On-Demand Sharing of a High-Resolution Panorama Video from Networked Robotic Cameras

Ni Qin and Dezhen Song

***Abstract*— Due to their flexibility in coverage and resolution, networked robotic cameras become more and more popular in applications such as natural observation, security surveillance, and distance learning. Equipped with a high optical zoom lens, a networked robotic camera can generate a giga-pixel-level panorama to cover its viewable region. As new live frames enter the system, this panorama can be updated as a panorama video. User requests are usually not limited to the current camera frame. A user may request a specific region associated with a specific time window. To satisfy different spatiotemporal requests for multiple concurrent users, we present systems and algorithms to allow the on-demand sharing of the high-resolution panorama video. The high-resolution panorama video is encoded into a patch-based representation to allow efficient storage and on-demand content delivery. We present system architecture, user interface, data representation, and encoding/decoding algorithms. In the experiment, we have implemented the system using the MPEG-2 codec. Experimental results show that our system can not only satisfy different spatiotemporal queries but also significantly reduce computation time and communication bandwidth requirement.**

I. Introduction

Consider a high-resolution pan-tilt-zoom camera installed in a deep forest. Connected to the Internet through a long-range wireless network communication, the robotic camera allows scientists and/or the general public to continuously 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 allow for observation at a distance without disturbing animals. For example, a Panasonic HCM 280A pan-tilt-zoom camera has a 22x motorized optical zoom. The camera can reach a spatial resolution of 500 megapixel per steradian at the highest zoom level. Since the camera has a 350° pan range and a 120° tilt range, the full coverage of the viewable region is more than 3 gigapixels if represented as a panorama. As the camera patrols the viewable region, the giga-pixel panorama is also continuously updated.

As illustrated in Fig. 1, there are many concurrent scientists and other online users who want to access the camera (or cameras if multiple cameras are installed for better coverage). Transmitting the full-sized ever-changing giga-pixel panorama video to every user is unnecessary and expensive in the bandwidth requirement. Each user may want to observe a different sub-region and time window of the panorama video. For example, an ornithologist is often interested in bird video data when the camera is aimed at the top of the forest in the morning.

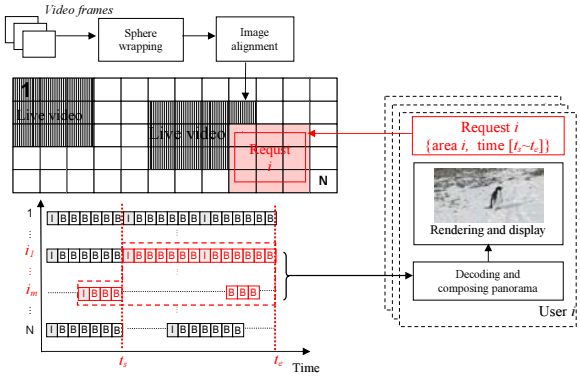

Fig. 1. Evolving panorama video system diagram. The left hand side illustrates the server side. The right hand side is a user at the client side. The grid at server represents a patch-based high-resolution panorama video system that allows multiple users to query different parts of the video concurrently. I's and B's indicate the I-frame and the B-frame used in MPEG-2 compression. A user sends a spatiotemporal request to server side and to retrieve the part of his/her interests in the panorama.

Since that both camera coverage and user requests have spatiotemporal constraints, how to efficiently organize the frames captured by the camera and satisfy various and concurrent user requests becomes a challenging problem. An analogy to this problem is the Google Earth (http://earth.google.com) 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 requests do not involve the time dimension whereas our system has to be run in near real time and satisfies spatiotemporal user requests.

In this paper, we present systems and algorithms that allow on-demand sharing of a high-resolution panorama video. It is the first panorama video system that is designed to efficiently deal with multiple different spatiotemporal requests. 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 request regions. We present a system architecture, user interface, data representation, and encoding/decoding algorithms followed by experiments. We begin with the related work.

This work was supported in part by the National Science Foundation under IIS-0534848/0643298.

N. Qin and D. Song are with Computer Science Department, Texas A&M University, College Station, TX 77843, USA (email: nqin@cs.tamu.edu and dzsong@cs.tamu.edu).

II. Related Work

Our system builds on the existing work of networked teleoperation [1] and panorama video systems [2].

