A Real-Time Interactive Multi-View Video System

Jian-Guang Lou, Hua Cai, and Jiang Li
Media Communication Group, Microsoft Research Asia,
5F, Sigma Building, 49 Zhichun Road, Beijing 100080, China
{jlou, huacai, jiangli}@microsoft.com

ABSTRACT

With the rapid development of electronic and computing technology, multi-view video is attracting extensive interest recently due to its greatly enhanced viewing experience. In this paper, we present the system architecture for real-time capturing, processing, and interactive delivery of multi-view video. Unlike previous systems that mainly focus on multi-view video capturing, our system is designed to provide multi-view video service with high degree of interactivity in real time, which is still challenging in the current state of the technology. The proposed architecture tackles many practical problems in system calibration, object tracking, video compression, interactive delivery, etc. With the proposed system, users can interactively select their desired viewing directions and enjoy many exciting visual experiences, such as view switching, frozen moment and view sweeping, in real-time and with great freedom.

Categories and Subject Descriptors: I.4.9 [Computing Methodologies] Image Processing and Computer Vision – Applications

General Terms: Performance, Design, Standardization

Keywords: multi-view video, video streaming system, video coding, source coding, channel coding, calibration, object tracking

1. INTRODUCTION

Video service now has largely changed our daily life. With the technical advances in various areas, more and more flexibility has been provided by various video services. For example, upgrading from pre-captured movie to broadcast television (TV) is the significant progress which allows consumers to watch video at home. Live TV broadcast is another big improvement that delivers real-time video to consumers through the TV channel. The analog video signal was then digitized and digital TV appeared, which gives better quality, more programs, and more flexibility. Also, many other applications, such as digital video disc (DVD), video-on-demand (VoD), video streaming over the Internet, etc., were invented to allow consumers to watch their favorite video programs more easily. After the above significant and exciting progress, the conventional video form (called single-view video hereinafter) is good enough in many cases. However, for interactive or entertainment-oriented applications, it still has several limitations. First, it only provides one view direction for an event at any time instance while users may want to watch the event from different directions. Second, users are in a passive position. Even in a live video service, users can only watch the pre-selected video contents whereas there is no or little interactivity between the users and the content capturing. Third, recording an event from one fixed/dynamic view direction is not always the best way, from both visual experience and event representation criteria. For example, in a high action sports game or in an exercise diagnosis, audience or instructors often want to watch the video from comprehensive views, which gives them better experience or helps them to make a correct judgment.

Thanks to the rapid development of electronic and computing technology, multi-view video service is becoming more and more feasible in practice. Different from the conventional single-view video, multi-view video consists of a set of video clips that are captured simultaneously from a group of video cameras with different view directions. In consequence, users can not only view a program from their desired view direction, but also enjoy some special visual effects provided by the multi-view video. As a natural extension to the single-view video, multi-view video can be widely used in many applications, including advertisement, educational program (such as surgical instructions), sports games, and some important events.

Due to its promising features and a large number of potential applications, multi-view video has been attracting more and more attention in recent years. For example, an ad hoc committee on 3D audio and video (3DAV) was founded by the MPEG community in 2001. Since then, some investigations and primitive activities have been made [1, 2]. Omni-directional video, free view point video, stereoscopic video and depth video have been primitively discussed in their investigations. Meanwhile, some research efforts were also made on the design of the capturing system. For instance, a 33-camcorder system called EyeVision (www.ri.cmu.edu) was employed to shoot Super-bowl 2001. The videos captured from the 33 camcorders are all input to a video routing switcher and an edited video was broadcast to TV viewers. In the EyeVision system, audience are still in a passive position, and no interaction of end users is involved. Later

MM'05, November 6–11, 2005, Singapore.

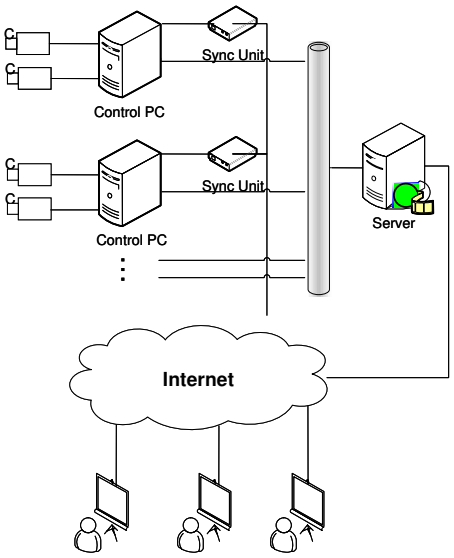


Figure 1: System architecture.

systems such as Digital Air's $Movia^{TM}$ (www.movia.com) and Wurmlin's 3D Video Recorder [12] were proposed to capture, store, and playback multi-view video. In the system of $Movia^{TM}$, the designers use miniature high speed, high definition, uncompressed digital cinema cameras that can produce full frame synchronous images at variable frame rates and variable shutter speeds. The uncompressed digital images are directly saved to some specially designed digital video recorders based on Intel and SGI powerful computers in real time. The recorded raw images will be further processed using post-production tools. On the other hand, based on the pre-captured multi-view video, a lot of post-processing techniques such as view interpolation [16], depth-based methods [9], and model-based methods [6] were widely studied to enhance the viewing experience.

Despite the above work on multi-view video systems and technology, the unique features of multi-view video have not been exploited fully, especially in a real-time and interactive scenario. In this paper, we propose a new system architecture that is designed to provide real-time, interactive, and reliable multi-view video service. This clearly differentiates our system from other existing ones that only put efforts on how to record multi-view video signals. The proposed system has many advantages benefiting from its real-time and interactive properties. It allows the audience to change his/her view direction and enjoy some special visual effects freely while watching the live video. Such a system could largely enhance the user experience for interactive and/or entertainment-orientated applications. Of course, there are lots of practical challenges to build such a real-time interactive system which will be addressed in this paper.

The rest of the paper is organized as follows. Section 2 outlines the hardware and software architecture of our system. From Section 3.1 to 3.5, we describe system calibration, object tracking, video capturing, multi-view video delivery, and the video viewer respectively. Some experimental results are presented and discussed in Section 4. In Section 5, more discussions are included. Finally, we conclude our work in Section 6.

