen, A. F., & Goldberg, K. (2006a). Exact algorithms for single frame selection on multi-axis satellites. *IEEE Transactions on Automation Science and Engineering*, *3*(1), 16–28.

Song, D., Qin, N., & Goldberg, K. (2006b). A minimum variance calibration algorithm for pan-tilt robotic cameras in natural

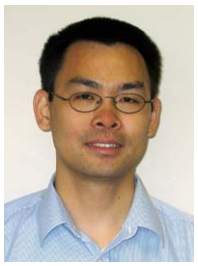
environments. In *IEEE international conference on robotics and automation (ICRA)*, Orlando, FL, May 2006.

Swaminathan, R., & Nayar, S. K. (2000). Nonmetric calibration of wide-angle lenses and polycameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(10), 1172–1178.

Trucco, E., Doull, A., Odone, F., Fusiello, A., & Lane, D. (2000). Dynamic video mosaicing and augmented reality for subsea inspection and monitoring. In *Oceanology international conference*, Brighton, UK, March 2000.

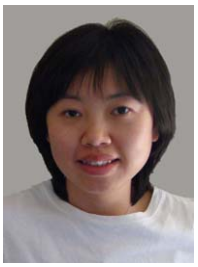
Xiong, Y., & Turkowski, K. (1997). Creating image-based VR using a self-calibrating fisheye lens. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition* (pp. 237–243), San Juan, Puerto Rico, June 1997.

Zhu, Z., Xu, G., Riseman, E. M., & Hanson, A. R. (1999). Fast generation of dynamic and multi-resolution 360-degree panorama from video sequences. In *Proceedings of the IEEE international conference on multimedia computing and systems* (Vol. 1, pp. 9400–9406), Florence, Italy, June 1999.



Dezhen Song is an Assistant Professor with Texas A&M University, College Station, Texas, TX, USA. Song received his Ph.D. in 2004 from University of California, Berkeley.

Song's primary research area is networked robotics, computer vision, optimization, and stochastic modeling. Dr. Song received the Kayamori Best Paper Award of the 2005 IEEE International Conference on Robotics and Automation (with J. Yi and S. Ding). He received NSF Faculty Early Career Development (CAREER) Award in 2007.



Ni Qin is currently a Ph.D. student in the Department of Computer Science at Texas A&M University. Ni Qin received her B.S. degree in Biology from Wuhan University, Wuhan, China in 1997, and M.S. degree in Medical Biochemistry from University of Mississippi Medical Center in 2000, and M.S. degree in Computer Science from Texas A&M University in 2003, respectively. Her research interests are in network tele-operation, multimedia streaming, and computer vision.

Ken Goldberg is Professor of Industrial Engineering and Operations Research at UC Berkeley, with secondary appointments in Electrical Engineering and Computer Science and in the School of Information. He received his Ph.D. in Computer Science from CMU in 1990. He also studied at the Univerisity of Pennsylvania, Edinburgh University, and the Technion. From 1991–1995 he taught at the University of Southern California, and in Fall 2000 was visiting faculty at the MIT Media Lab. Goldberg and his students work in three areas: Algorithmic Automation, Medical Robotics, and Networked Tele-Robotics. He is Founding Chair of the Advisory Board to the IEEE Transactions on Automation Science and Engineering. Goldberg received the National Science Foundation Young Investigator Award in 1994, the NSF Presidential Faculty Fellowship in 1995, the Joseph Engelberger Award for Robotics Education in 2000, and the IEEE Major Educational Innovation Award in 2001. Goldberg is currently VP of Technical Activities for the IEEE Robotics and Automation Society and Director of the Berkeley Center for New Media.