Our system is designed to allow multiple online users to share access to robotic cameras. In the taxonomy proposed by Chong et al. [3], these are Multiple Operator Single Robot (MOSR) systems or Multiple Operator Multiple Robot (MOSR) systems. An Internet-based MOSR system is described by McDonald, Cannon, and their colleagues [4], [5]. In their work, several users assist in waste cleanup using Point-and-Direct (PAD) commands. Users point to cleanup locations in a shared image and a robot excavates each location in turn. In [6] Goldberg et al. propose the "Spatial Dynamic Voting" (SDV) interface for a MOSR system. 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 the 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 [7], distance learning [8], remote environment visualization [2], and natural environment observation [9]. 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 [10]–[13]. However, due to the fact that it can not distribute pixels evenly in the space and the resolution limitation imposed by CCD sensors, it cannot generate high-quality video. A panorama video can also be generated by aligning videos from multiple cameras [8], [14]. Although the design can provide complete coverage with live video streams, those system 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 [15]–[18] 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) [18] and Rav-Acha et al.'s dynamosaics [19] 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. [20] 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 I. 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.

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. [21], [22] 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. [13] propose to partition the panorama into six vertical slices spatially and compress each sliced video sequence separately using MPEG-2. When a user requests for 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. However, its efficiency of encoding decreases as the camera tilt range increases. Also it repeats the data from previous frame when there is no live coverage; it is not efficient or a faithful representation of the remote environment. Our work advances the idea of partitioning panorama into 2-D patches and significantly reduces computation time and bandwidth by only encoding/transmitting updated patches.

Our lab has developed various algorithms for pan-tilt-zoom cameras. In [23], [24], 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 [25]. In this work, we focus on on-demand content delivery for pan-tilt-zoom cameras.

III. System Architecture

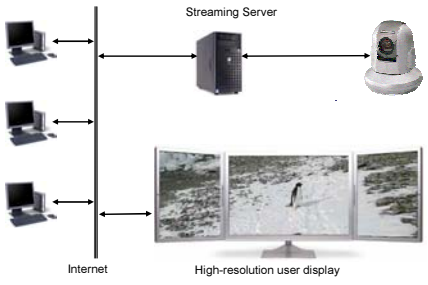

Fig. 2. System architecture and user interface

Fig. 2 illustrates our system architecture. Any user with Internet access can access our system. Users log on to our system and send their requests to a streaming server. The streaming server directly connects to the camera. The system is not limited to a single camera. The development can be easily extended to a multiple-camera system to increase concurrent coverage as long as their image frames can be

System	Camera	Bandwidth	Video Output	Sample Systems
Wide angle lens / mirrors	Single fixed	Low	Low quality live stream	[10]–[13]
Multiple camera panorama video	Multiple fixed	High	Live panoramic video	[8], [14]
Panoramic video texture	Single pan	High	Pseudo-live panorama video by changing video temporal display	[18]
Dynamosaics	Single pan	High	Pseudo-live panorama video by changing space-time volume	[19]
Motion panorama	Single	Low	Static panorama background overlaid with live moving objects trajectory	[15], [20]
Our system	PTZ cameras	Low	Partial live panorama	This paper

TABLE I

A COMPARISON OF EXISTING PANORAMIC VIDEO SYSTEMS.

projected into the same spherical panorama. Here we use a single camera to illustrate the idea. Note that the camera cannot provide concurrent coverage of the entire viewable region due to its limited field of view and the limited number of pixels in its CCD sensor. The motion of the camera can be controlled by preprogrammed patrolling sequence, sensor inputs, or the optimization of frames from competing user inputs [23], [24].

The user interface consists of two parts: a static background panorama that covers the user requested region and a video segment superimposed on top of the background panorama if there are video data collected for the requested time duration. Therefore, depending on user requests and camera configurations, the streaming server may transmit different contents to a user such as a pre-stored video segment, a high-resolution static image with the time stamp 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 streaming server should perform as much computation in generating and delivering panorama video as possible. The streaming server employs our evolving panorama to accomplish the task.

A. Evolving Panorama

An evolving panorama is the data representation we design to deal with spatiotemporal camera frame inputs and user requests. 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 [25], we have shown that image alignment on the 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 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. In fact, we do not store and build the whole panorama in order to avoid expensive computation.