2. SYSTEM ARCHITECTURE

To fully exploit the unique features of multi-view video, we propose a system architecture in this paper. In order to support users' interactivities, our system is built upon a computer network. The system can be used for both on-demand delivery and live broadcast. As shown in Fig. 1, the system mainly consists of N video cameras, N pan-tilt units, a set of control PCs and synchronization units, a server, a network backbone, and many receivers (clients). These components can be classified into three parts: capturing part, server part and client part.

The capturing part is composed of cameras, lenses, synchronization units, pan-tilt units, and control PCs. Every camera is put on a pan-tilt unit, which can be controlled by a control PC. There are several types of spatial camera configuration for multi-view video, e.g. parallel view, convergent view and divergent view. Each of them has its own appropriate applications. In order to capture a single dynamic event from different view directions, we use the convergent configuration by placing these cameras on an arc with equal distance between them. Moreover, to make the cameras operate simultaneously, we connect their synchronization units to the control PCs. The generated synchronization signals can make all the cameras trigger and shoot at the same time instant. As a result, a dynamic event can be captured simultaneously by multiple cameras from different view directions. After that, the captured video signals are compressed in the control PCs and then sent to the server through a network backbone, e.g. a gigabit Ethernet.

The server part collects the N compressed video streams from the control PCs and provides multi-view video service to end users. In addition, the server also acts as a camera manager. It generates control signals for cameras, changes the cameras' view directions, and sets the properties of the cameras through sending commands to the control PCs.

The client part receives multi-view video bit streams from the server. It then decodes bit stream and displays the video for end users. By controlling the client part, users can enjoy live interactive multi-view video. Besides the features provided by traditional video services, with the proposed system, users can enjoy many exciting visual experiences in real-time and with great freedom as shown in Fig. 2:

> ***View Switching***: Users are able to switch flexibly from one camera view direction to another as the video continues along time.
>
> ***Frozen Moment***: In the frozen moment, time is frozen and the camera view direction rotates about a given point. One example is that, as shown in Fig. 2, a user can view frames $f_1(i)$, $f_2(i)$, ..., $f_N(i)$ back and forth at the i^{th} frame of time instant.
>
> ***View Sweeping***: It involves sweeping through adjacent view directions while the time is still moving. It allows the user to view the event from different view directions. One example is that, as shown in Fig. 2, a user can view frames $f_1(i)$, $f_2(i+1)$, ..., $f_N(i+N-1)$ starting at the i^{th} frame of time instant.

3. SYSTEM MODULES

In this section, we focus on the major modules including system calibration, object tracking, video capturing, video delivery, and video viewer.

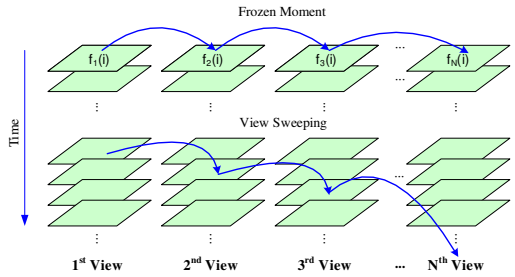


Figure 2: Examples of visual effects provided by multi-view video ($f_n(i)$ denotes the i^{th} frame of the n^{th} view).

The system calibration consists of geometrical calibration and color normalization. In a multi-view video system, multiple cameras are deployed to capture videos simultaneously at different viewing angles. Geometrical calibration tries to estimate the geometrical relationships among multiple cameras and their hand units. Based on this, the cameras can be driven to shot at the same event. Meanwhile, color normalization aims to make the color experience smoother during view transition.

In a dynamic event space, the object that we are interested in often moves around. In order to capture videos of the moving object, we should rotate the cameras to track it. This is implemented in the object tracking module.

Video capturing plays an important role in the whole system. It collects, compresses, and protects the video signals obtained from cameras. The main challenge for video capturing is its heave computational cost. We proposed a complexity scalable coding framework in our video capturing module.

After video capturing, the captured video will be collected by the server, and finally delivered to end users. According to the unique using style of multi-view video observed from the user study, we find a proper tradeoff among flexibility, latency and bandwidth for the multi-view video delivery.

In addition, a client multi-view video viewer is designed to support real-time viewing and off-line playback.

3.1 System Calibration

3.1.1 Geometrical Calibration

All video cameras have to look at the same central event point in the event space very precisely. The difference between the positions of the central event point in two neighboring images should be less than 2 pixels. The distance between each pair of successive cameras should be constant, and the poses of the cameras from the first to the last should vary smoothly. Otherwise, users will observe an awkward jittering during view transition. In our system, the view direction of each camera can be changed by its pan-tilt unit. After precisely adjusting the cameras' heights, we can use our object tracking module to drive all cameras to point to the same central event point. Each camera's intrinsic parameters, extrinsic parameters, and hand-eye relationship are employed during the object tracking procedure. Before the object tracking, the cameras should be carefully calibrated.

Camera calibration is a classical problem in computer vision. Many approaches have been proposed in the past decades. In our system, we use Zhang's plane-based calibration [14] method to estimate the intrinsic parameters. The major advantage of the plane-based method is that it is much easier to make a planar pattern than a 3D reference object.

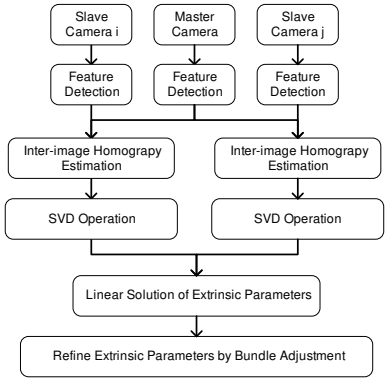


Figure 3: The pattern free calibration.

After the calibration of intrinsic parameters, we then focus on the calibration of extrinsic parameters. In general, multi-camera calibration can be realized by a two-step scheme: First, we calibrate each camera with a single camera calibration process. Then, we study the relationship between these cameras if we use the same reference object in the calibration of all these cameras. However, this method is only suitable for close range applications. As the dimensions of the view volume increase, setting up a precisely planar calibration pattern also becomes difficult. Fortunately, in most environments, e.g., a gymnasium, there is always a dominating plane, e.g., the ground plane. We propose a calibration method which can be used to calibrate the cameras in a large scale scenario based on detected features on the ground plane without a calibration pattern, thus, we call the algorithm as pattern free calibration.

