Automating Inspection and Documentation of Remote Building Construction using a Robotic Camera

Dezhen Song[1], Qiang Hu[1], Ni Qin[1], and Ken Goldberg[2]

1: CS Department, Texas A&M University, College Station, TX 77843
2: IEOR and EECS Department, University of California, Berkeley, CA 94720

Draft date: January 30, 2005

***Abstract*— When constructing buildings, frequent inspection and detailed visual documentation are important but may not be feasible in remote or dangerous environments. We describe a networked robotic camera system that can automatically monitor construction details and allow remote human experts to zoom in on features as construction proceeds to archive the construction process over time, thereby reducing travel cost and human risk. We describe system architecture, interface design, data structures, and algorithms for such systems. We also report initial experimental results from cameras at two outdoor construction sites.**

***Index Terms*— automation, construction, networked robots, pan-tilt-zoom camera, panoramic display.**

I. Introduction

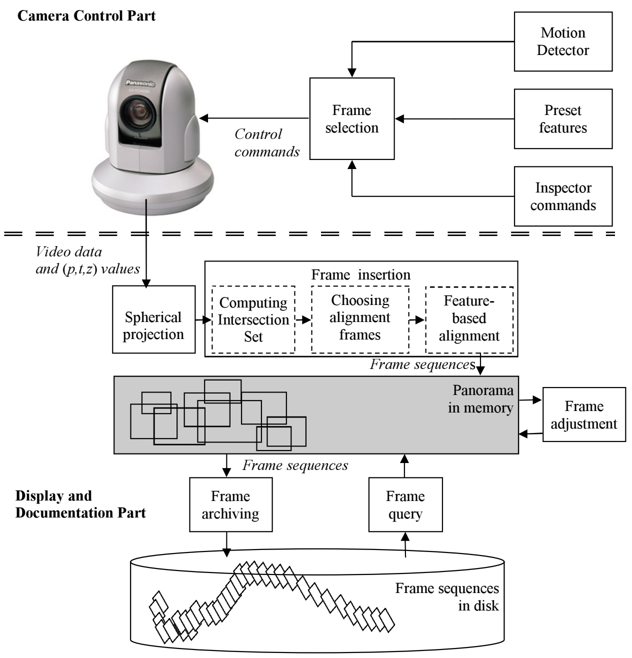
Construction of large buildings and structures such as bridges involves a complex and highly precise sequence of operations. Small errors in alignment, reinforcement, or materials can result in extremely costly repairs or catastrophic failures. Regular inspection and documentation are well-established aspects of construction practice but may not be feasible when construction is performed in remote and dangerous environments. This paper describes a networked robotic camera system that can automatically monitor construction details, allowing human experts to identify key features to track as construction proceeds and a panoramic interface that provides a visual archive of the construction process over time to reduce travel costs and human risk.

Recent developments in wireless telecommunications facilitate easily deployable low-bandwidth connectivity to remote construction sites. A new class of low-cost networked pan-tilt-zoom robotic video cameras allows fast deployment of systems that can provide high resolution images from a wide field of view in the remote environment.

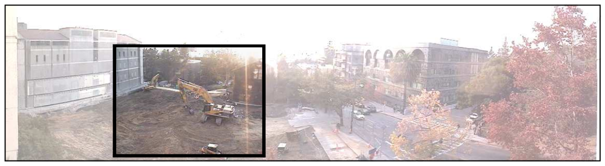
One example is the Panasonic WV-CW864A camera. With 22x zoom motorized optical lens, 360° pan range, and 90° tilt range, this robotic camera can provide resolution up to 500 million pixels per steradian, two orders of magnitude higher than the best available fixed position omnidirectional camera, at a fraction of the cost.

As illustrated in Figure 1, our system automatically steers a networked pan-tilt-zoom camera to inspect and document construction activities. The input is a set of preset image features, human inspector commands, and on-site motion detectors. The resulting "foveal" video images are aligned

This work was supported in part by Texas Engineering Experiment Station startup funds, by the National Science Foundation Grant IIS-0113147, by Intel Corporation under Intel Research Council Grant, by Panasonic, and by UC Berkeley's Center for Information Technology Research in the Interest of Society (CITRIS). For more information please contact dzsong@cs.tamu.edu or goldberg@ieor.berkeley.edu.