Different clients might have different spatiotemporal requests. It is important to understand the relationship between the evolving panorama and user requests.

B. Understanding User Requests

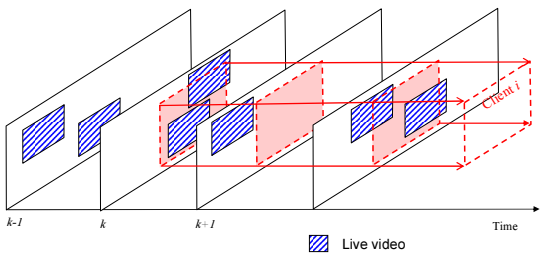

Fig. 3. The relationship between the evolving panorama and a user request. The striped regions indicate how the evolving panorama updates as camera frames arrive. The shaded box indicates the part of the data the user queries.

For a giga-pixel 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. 3, a typical user request can be viewed as a 3D rectangular query box in space and time. Define r_i as the ith request,

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

where (u, v) defines the center position of the requested 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. Fig. 3 only illustrates a single user request. At any time k, there may be many different concurrent requests. Addressing the need of different and concurrent requests is the requirement for our system.

With the concept of the evolving panorama and user requests, we are ready to introduce the data representation and algorithms for the system.

IV. Data Representation and Algorithms

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 performance.

A. Patch-based Evolving Panorama Video Representation

We partition the panorama video into patches and encode each patch individually using MPEG-2 algorithms. The grid in Fig. 1 shows a snapshot of the patch-based panorama at a given time. 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, $j = 1, ..., N$ for a total of N patches. Each patch contains a set of video data $p_j = \{p_{jk} | k = 1, ..., \infty\}$ across the time dimension. Define F_k as the camera coverage in the viewable region at time k. 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,

$$\mathbb{P}_t = \{p_{jk} | j = 1, ..., N, k = 1, ..., t, p_{jk} \cap F_k \neq \emptyset\}. \tag{3}$$

B. Frame Insertion Algorithm

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

$$\mathbb{P}_t = \mathbb{P}_{t-1} \cup \{p_{jt} | j \in \{1, ..., N\}, p_{jt} \cap F_t \neq \emptyset\}. \tag{4}$$

Implementing Equation (4) on the streaming server is non-trivial. 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 below.