3.1.1.1 Pattern Free Calibration

We set one camera as the reference camera, for example, the master camera that shares a large field of view with the slave cameras. Fig. 3 provides a general flow diagram of the pattern free calibration procedure of the interactive multi-view video system. First, we extract the feature points in each image of the master and slave cameras. Using these feature points, we estimate a set of inter-image homographies that map the features in each image of the slave cameras to the image of the master camera. By exploiting the properties of these inter-image homographies, we obtain a linear solution of the extrinsic parameters. Supposing the matrix $\mathbf{H}$ be a homography from the reference image to a slave image, we define $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{A}^T\mathbf{H}^T\mathbf{A}_1^{-T}\mathbf{A}_1^{-1}\mathbf{H}\mathbf{A} \quad (1)$$

where $\mathbf{A}$ and $\mathbf{A}_1$ are the intrinsic parameters of the master and the slave camera respectively. According to the definition of $\mathbf{H}$ [7], we find the matrix $\mathbf{M}$ can be decomposed to

$$\mathbf{M} = \lambda(\mathbf{I} + \mathbf{n}\mathbf{t}^T + \mathbf{t}\mathbf{n}^T + k^2\mathbf{n}\mathbf{n}^T) \quad (2)$$

where $\mathbf{n}$ is the unit normal vector of the ground plane, $\mathbf{t}$ is the normalized translation from the master camera to the

slave camera, λ is a non-zero scale factor, and k is defined by $k^2 = t^T t$.

On the other hand, we find that the matrix $\mathbf{M}$ is a 3×3 symmetric and nonnegative definite matrix which contains three positive eigenvalues and eigenvectors. Supposing λ_1, λ_2 and λ_3 ($\lambda_1 < \lambda_2 < \lambda_3$) to be three eigenvalues of the matrix $\mathbf{M}$, and their corresponding eigenvectors are $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$, we can find that the cameras' extrinsic parameters satisfy the following linear equations based on these eigen components

$$\begin{cases} \mathbf{v}_1^T \mathbf{n} &= b_1(a_1 p + 1) \\ \mathbf{v}_2^T \mathbf{n} &= 0 \\ \mathbf{v}_3^T \mathbf{n} &= b_3(a_3 p + 1) \end{cases} \quad (3)$$

where b_1, b_3, a_1, a_3 and p can be calculated from the eigen components

$$p = \sqrt{\lambda_1 \lambda_3} - 1 \quad (4)$$

$$k = \sqrt{\lambda_3} - \sqrt{\lambda_1} \quad (5)$$

$$a_{1,3} = \frac{\pm\sqrt{k^2 + 4p + 4} - k}{2k(p+1)} \quad (6)$$

$$b_{1,3} = \frac{1}{\sqrt{a_{1,3}^2 k^2 + 2a_{1,3} p + 1}} \quad (7)$$

Given more than 2 homographies, we can obtain a set of linear equations, and the camera parameters $\mathbf{n}$ can be estimated by a least squares minimization.

Then, we refine the extrinsic camera parameters by bundle adjustment procedure based on a Levenberg-Marquardt (LM) method.

Once the intrinsic and extrinsic parameters are calibrated, the hand-eye parameters can be further calibrated based on the extrinsic parameters with more than 2 pan-tilt position changes. The hand-eye calibration method described in [11] is adopted in our system.

3.1.2 Color Calibration

Color calibration is also a very important step in the preparation of the system. The goal of color normalization is to make the illumination change smoothly while users watch the views from the first camera all the way to the last camera. Without careful color calibration, users will notice an annoying flicker in the frozen moment effect. Thus, the color calibration largely influences the overall performance of the system. Color calibration consists of white balancing and intensity normalization. The objective of intensity normalization is to make the captured images of different cameras possess a similar intensity distribution.

By studying the captured images from the neighboring cameras, we found that neighboring views often shared some overlapped image regions. If we can find all overlapped regions from all neighboring views (e.g., every 3 views) and the correspondence among them, we can then know the relationship of the color responses (BTF, brightness transform function) in these cameras. In order to find the overlapped regions from the neighboring views, we adopt a matching process based on color segmentation. The robustness of the color segmentation based matching algorithms has been proven in the area of stereo matching [10]. After the matching process, we can smooth the variation of color and brightness from view to view. The approach is to minimize the difference between the histograms of neighboring views. Instead of taking only one neighboring view as a reference image [5], we use both the left and right neighbor views of the current view as reference images.

From experiments, we find that there are some artifacts if we directly apply the color histogram specification on the images. This is because spatial continuity is not taken into consideration in histogram specification. Artifacts usually happen on the edges or in regions with rich texture. In order to preserve the local detail and remove the artifacts, we first apply the normalization to the original images, and only record the color modification values as a 2D difference fields. The value on each pixel position of the 2D difference field is the changed value of the pixel under the normalization. Because of the global effect of the histogram specification, some isolated pixels or edges are mapped to wrong intensities, while the neighboring pixels of these inaccurately mapped pixels often remain correct. We can recover those inaccurately mapped pixels by smoothing the difference field with a median filter. In our system, a 3×3 median filter is utilized.

The overall algorithm is described as:

Step 1: Do meanshift segmentation for all view images;
Step 2: Find the match between every two sequential images I_i and I_{i+1};
Step 3: Find the BTF_i using histograms;
Step 4: Iteratively refine BTFs by smoothing the difference of sequential histograms;
Step 5: Obtain the change fields by appling the BTFs onto images;
Step 6: Filter the change fields, and apply them onto the images.

In the real world, lighting condition often varies with time. Furthermore, the characteristics of CCD chips in cameras can be influenced by temperature. We have to redo the color calibration at certain time intervals.