(a)



(b)

Fig. 1. *Top Figure (a) illustrates system architecture; camera motion is determined by a combination of preset points, human inspector commands, and motion detector inputs. The resulting video sequences are aligned and inserted into the evolving panorama. Lower Figure (b) illustrates panoramic interface, the inset frame is a sample detail captured by the robotic camera and insertion algorithms.*

and inserted into a coherent panoramic display. Figure 1(b) illustrates the evolving panorama interface.

The evolving panorama can structure and organize the documented video frames, but minor errors in camera position can produce large registration errors. For example, accurate registration of a 640×480 image at zoom = 10x into a panorama requires angular position accuracy within $0.00625°$, 100 times more than the accuracy that is currently available in commercial robotic cameras. Furthermore, if stored naively, the evolving panorama can consume a large amount of memory. For example, the evolving panorama in Figure 1(b) could have a maximum resolution of 28800×9600 at zoom=10x. We describe system architecture, interface design, and the Frame Graph (FG), a new data structure and algorithms that efficiently manage the evolving panorama.

We report experiment results from two testing sites.

II. Related work

1) Multiple-Camera System and Wide Angle System: When low/variable image resolution is acceptable, an evolving panoramic display can be maintained with a single wide-angle camera using a fish eye lens or parabolic mirror [2], [10], [20], [6]. When sufficient bandwidth is available, an evolving high-resolution panorama can be maintained with multiple fixed cameras. Swaminathan and Nayar [16] use four wide angle cameras to monitor a $360°$ field of view. Similarly, Tan, Hua, and Ahuja [17] combine multiple cameras with a mirror pyramid to create a single-perspective and high resolution panoramic video. Liu, Kimber, and Foote [9] combine four fixed cameras with a robotic camera that can selectively zoom in on details. Our approach could be combined with one or more fixed cameras, but since bandwidth is limited, we focus on using only one robotic camera to monitor the environment.

2) Image Mosaicing Techniques: Generating a single wide-field panoramic image from a set of overlapping images is sometimes referred to as “image mosaicing” [3], [12]. Given a set of overlapping images, the objective is to find the best set of transform parameters for each image. Our earlier paper [15] provides a comprehensive overview of three types of approaches including direct method, frequency domain registration, and “feature based” method.

3) Constructing a 3D Scene from Video Frames: Constructing a 3D scene from either calibrated or un-calibrated video frames is a very popular problem in both robotics and computer vision [11], [18]. The similarity between this problem and our problem is that both use overlapping frames to establish transformation matrices. The difference is that 3D modeling requires frames captured from different perspectives whereas panorama construction prefers frames from a single perspective. For two given frames, a 3D model can only be constructed for intersection region of the two frames whereas a panorama generated from our problem covers union region of the two frames.

4) Dynamic Panorama: A dynamic panorama refers to a updateable panorama built from a pre-recorded sequence of consecutive video images [7], [19], [21]. Current methods do not take the image registration error into consideration. Therefore, it either has limited number of frames or relies on extensive frame matching computation which can not process live video data. Hence, the dynamic panorama has to be pre-computed off-line before streaming and has been referred as film-based panorama in [6].

The idea of dynamic panorama also inspires work on developing panorama video streaming protocol. Kim et al [8] develop a panorama video streaming protocol for a pan-tilt camera system. They capture live video using a fixed lens camera and assume camera pan and tilt readings are accurate enough to register frames. They expand MPEG algorithm by slicing camera horizonal field of view into vertical strips and propose inter-strip and intra-strip compression ideas. Their work has not addressed image registration error accumulation problem and do not support spherical panorama.

5) Our Previous Related Work and Contribution: In previously reported work, we developed camera control interfaces for multiple simultaneous tele-operators [13], [14]. In [15], we reported an algorithm to address the registration frame selection problem based on the balance of registration error and computation speed. In this paper, we present a frame sequence management data structure with its supporting algorithms, system architecture, and experiment results from two testing sites.

III. System Architecture and Interface Design

As illustrated in Figure 1(a), our system has two parts: camera control part and display-documentation part. The camera control part accepts three types of commands: preset features, inputs from motion detector, and occasional inspector commands.

The programmable preset features ensure that the camera periodically patrols and searches for interesting regions. It includes two type of camera control commands: fixed locations and particular features. The former are good for a complete coverage of the known and fixed locations of the construction site whereas the later are good for the known and dynamic points of interest. For example, the later can help to track the motion of an excavator.