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;
```

C. User Query Algorithm

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

$$r_i \cap \mathbb{P}_t = \{p_{jk} | j \in \{1, ..., N\}, k \in [t_s, t_e], \\ p_{jk} \cap r_i \neq \emptyset, p_{jk} \neq \emptyset\}. \tag{5}$$

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 the following algorithm.

Algorithm 2: User Query Algorithm

input : r_i
output: $r_i \cap \mathbb{P}$ in MPEG-2 format
Identify patch set $S = \{p_j | j \in \{1, ..., N\}, p_j \cap r_i \neq \emptyset\}$;
for *each* $p_j \in S$ **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};

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. 3, a user-requested region does not overlap with camera coverage at time $k + 1$. It is possible that a user request 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.

V. Experiments and Results

We test our algorithms using a Dell Dimension DX with a 3.2Ghz Pentium dual-core processor and 2GB RAM. The video camera is a Panasonic HCM 280a. It has a $2.8^\circ - 51^\circ$ horizontal field of view. We have implemented our algorithms using Visual C++ in Microsoft Visual 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 the data from field tests. As illustrated in Fig. 4, 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.

(a) Construction site of the CITRIS II building at UC Berkeley.

(b) Central Park, College Station, TX

Fig. 4. Experiment sites.

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 II 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 (same as [13]). The file size in Table II 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 as opposed to the benchmarking case which repeatedly encodes the un-updated regions. The speed column compares the computation speed under the various patch size settings with the benchmarking case. As shown in the Table II, 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

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

TABLE II
Storage and computation speed versus different patch sizes.

smaller for a smaller patch size in the MPEG-2 encoding.

In the second test, we are interested in studying how much bandwidth is needed for a normal user query. We assume 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. Again, the last row of the table is the bencharking case. Table III summarizes the test results. As expected, a smaller patch size is preferred because it requires less bandwidth.

	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	2742×909	7387.7	7387.7

TABLE III

Bandwidth for a user query versus different patch sizes.

VI. Conclusion and Future Work

In this paper, we present a patch-based panorama video encoding/decoding system that allows multiple online users to share access to pan-tilt-zoom cameras with various spatiotemporal requests. We present the evolving panorama as the data representation. We also present algorithms for efficient frame insertion operations and user query operations. We have implemented the system and conducted field tests. The experiments have shown that our system can significantly reduce the storage needs and bandwidth requirements of online users.

Acknowledge

Thanks to K. Goldberg and D. Volz for his insightful discussions. Thanks are given to H. Lee, Y. Xu, Z. Goodwin, B. Green, and C. Kim for their contributions to Networked Robots Lab in Department of Computer Science, Texas A&M University.

References

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

[2] R. Benosman and S. B. Kang, *Panoramic Vision*. Springer, New York, 2001.

[3] 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*, vol. 4, April 2000, pp. 3138–3343.

[4] D. J. Cannon, "Point-and-direct telerobotics: Object level strategic supervisory control in unstructured interactive human-machine system environments," Ph.D. dissertation, Stanford Mechanical Engineering, June 1992.

[5] 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*, vol. 1, April 1997, pp. 418–424.

[6] K. Goldberg, D. Song, and A. Levandowski, "Collaborative teleoperation using networked spatial dynamic voting," *The Proceedings of The IEEE*, vol. 91, no. 3, pp. 430–439, March 2003.

[7] J. Foote and D. Kimber, "Flycam: Practical panoramic video and automatic camera control," in *IEEE International Conference on Multimedia and Expo. ICME 2000, New York, NY*, vol. 3, July 2000, pp. 1419–1422.

[8] ——, "Enhancing distance learning with panoramic video," in *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001, pp. 1–7.

[9] D. Song and K. Goldberg, "Networked robotic cameras for collaborative observation of natural environments," in *The 12th International Symposium of Robotics Research, (ISRR), San Francisco, CA*, October 2005.

[10] S. Baker and S. K. Nayar, "A theory of single-viewpoint catadioptric image formation," *International Journal of Computer Vision*, vol. 35, no. 2, pp. 175 – 196, November 1999.

[11] S. K. Nayar, "Catadioptric omnidirectional camera," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico*, June 1997, pp. 482–488.

[12] Y. Xiong and K. Turkowski, "Creating image-based vr using a self-calibrating fisheye lens," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico*, June 1997, pp. 237–243.

[13] K. Ng, S. Chan, and H. Shum, "Data compression and transmission aspects of panoramic videos," *IEEE Transactions On Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 82–95, Jan. 2005.

[14] R. Swaminathan and S. K. Nayar, "Nonmetric calibration of wide-angle lenses and polycameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1172–1178, Oct. 2000.

[15] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu, "Efficient representations of video sequences and their applications," *Signal Processing : Image Communication*, vol. 8, Nov 1996.

[16] E. Trucco, A. Doull, F. Odone, A. Fusiello, and D. Lane, "Dynamic video mosaicing and augmented reality for subsea inspection and monitoring," in *Oceanology International Conference, Brighton, UK*, Mar. 2000.

[17] Z. Zhu, G. Xu, E. M. Riseman, and A. R. Hanson, "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, Florence, Italy*, vol. 1, June 1999, pp. 9400–9406.

[18] A. Agarwala, C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski, "Panoramic video textures," in *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005), Los Angeles, CA*, July 2005.

[19] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg, "Dynamosaics: Video mosaics with non-chronological time," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[20] A. Bartoli, N. Dalal, and R. Horaud, "Motion panoramas," *Computer animation and Virtual Worlds*, vol. 15, pp. 501–517, 2004.

[21] M. Irani, S. Hsu, and P. Anandan, "Video compression using mosaic representations," in *Signal Processing: Image Communication*, vol. 7, 1995, pp. 529–552.

[22] M. Irani and P. Anandan, "Video indexing based on mosaic representations," in *Proceedings of the IEEE*, vol. 86, no. 5, May 1998, pp. 905–921.

[23] 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), Las Vegas, NV*, vol. 2, Oct. 2003, pp. 1087 – 1093.

[24] D. Song, A. F. van der Stappen, and K. Goldberg, "Exact algorithms for single frame selection on multi-axis satellites," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no. 1, pp. 16–28, January 2006.

[25] N. Qin, D. Song, and K. Goldberg, "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, Florida*, May 2006.