3.2 Object Tracking

In our system, cameras are configured in a master-slave way. That is, one of the cameras is selected as a master camera, and all the others work as slave cameras. The master camera is often controlled by a camera man. The slave cameras can be driven to point to the same interest point of the master camera. This is realized by a so-called master-slave tracking process. Fig. 4 shows the idea of master-slave tracking. In the system calibration step, we have obtained the geometrical relationships between the world coordinate w, the cameras C_0, C_1 and the pan-tilt units h_0, h_1. Given these parameters and the position of the master pan-tilt unit, we should determine the position of each slave pan-tilt unit. In fact, only given the rotation parameters of the master camera, we cannot determine the target event point. In our system, we suppose that the position of the target point is located at an event plane (the relationship is represented by $n^T X = d$) where most of interesting events happen. For example, in a soccer game, the event plane can be approximated by half of the average heights of all players. Experimental results show that a slight bias on the event plane does not largely influence the performance.

To obtain the pan/tilt parameters of the slave cameras, we firstly determine the target point in the 3D event space. Given the rotation R of the master camera, we can determine the resulting projection matrix $\mathbf{M}$ by

$$\mathbf{M} = \mathbf{M_0 Y R Y^{-1}} \quad (8)$$

where $\mathbf{M_0}$ and $\mathbf{Y}$ are the original projection matrix and the hand-eye relationship respectively. If we rewrite the matrix as

$$\mathbf{M} = [\mathbf{N}\ \mathbf{v}] \tag{9}$$

where $\mathbf{N}$ is a 3×3 matrix and $\mathbf{v}$ is a 3-vector, the target point $\mathbf{X} = [x, y, z]^T$ can be determined by the following equations:

$$\begin{cases} -\mathbf{N^{-1}v} + t_1 \det(\mathbf{N}) N_3^T &= \mathbf{X} \\ n^T \mathbf{X} &= d \end{cases} \tag{10}$$

where N_3 is the third row of the matrix $\mathbf{N}$, and t_1 and $\mathbf{X}$ are the unknown variables.

The next task is to determine the rotation parameters of every slave camera based on the obtained target point. For each slave camera, if we directly establish an equation under the constraint that the optical axis of the camera must go through the target point, we will obtain a trigonometric equation of 8^{th} order, which is quite difficult to be solved. Our solution is to decouple the procedure into two steps. In the first step, we calculate the intersection point p of the optical axis and a sphere. The sphere is centered at the origin of the coordinate system of the slave's pan-tilt unit, and the target point is located on the sphere. This means both the point p and the target point are located on the surface of the sphere. Now, the problem is converted to finding rotating parameters that make the point p coincident with the target point on the sphere. In general, there are four pairs of solutions. We choose the one which requires less change in pan-tilt angles.

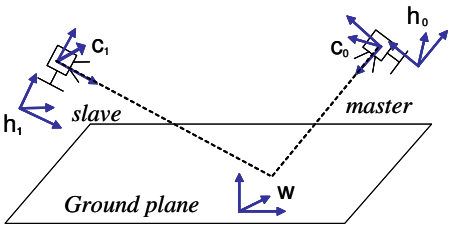


Figure 4: Master-Slave Tracking. w is the world coordinate system, h_0 and h_1 are the hand coordinate systems, and C_0 and C_1 are the camera coordinate systems.

3.3 Video Capturing

Video capturing module collects, compresses, and protects the video signals obtained from cameras. Although some special hardware like a video capturing card can compress video in real-time, they are often not flexible enough to meet different requirements (such as different video resolutions, different frame rates, and different quality levels, etc.). Thus, in our system, a pure software implementation is adopted.

When designing the software implementation, computational complexity is the most challenging factor. This is because video capturing, especially for the video compression module, consumes most of computing resources of a control PC. Also, system scalability should be taken into account. In the rest of this subsection, we will discuss how to design video compression and channel coding modules in order to well tackle the above challenges.

3.3.1 Complexity Scalable Video Coding

Real-time video coding faces a big challenge from computational complexity. It is not a cost-effective way to simplify the algorithms to meet a specific scenario (e.g., a given video resolution and bit rate for a certain device), since there are so many different scenarios. To achieve better complexity adaptation and system scalability, we propose a computational-complexity scalable video encoder that can offer a trade-off between the coding efficiency and the embedded available computational performance.

It is known that motion estimation (ME) consumes most of the computing time in real-time video coding systems [3]. Although there are significant advances in fast ME techniques [8, 15] in recent years for alleviating the heavy computation load, ME still consumes the largest amount of computational resources in real-time video encoding. Hence, we put our efforts on the design of a novel ME framework that offers fine-granular computational-complexity scalability [13] [1]. The key idea is to partition the ME process into multiple search passes. A priority function is then used to represent the distortion reduction efficiency of each pass. According to the predicted priority of each macro block (MB), computational resources are allocated effectively in a progressive way. As a result, the ME process can be stopped at any time with a progressively improved performance, and thus scalability is achievable. Furthermore, the proposed scheme can be easily integrated with many existing fast ME algorithms, such as NTSS [8], DS [15], and etc.

3.3.2 Channel Coding

Although the compressed video could be transmitted to the receivers' end directly, the quality will often be hurt seriously due to the error-prone feature of the communication channel. Automatic repeat request (ARQ) based methods are often adopted to rescue packet losses. However, they are usually not acceptable for real-time interactive applications because of the excessive end-to-end delay caused. On the other hand, forward error correction (FEC) techniques [4] can correct packet losses promptly without any further intervention from the sender: when K data packets are protected with T parity packets, packet losses can be recovered successfully if K out of K + T packets are received.

To improve system scalability, the channel coding process is separated into two passes: parity packets creation at the control PCs' side and parity packets selection at the server side. First, at the control PCs' side, the video bit stream is packetized into fixed length data packets. The Reed Solomon code [5] is then used to create T parity packets for every K data packets. After that, both of the data packets and parity packets are transmitted to the server side. Next, when the video bit stream of a certain view direction needs to be delivered from the server side, the server will select and transmit $\hat{T}$ ($\hat{T} \leq T$) parity packets to the end users according to the specific channel conditions. As a result, since most of computationally intensive calculation of the channel coding process is distributed to each control PC, good system scalability can be attained.