Sporadic motions are captured by motion detectors, which also generate camera control commands. The motion detectors could be real pyroelectric sensors that are installed in the scene or just a motion detector built on image analysis [5].

Inspectors may also want to control the camera directly from time to time. With the highest priority, the inspector commands can always overrule autonomous commands from preset features and motion detectors. The priority sequence for the three types of commands is also configurable. Weighted by their priorities, commands are feeded into a frame selection module. Using the method in [13], [14], the frame selection module generates a single camera control command based on priority, geometric relationship between different commands, and previous camera visits.

At the same time, the system updates panorama and documents frame sequences. Both video data and camera pan-tilt-zoom values are transmitted to our system. Frame sequences are generated by projecting video frames onto a spherical surface for alignment. The up-to-date part of the evolving panorama is stored in memory for display and the historical part of the evolving panorama is stored in hard disk. Frame insertion, archiving, and adjustment algorithms are developed to manage the frame sequences and will be discussed below.

Figure 1(b) illustrates the interface design. It is usually displayed in high resolution and wide angle monitors. In our settings, we use a pair of 19-inch LCD monitors with a combined resolution of 2560×1024 to visualize the evolving background panorama. With a resolution of 640×480, the live video window is superimposed on top of the evolving panorama. To provide a good spatial reference, the position of the live video window changes as the camera moves around.

IV. Hardware and Software

Fig. 2. *The Pan-Tilt-Zzoom cameras tested with our system.*

We have tested our system using three types of Pan-Tilt-Zoom cameras as illustrated in Figure 2. Table I 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 image coordinate and world coordinate. Maximum speed of the camera determines how fast a camera can travel between different pan-tilt-zoom settings.

Using DirectX 9.0 SDK, our software has been developed for windows-based platform using Microsoft Visual Studio .Net 2003. It runs on Win32 compatible platforms. The testing machining used is a PC with 1Ghz AMD Athlon CPU, 512Mb RAM, and 100Mbs network connection.

V. Algorithms

To build the system, we need to solve both the camera control problem and the panorama-based documentation problem. Since the camera control problem has been addressed in [14] and the incremental frame registration problem has been addressed in [15], we concentrate on how the panorama is managed by Frame Graph, which is a new data structure, and its algorithms including frame insertion, archiving, and adjustment. We begin with inputs and assumptions.

A. Inputs and Assumptions

Since we use the incremental frame registration method introduced in [15], we follow the same assumptions. For completeness of this paper, we brief them here.

1) Definition of Frame Sequence: When the camera is moving, images are blurred and must be discarded. Once the camera has stopped, we define a *frame sequence* as a sequence of camera frames from some fixed pan-tilt-zoom setting,

$$F = \{C, p, t, z, X\}, \tag{1}$$

where C stands for the frame content data set including the beginning time and ending time of the frame sequence, (p, t, z) are the approximate camera pan, tilt, and zoom values obtained from the camera, and X is a set of unknown image alignment parameters.

Since the camera does not move for the duration of a frame sequence, we compute the alignment parameters using the first image of each frame sequence and use the same alignment parameters to transform the last image of the sequence to update the panorama. Below, we refer to the “frame” as the first image from a frame sequence.

2) Definition of an Evolving Panorama: The evolving panorama includes all previous frame sequences inserted in temporal order. Each panorama has a reference frame. The positional parameters X of other frame sequences are computed with respected to the reference frame. The reference frame is also the first frame of the panorama.

3) Known Camera Intrinsic Parameters: Frame resolution, camera focus length, and CCD sensor size are known.

4) Approximate Camera Pan, Tilt, Zoom Position: As noted above, camera pan-tilt-zoom (p, t, z) parameters are inherently approximate because of the accuracy limitation in camera potentiometers.

5) Pair-wise Alignment: Define m_{jl} as the number of overlapping pixels between frame j and frame l, the pair-wise alignment algorithm outputs the relative offset X_{jl} between the two frames. We use an $O(m_{jl})$ time feature-based method, which runs linear to the number of overlapping pixels.

6) Minimum Variance Matching (MVM) Algorithm: When frame j enters the system, it may have many overlapping frames, which define overlapping frame set M_j. Computing its alignment parameters with respect to entire set M_j is computationally expensive and does not necessarily yield a good result. In [15], we present an algorithm to choose $\hat{M}_j \subseteq M_j$, which is an optimal subset of the existing frames to register frame i and minimize its registration error. The MVM algorithm runs at $O(k \log k)$ time for $k = |M_j|$. The MVM algorithm limits the total number of overlapping pixels between $\hat{M}_j$ and frame j to be no more than p pixels.

Results from [15] prove that the optimal alignment parameter X_j of frame j is a weighted sum of of pair-wise alignment results,

$$X_j = \frac{\sum_{l \in \hat{M}_j} \left(m_{jl}(X_l + X_{jl}) \right)}{\sum_{l \in \hat{M}_j} m_{jl}}, \tag{2}$$

where X_l is the known alignment parameter for frame l.

B. Frame Graph

Our evolving panorama is a collection of parameterized frame sequences stored in Frame Graph (FG), which is a variation of planar 2D graph. In an FG, node j contains,

Camera	pan	tilt	zoom	focus length	F-number	max speed	HFOV	CCD
VCC3	$-90^\circ \sim +90^\circ$	$-30^\circ \sim +25^\circ$	10x	$4.2 \sim 42mm$	f/1.8 to 2.9	$70^\circ/sec$	$4^\circ \sim 46^\circ$	1/4in
VCC4	$-100^\circ \sim +100^\circ$	$-30^\circ \sim +90^\circ$	16x	$4 \sim 64mm$	f/1.4 to 2.8	$70^\circ/sec$	$3^\circ \sim 47.5^\circ$	1/4in
HCM 280	$-175^\circ \sim +175^\circ$	$0^\circ \sim -120^\circ$	21x	$3.8 \sim 79.8mm$	f/1.6 to 3.6	$200^\circ/sec$	$2.6^\circ \sim 51^\circ$	1/4in

TABLE I

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

- *node ID j,*
- *frame sequence F_j,*
- *rectangle R_j that describes the image coverage area, and*
- *total number of pixels of image m_j.*

Edge e_{jl} links node j and node l, which contains,

- *edge ID in format of jl,*
- *indicator variable I_{jl} to show if the edge has been used for alignment, where*

$$I_{jl} = \begin{cases} 0 & \textit{no alignment} \\ 1 & \textit{frame j is aligned to frame l} \\ -1 & \textit{frame l is aligned to frame j} \end{cases}$$

- *relative offset X_{jl} between node j and node l if $I_{jl} \neq 0$,*
- *number of overlapping pixels m_{jl}, and*
- *rectangle that describes the overlapped area R_{jl}.*

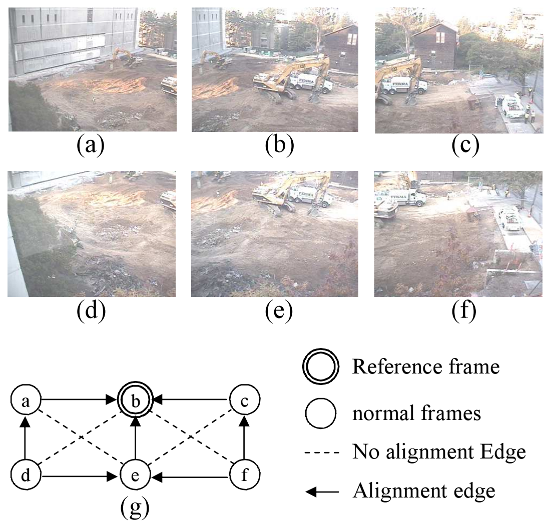


Fig. 3. *An example of Frame Graph with six frames. Figure (a e) are frames and Figure (b) is the corresponding FG.*

Figure 3 illustrates a sample FG with six frame sequences. For frame sequence j, its M_j is just its edge set and $\hat{M}_j$ is just the set of edges that have $I_{jl} = 1$. Alignment edges and nodes formulate a Directional Acyclic Graph (DAG) with its only sink located at the reference frame. As a data structure, FG also has a set of maintenance algorithms including frame insertion, archiving, and adjustment.

C. Frame Insertion Algorithm

Each time after the camera changes its pan-tilt-zoom settings, a new frame sequence will be generated and needs to be inserted into the FG. As illustrated in Figure 1, frame insertion algorithm contains three parts: computing intersection frames, choosing the optimal alignment frames, and performing pair-wise alignment. As stated in Section V-A.6, our previous work has addressed the second problem but not the frame insertion problem as whole.