3.4 Multi-View Video Delivery

Through the high speed LAN, the captured N video streams

[1] In addition to this new ME framework, complexity scalability is also achieved through some other improvements such as coding mode decision, etc. Due to space limitations, we will only describe briefly the ME part in this paper.

are transmitted respectively from the control PCs to the server. The server will then process these N streams and provide real-time interactive multi-view video service to the end users. The service includes not only conventional live video, but the special visual effects mentioned in Section 1, that is, *view switching*, *frozen moment* and *view sweeping*.

Ideally, users should be able to enjoy the special visual effects at any time they want, and the effects should be rendered immediately after users' actions. Meanwhile, the service should be provided with minimum bandwidth consumption. However, these requirements are inconsistent. For example, lower latency and more flexible action often result in higher bandwidth cost. To find a proper tradeoff among flexibility, latency, and bandwidth cost, we carried out a user study at the beginning of our work. We found three key observations from the study:

1. Users often stay in their favorite view directions for a relatively long period, so they will not change their view directions frequently.

2. When an exciting event happens, users will be more interested in watching a *frozen moment* and *view sweeping*. Moreover, users can get a more exciting experience from the *frozen moment*, whereas experience from the view sweeping is not always attractive. This is because the experience of *view sweeping* is very similar to the experience from an ordinary moving camera.

3. Users can tolerate a relatively long, e.g., 1 second, view switching latency. This is similar to the consumers' attitude on the program switching latency in digital TV. On the other hand, lower latency should be provided for the *frozen moment* and *view sweeping*.

Based on the above observations, we prepared two kinds of video streams at the server side. The first one, denoted by $\mathbf{V_i}$ $(1 \leq i \leq N)$, is the conventional single-view video stream that is captured individually at the control PC:

$$\mathbf{V_i} = \{f_i(1), f_i(2), f_i(3), ...\} \tag{11}$$

where $f_i(n)$ stands for the n^{th} frame of the i^{th} view direction. As discussed in Section 3.3, each $\mathbf{V_i}$ is compressed independently by the MPEG-liked encoder with a τ_v-second key frame interval. This key frame interval determines the view switching latency. Hence, according to our third observation, τ_v is set to 1 second in our system.

The other kind of streams are the frozen moment stream $\mathbf{F}$ and the view-sweeping stream $\mathbf{S}$, which provide respectively the *frozen moment* effect and the *view sweeping* effect. Each stream consists of many snapshots:

$$\begin{aligned} \mathbf{F} &= \{F(1), F(2), F(3), ...\} \\ \mathbf{S} &= \{S(1), S(2), S(3), ...\} \end{aligned} \tag{12}$$

while each snapshot is composed of N frames from different view directions:

$$\begin{aligned} F(n) &= \{f_1(n), f_2(n), ..., f_N(n)\} \\ S(n) &= \{f_1(n), f_2(n+1), ..., f_N(n+N)\}. \end{aligned} \tag{13}$$

Although the corresponding frames of $F(n)$ and $S(n)$ have already been compressed in $\mathbf{V_i}$, they cannot be used directly to form $F(n)$ and $S(n)$. This is because, firstly, $\mathbf{V_i}$ is encoded in a temporally predictive manner, thus decoding a certain P frame requires all of its dependent frames up to the recent I frame. Secondly, even if all these frames are encoded as I frames that do not depend on other frames, the compression efficiency will be very low. Therefore, these frames need to be re-encoded.

Since frames of $F(n)$ or $S(n)$ are captured from the same event but with different view directions, they are highly correlated. To exploit the view correlation, we predictively encode frames of the same snapshot similar to the conventional motion-compensated video encoding. The first frame, $f_1(n)$, is encoded as an I frame, and the subsequent $N-1$ frames are encoded as P frames. Such a compression strategy has three advantages: (1) A higher coding efficiency can be achieved as the view correlation is utilized. (2) Each snapshot can be decoded independently without knowledge of other snapshots, since it is encoded separately without prediction from other frames of different snapshots. This not only simplifies the implementation, but also reduces the decoding latency. (3) The decoder can treat the bitstream as a single video stream of the same format, no matter what kind of effect it provides. This is very important for compatibility with decoders in many end devices, such as the set-top box.

As computation is required to re-encode snapshot $F(n)$ or $S(n)$, it is difficult for the server to process every snapshot due to its limited computing resources. On the other hand, it is unnecessary to include every snapshot into stream $\mathbf{F}$ or $\mathbf{S}$, especially for events with slow motion. Because of the above reasons, the snapshots should be sub-sampled first. In our system, a snapshot will be generated every 15 frames. That is, the practical sub-sampled $\mathbf{F}$ and $\mathbf{S}$ are:

$$\begin{aligned} \mathbf{F} &= \{..., F(n-15), F(n), F(n+15), ...\} \\ \mathbf{S} &= \{..., S(n-15), S(n), S(n+15), ...\}. \end{aligned} \tag{14}$$

On the other hand, before re-encoding frames in $F(n)$ and $S(n)$, the server must call a decoding process since these frames have already been encoded by the control PCs. To reduce decoding complexity, we should compress these frames at the control PCs as I frames so that each frame can be decoded independently. This can be done by performing I-frame encoding at control PCs for these snapshot frames. For example, in addition to generating $\mathbf{V_i}$ during the video capturing, we must create an I frame every 15 frames for each view direction in order to form the frozen moment stream $\mathbf{F}$.

The generated streams $\mathbf{F}$ and $\mathbf{S}$ are then channel coded at the server with the same method used at the control PCs. After that, multi-view video delivery will be performed either in a streaming mode or in a broadcast mode.

3.4.1 Streaming Mode

In the streaming mode, all of the streams $\mathbf{V_i}$, $\mathbf{F}$, and $\mathbf{S}$ are used for interactive delivery. In particular, the server will buffer $\mathbf{F}$ and $\mathbf{S}$ for τ_B seconds in order to combat the network latency. When a certain user subscribes to the server, the multi-view video service will be provided. Usually, the user will first see a default view direction, which might be the most attractive one among the N view directions. The user can then switch to other view directions, or enjoy the *frozen moment* or *view sweeping* effect by controlling the client player as shown in Fig. 5.