On the other hand, according to the $p-$ pixel limit imposed by the MVM algorithm in Section V-A.6 and the complexity bound of the pair-wise algorithm in SectionV-A.5, the overall pair-wise alignment time is $O(p)$. The remaining part is to find the existing frames that intersect the new frame, which is to find M_j for new frame j.

Assume there are n nodes in the FG at the moment. If n is small, an $O(n)$ linear brute-force search can identify the set. However, n grows as the number of frame sequences accumulates. Computing M_j efficiently requires an indexing data structure. Since we want to find out all overlapping frames, each of which is represented by a rectangle, this formulates a range search problem with the query window defined by the new frame. However, a regular 2D range searching problem [1] only reports points that intersect a query rectangle whereas the queried objects are also rectangles in our problem. A simple solution is to store center points of all existing frames and enlarge the query rectangle, which is similar to compute Minkowski Sums [4] for each queried rectangle. Therefore, we can identify set M_j in $O(\log^2 n + k)$ for $k = |M_j|$. With M_j, we can establish the edges between the new node and the existing nodes. The complete frame insertion algorithm is described as follows,

Frame Insertion Algorithm

```
Compute $M_j$ using range search,                         $O(\log^2 n+k)$
Add edges to FG,                                          $O(k)$
Run MVM Algorithm to get $\hat{M}_j$                      $O(k \log k)$
Run pair-wise alignment algorithm for each edge in $\hat{M}_j$,
                                                          $O(p)$
Update alignment edges,                                   $O(k)$
Insert the center point of the new frame to the range tree.
                                                          $O(\log^2 n)$
```

Theorem 1: If a range tree is used as indexing data structure, it takes $O(\log^2 n+k \log k+p)$ time to insert a new frame to a Frame Graph.

D. Frame Archiving Algorithm

A new frame may cover an old frame. If a frame has been mostly covered by its later neighboring frames, we should archive the frame to hard disk to reduce the number of nodes in the FG. Define $p_t \geq 1$ be the minimum number of pixels a frame has to contribute to the panorama, frame archiving algorithm is performed right after new frame j has been inserted,

Frame Archiving Algorithm

```
For each node $l \in M_j$                                              $O(k)$
    Compute region $\bar{R}_l = \{\cup_i R_{li}, i \in M_l, t_i > t_l\}$,     $O(k)$
    If $pixel\_number(R_l - \bar{R}_l) < p_t$,
        archive node $l$ and its edges                                 $O(k)$
        delete $l$ from the range tree                           $O(\log^2 n)$
    End If
End For
```

Theorem 2: It takes $O(k^2 + k \log^2 n)$ time to find and archive the old frames that are covered by a new frame.

E. Frame Adjustment Algorithm

On the other hand, a new frame may provide better alignment choice to its overlapping frames which leads to frame adjustment algorithm. After frame j enters the system, there is a subset of overlapping images $M_j - \hat{M}_j$ that are not used to align frame j. Based on [15], we know that the frames with big alignment errors are located in the subset. The frame adjustment algorithm is targeted at the worst aligned frame l in set $M_j - \hat{M}_j$. Define M_l and $\hat{M}_l$ be the set of overlapping frames and the set of alignment frames for frame l respectively. Let m_{jl} be the number of overlapping pixels between frame l and frame j.

Frame Adjustment Algorithm

```
Find the node $l \in M_j - \hat{M}_j$,                                   $O(k)$
Update $\hat{M}_l$ using the MVM Algorithm in Section V-A.6,
                                                                       $O(1)$
If $j \in \hat{M}_l$,
    Run pair-wise alignment algorithm between frame $l$ and
        frame $j$,                                                  $O(m_{jl})$
    Update alignment edges for frame $l$,                             $O(k)$
    Recursively adjust frames that aligned to frame $l$               $O(n)$
End If
```

As illustrated in the algorithm, for the adjusted frame l, we only need to perform one pair-wise alignment between frame l and frame j, which yields X_{jl} according to Section V-A.5. Equation 2 tells us that X_l can be refined incrementally because of the weighted sum format. Changing of X_l leads to the adjustment of all other frames that either directly or indirectly aligned to frame l. Since $n > k$, the total complexity of the frame adjustment algorithm is,

Theorem 3: It takes $O(n+m_{jl})$ time to adjust the alignment parameters of frame l and other effected frames after frame j enters the system.

VI. Experiments and Results

Our system is incrementally built on the lessons learned from experiments. The initial developments started in April of 2002. After one year's development and test, we deployed our first construction camera system in June 2003 to monitor Stanley Hall construction in UC Berkeley. At 285,000 square feet and 11 floors, the new Stanley building is the largest campus construction project in 20 years. The $162 million project is a large-scale research and teaching building scheduled to open in 2006. The camera used is a Canon VCC3 model. Figure 4(a) describes the site. At that time, we focused on building the camera control part and over 93060 frames have been recorded in the subsequent 2 years. Our data shown that the most frequent users were construction project managers. From their feedback, we noticed that there is a great interest for high-resolution panoramic video inspection and documentation system.

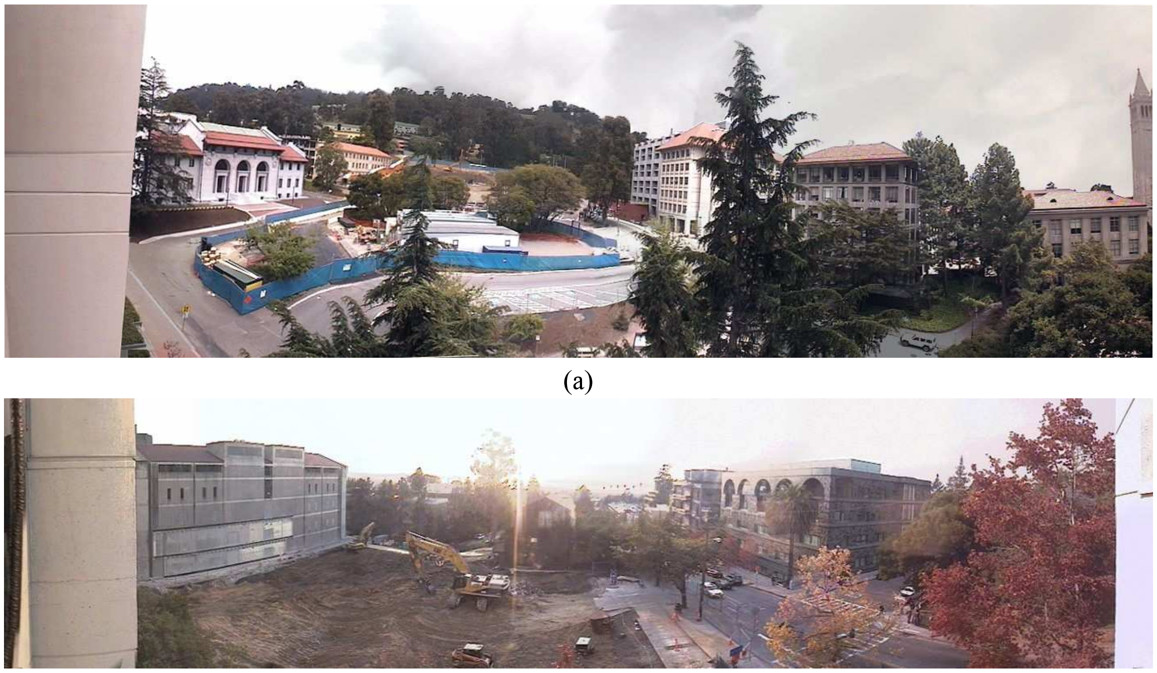
Our development of evolving panorama has started since the summer of 2003. We have deployed a Canon VCC4 camera in UC Berkeley Alpha Lab. As a test of concept, we built small panorama that were consisted of 8 frames. We soon have identified the problem that camera pan-tilt-zoom values can not provide adequate accuracy for frame registration and traditional static panorama generation method is either too slow to fit speed requirement or limited to simple small scale cylindrical panoramas. At the same time, we superimpose a live video on top of the panorama to provide a "context + focus" type of interface.

In September of 2004, we installed a Panasonic HCM 280 camera on top of MLK Student Union Building to view Berkeley Sproul Plaza, which is a very populated location in UC Berkeley Campus. This installation provides us the opportunity to test the combination of live video window with a panorama background in public. We received positive feedbacks from the one month and half test.