If a *view switching* command is received, the server will continue sending video stream of the current view direction until reaching the next I frame. After that, it will send

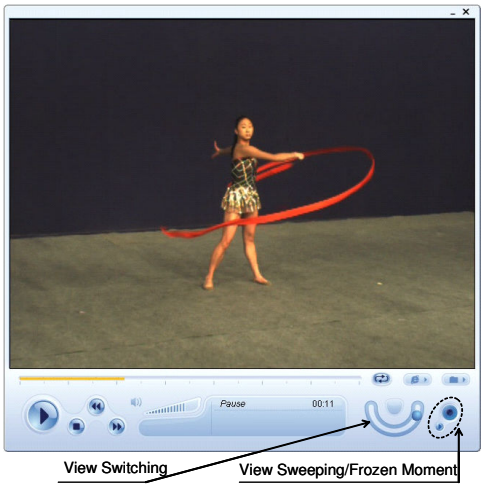


Figure 5: The user interface of a remote client.

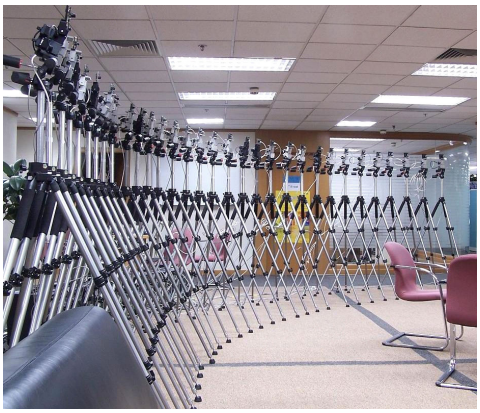


Figure 6: A deployment of our system.

video stream of the new view direction starting from that I frame. Thus, the maximum latency perceived by that user is $\tau_n + \tau_v$, where τ_n is the network latency.

If a *frozen moment* or *view sweeping* command is received, the server will look for the appropriate snapshot $F(n)$ or $S(n)$ from the buffered **F** or **S** stream. Here the appropriate snapshot means the one whose time stamp is close to the command's creation time. Then, the found snapshot will be sent immediately. After sending the snapshot, the server will send the video stream of the current view direction as usual. Obviously, the maximum latency perceived by that user is $\tau_n + \tau_f$ or $\tau_n + \tau_s$, where τ_f and τ_s are the snapshot interval for **F** and **S** respectively. In our system, both of them are equal to 0.5 seconds as described before.

3.4.2 Broadcast Mode

The broadcast mode is proposed for users with fixed downlink channel bandwidth, e.g., cable TV users. To reduce the bandwidth cost, we only select $\hat{N}$ view directions from the original N views. These $\hat{N}$ views will be delivered to end users together with one special stream, either **F** or **S**. Users' operations will be performed directly upon the received multiple streams, whereas no interactivity is required between the end users and the server. In this case, the maximum latency for *view switching*, *frozen moment*, and *view sweeping* is equal to τ_v, τ_f, and τ_s, respectively.

3.5 Multi-view Video Viewer

We developed a prototype of a multi-view video viewer on a PC as software. This viewer connects to the server, receives the video stream and renders it. In addition, it also supports playing back a stored multi-view video file. Fig. 5 shows our client player. Besides the traditional control commands of a media player, our viewer also supports control interfaces for view selection and special effects. The user can interact with the system, changes his/her view direction, and enjoys the visual effects of frozen moment and view sweeping with one click of the mouse on the control buttons.

4. EXPERIMENTS AND DISCUSSIONS

The proposed system architecture is deployed for various tests. 32 video cameras are used to capture the dynamic event from 32 view directions. A lens with a fixed 8mm focal length is used for each camera. These 32 cameras are located at equal interval along an arc. The radius of the arc is about 6.5 meters. As our experiments indicate, a change of view angle below 5 degrees can basically provide a smooth viewing experience in the *frozen moment*, so the view angle difference between two adjacent cameras is set to 3 degrees in this implementation. As a result, an over 90-degree view angle can be provided. Fig. 6 shows a typical deployment of our system.

The position of each camera is controlled by the pan-tilt unit via the RS232 interface. To guarantee control accuracy, we specially designed the pan-tilt units which can be driven by step motors precisely, i.e., 0.0127 degree per step.

Each camera is then connected to a control PC through its 1394 interface. Due to the bandwidth limit of the 1394 bus and the processing capability of the control PC, every two cameras are connected respectively to one control PC. As a result, 16 control PCs are required. These control PCs are synchronized by 16 synchronization units. After that, each camera captures video signals simultaneously at a 640×480 resolution with 30 fps. Next, the control PCs are connected to a server through a gigabit Ethernet. In this implementation, both the control PCs and the server have the same hardware configuration: a Pentium-IV 2.8G Hz CPU and a 512M RAM.

Fig. 7 shows some sample images of a frozen moment and sweeping effect video snapshots that are captured by this system. As can be seen from these images, the frozen-moment and view sweeping effects really provide users a brand new visual experience.

During the construction of the system, various tests have been carried out. Before the overall test of the whole system, we first examine the main system components, including camera calibration, color normalization, video capturing, multi-view video delivery, and the server's load. Then, some user studies are conducted.

4.1 System Performance Evaluation

4.1.1 Geometrical calibration

Our proposed pattern free calibration algorithm is used for geometrical calibration. Firstly, as shown in Fig. 8(a), we randomly place dozens of black patches on the ground

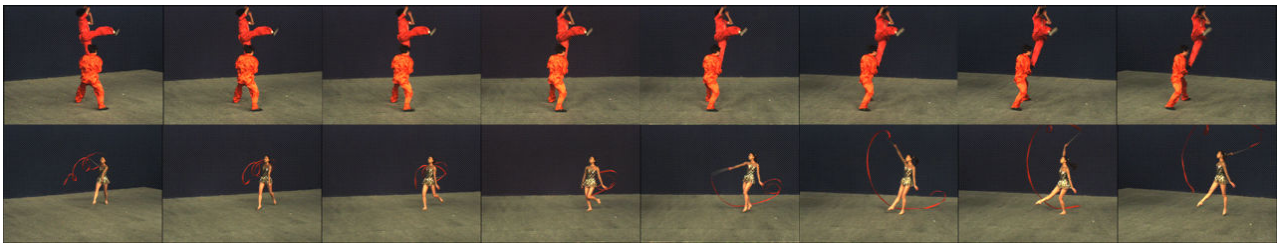
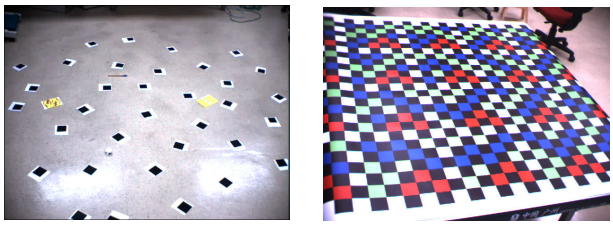