Shortly after the successful test in Sproul Plaza, we have moved our Canon VCC3 camera from the Stanley Hall construction site to a new CITRIS II building construction site in UC Berkeley. This $120 million project will add additional 145580 square feet research and teaching space to Berkeley campus at the of year 2007. The new installation is requested by construction contractors. Figure 4(b) illustrates the camera view at a resolution of $2600 * 900$ pixels. In the meaning time, our development in algorithms significantly reduces the panorama construction and update time. It took 3 minutes to construct the 8-frame panorama in October of 2003 but only 9.7 seconds to construct the 21-frame panorama in Figure 4(b) based on our latest results. After the panorama is constructed, it only takes 331 milliseconds to update it, which is as fast as the camera can be tele-operated. We are currently upgrading the system using a Canon VCC4 camera to achieve higher resolution. More experiments and results will be reported in the future revisions of this paper.

VII. Conclusions and Future Work

We present an automated video inspection and documentation system for construction sites using a networked robotic camera. Controlled by preset features, inputs from motion detectors, and occasional inspected commands, our system automatically scans and records regions of interests in a construction site. We construct an evolving panoramic display from the captured frame sequences. Whenever the camera changes its pan-tilt-zoom settings, we update the panorama by inserting a new frame sequence to our Frame Graph, which consists of a data structure and a set of algorithms to manage frame sequences. We report system details and results from our 3-year experiments in 2 testing sites.



(a)

(b)

Fig. 4. *Panoramas from two testing sites.*

ACKNOWLEDGMENTS

Thanks S. Frankel. R. Bajcsy, C. Séquin, D. Garcia, G. Baldwin, and S. McNally for help on camera installations and experiments. Thanks are given to X. Ling, and V. Jan for implementing part of the project. We thank B. Lin, A. Dahl, J. Schiff, I. Chen, K. Paulsen, J. Young, M. Gosalia, T. Shlain, G. Gershoni, J. Lecavalier for their contributions in Demonstrate system development and camera installation. Our thanks to A. Raima, L. Wilbur R. Volz, R. Gutierrez-Osuna, T. Ioerger, R. J. Luntz, P. Wright, D. Plautz, C. Cox, D. Kimber, Q. Liu, J. Foote, L. Wilcox, Y. Rui, K. "Gopal" Gopalakrishnan, R. Alterovitz, and I. Y. Song for insightful discussions and feedback.

REFERENCES

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

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

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

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. Springer, 1991.

[5] D. J. Fleet, M. J. Black, Y. Yacoob, and A. D. Jepson. Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3):171–193, Feb.-Mar. 2000.

[6] J. Foote and D. Kimber. Enhancing distance learning with panoramic video. In *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.

[7] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu. Mosaic representations of video sequences and their applications. *Signal Processing: Image Communication*, 8(4):327–351, May 1996.

[8] B. Y. Kim, K. H. Jang, and S. K. Jung. Adaptive strip compression for panorama video streaming. In *Computer Graphics International (CGI'04), Crete, Greece*, June 2004.

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

[10] S. K. Nayar. Catadioptric omnidirectional camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 482–488, June 1997.

[11] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool. Metric 3D surface reconstruction from uncalibrated image sequences. In *Proc. SMILE Workshop (post-ECCV'98)*, pages 138–153. Springer-Verlag, June 1998.

[12] Y. Y. Schechner and S. K. Nayar. Generalized mosaicing. In *Proceedings of the 8th IEEE International Conference on Computer Vision, Vancouver, British Columbia, Canada*, volume 1, pages 17–24, July 2001.

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

[14] D. Song, A. Pashkevich, and K. Goldberg. Sharecam part II: Approximate and distributed algorithms for a collaboratively controlled robotic webcam. In *IEEE/RSJ International Conference on Intelligent Robots (IROS)*, Nov. 2003.

[15] D. Song, N. Qin, and K. Goldberg. Algorithms for maintaining a high-resolution panoramic display with a tele-operated robotic camera. In *(Submitted to) 1st International Conference on Robotics: Science and Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts*, June 2005.

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

[17] K.-H. Tan, H. Hua, and Ahuja N. Multiview panoramic cameras using mirror pyramids. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):1941– 946, July 2004.

[18] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137 54, 1992/11/. Copyright 2005, IEE.

[19] E. Trucco, A. Doull, F. Odone, A. Fusiello, and D. M. Lane. Dynamic video mosaics and augmented reality for subsea inspection and monitoring. In *Oceanology International, United Kingdom*, March 2000.

[20] Y. Xiong and K. Turkowski. Creating image-based VR using a self-calibrating fisheye lens. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 237–243, June 1997.

[21] 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 *IEEE International Conference on Multimedia Computing and Systems, Florence, Italy*, volume 1, pages 9400–9406, June 1999.