Figure 7: Sample image sequence. The upper row contains images from a frozen moment effect, while the lower row contains images from a sweeping clip. From the left to the right one, the images are sequentially captured from view 0, 4, 8, 12, 16, 20, 24 and 28.

plane as seeds of feature points. After that, we set the image from the master camera as the reference image. Geometrical calibration is then carried out automatically. Re-project error [7] is used for evaluating the calibration accuracy. The average re-projection error of our method is 0.7 pixels, with a standard variance of 0.34 pixels.

Moreover, the proposed method is also compared with Zhang's planar-based method [14], which can be used for calibrating the extrinsic parameters in a small volume scene. As shown in Fig. 8(b), a printed pattern plane with $2m \times 3m$ is placed in the views of these cameras. Both our proposed method and Zhang's method are used respectively for extrinsic parameters calibration. To compare the performance of different methods, we project the feature points from the slave images onto the reference image using the calibrated results, and then measure the re-project errors. The average re-projection errors of both methods are quite similar, i.e. 1.66 pixels for Zhang's method and 1.68 pixels for our method. It indicates that the accuracy of our method is comparable to Zhang's algorithm.



(a) (b)

Figure 8: (a) Ground plane with random distributed black patches; (b) A planar pattern.

In our system, the master-slave tracking only depends on the calibration results, and its accuracy is very similar to that of the calibration algorithm.

4.1.2 Color Normalization

Color normalization largely smoothen the visual experience during view transition. In Fig. 4.1.2, the left two are the images before color normalization, and the right two are the corresponding images after normalization. Experimental results show that our color normalization module meets the system requirement.

4.1.3 Video Capturing

With the proposed complexity scalable video encoder, a

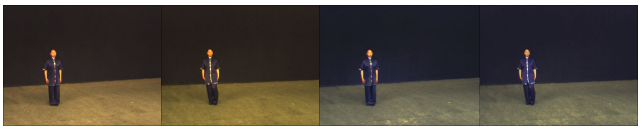


Figure 9: Color normalization results.

P-frame encoding speed within a range of 200 fps to over 400 fps for the standard CIF (352×288) test sequences can be achieved on the control PC. This indicates that the new encoder greatly improves the complexity adaptation and system scalability.

When two video streams (640×480 and 30 fps) are compressed simultaneously on the control PC together with the proposed channel coding strategy, the CPU usage keeps below 70%.

4.1.4 Multi-View Video Delivery

The multi-view video delivery, the network overhead is an important aspect for the end-to-end performance. In our system, the average bit rate of an individual view is about 1.5M bps. The bit rate for the frozen-moment stream and the view sweeping steam is a bit higher, as they are usually more complex than the individual view. The average peak-signal-to-noise-ratio (PSNR) value is around 35 dB. Of course, the bit rate depends on a lot of practical factors including the complexity of the background, the motion in the video, the lighting condition in the event space and the video quality we try to present. By using more efficient video compression algorithms for both the individual view and the special effect streams, the network overhead can be further reduced.

The bandwidth cost during the interactive delivery is also evaluated. As shown in Fig. 10, two users A (red curve) and B (blue curve) connect to the server. At the beginning, both of them watch the same view, which costs them an average bit rate of 1.5M bps. Next, at time stamp 0.66s, user B subscribes for a frozen-moment effect, then the bandwidth cost increases to a peak value of 2.8M bps. After that, at timestamp 1.66s, user B keeps watching the original view as the frozen-moment effect has been finished. Then both of them have the same bandwidth cost.

Finally, a real-time test was performed through a broadband network. The system was set up in Beijing, China, while the interactive multi-view video service was provided in the streaming mode to multiple users in Seattle, USA. Users can enjoy view switching, frozen moment, and view

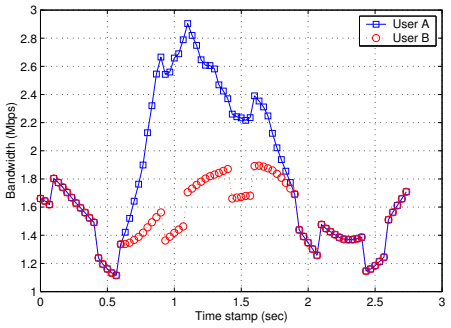


Figure 10: The network throughput of interactive delivery. Red curve is the network throughput of a single view stream, and the blue curve is that of a clip containing a frozen moment effect.

sweeping effects with an about 2M bps bandwidth cost and a latency of $0.2 \sim 0.8$ seconds.

4.1.5 Server Load

Two tests regarding the system performance were performed. The first one studies the relationship between the computational cost on the server side and the number of cameras adopted. During the test, no users connected to the server, thus most of the computing resource is consumed by re-encoding the snapshots. As can be seen in Fig. 11(a), the server's CPU load increases approximately linearly when the number of cameras increases. The CPU load increases about 2% when a camera is added to the system.

The second test studies the relationship between the computational cost on the server side and the number of users served in the streaming mode. In the test, all of the 32 cameras are adopted and all clients are located in a local area network. To simulate the interactivity between the server and the users, we added a random command generator upon each client. As a result, a random view switching command is sent by each client with a mean time interval of 30 seconds. Meanwhile, a random frozen moment command is sent with a mean time interval of 10 seconds. In other words, we assume that a user switches to a new view direction approximately every 30 seconds and watches the special effect every 10 seconds. As shown in Fig. 11(b), the server's CPU load increases approximately linearly when the number of users increases. And the average CPU load for one user is about 0.3% CPU power. On the other hand, in the broadcast mode, the server will not be influenced by the change of the number of users.

4.2 User Interaction Study

A user experience study has been carefully conducted to find out how users perceive the system and when they use the special effects. During the study, we record users' interactions when they use our system to watch some captured sports videos. Four sports actions have been provided in the experiment, including gymnastics and Chinese martial arts.

Fig. 12 shows some user study results where the frequency of the users' subscriptions of frozen moment are provided. In the experiments, more than 10 users are invited to use the system. From the results, we can find:

1. Users really enjoy the features provided by our system. Particularly, they enjoy the frozen moment effect when there is an exciting action in the video. For example, in Fig. 12(a) and Fig. 12(d), the frequency of using the frozen moment is very high when the player makes a curvet.

2. Different users often have a similar judgment about what is an exciting action. Correspondingly, their actions on the subscriptions of frozen-moment are also similar. This can be seen from the peak values in Fig. 12, especially for Fig. 12(a) and Fig. 12(d) where there are several sudden and fancy actions.

3. Videos with different contents bring different using styles of the system. The experiments show that the user response is quite different between the martial arts and the gymnastics. It is because that the gymnastics often are performed smoothly without sudden and fancy actions as in martial arts.

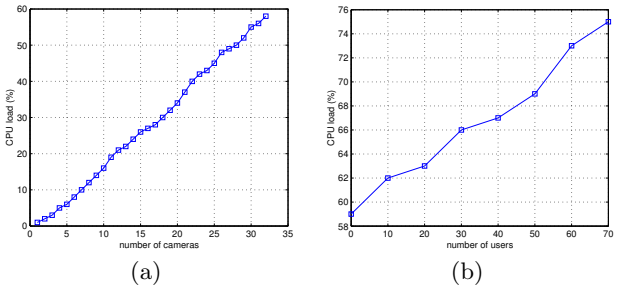


Figure 11: (a) Computational cost vs. number of cameras. (b) Computational cost vs. number of users.

5. DISCUSSION

Throughout the system design, we found several major challenges that should be taken into consideration carefully. Note that these challenges are often inter-dependent. When a certain one is resolved or alleviated, others may become more critical. Therefore, a careful tradeoff has to be made among different challenges.

The first one is **smooth view transition**. The viewing experience of view switching, frozen moment and view sweeping largely depends on the smoothness of view transition, which leads to three system requirements. First, all cameras should be calibrated accurately to shoot at the same event point. Secondly, the view direction should change relatively slowly. Thirdly, the color responses of the neighboring cameras should be as close as possible.

The second one is **interactivity**. A high degree of interactivity not only means users can choose their desired view directions and enjoy some special visual effects, but also means the latency after their actions should be low. Although the propagation delay of the communication channel contributes part of the system latency, the dominating factor of the latency depends on the way how the multi-view video contents are organized and delivered.

Then, the system should handle high **computational complexity**. The whole system consists of many computationally intensive modules, such as camera calibration, object tracking, video compression, channel coding, and multi-

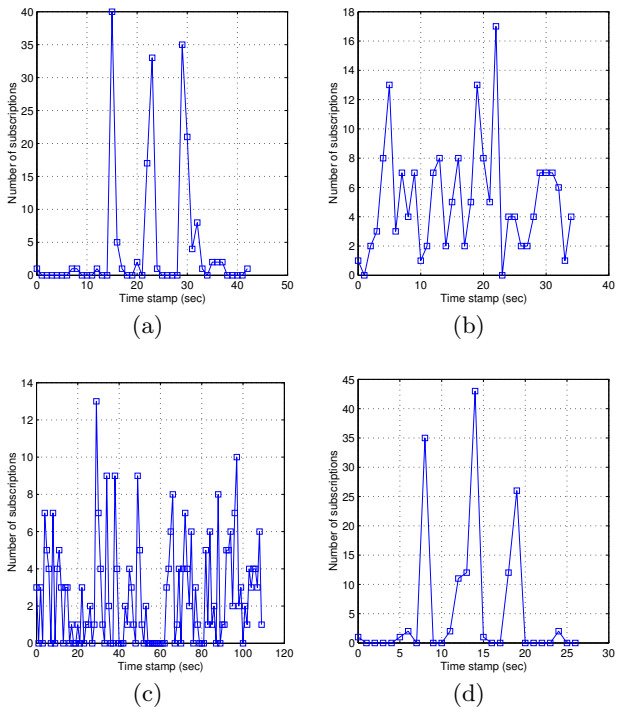


Figure 12: Frozen moment requirement frequency (a) frequency of martial art played by one player. (b) frequency of gymnastics played by one player. (c) frequency of gymnastics played by two players. (d) frequency of martial art played by two players.

view image transcoding. Therefore, how to design and allocate these modules appropriately subject to the limited computing resource constraint is an important issue.

Finally, **system scalability** is also an important requirement. From the system adaptation point of view, different numbers of cameras may be required in different application scenarios. Therefore, in addition to providing a given number of view directions, the system should allow easy addition of more view directions when more cameras are adopted. This also makes the above two challenges (interactivity and computational complexity) even more critical.

6. CONCLUSIONS

As the use of cameras becomes more popular, computer processing power becomes stronger and network bandwidth becomes broader, users desire to leverage these advantages to pursue a richer multimedia experience. In response to the increasing requests from users, multi-view video brings in many brand new viewing experiences.

However, it is still a major challenge to provide interactive multi-view video service in real-time. In this paper, we addressed this problem and proposed a system architecture for providing real-time interactive multi-view video service. The system is designed in a distributed way, where it can be naturally classified into three parts: capturing part, server part, and client part. To tackle the challenges of providing real-time interactive service, we designed the main modules including system calibration, object tracking, video capturing, and multi-view video delivery. To the best of our knowledge, this is the first attempt at building a real-time interactive system for multi-view video over IP streaming. In addition, we believe that our work has four main contributions: first, a pattern free calibration algorithm is proposed to dramatically reduce the workload for calibration in a large environment space. Second, a computational-complexity scalable video encoder is proposed for real-time video capturing with the ability of adjusting the encoding speed according to the computing resources. Thus better complexity adaptation and system scalability are achieved. Third, the multi-view video contents are re-organized and re-encoded into a frozen moment stream and a view sweeping stream. Hence the most exciting viewing experience can be provided with low bandwidth consumption. Fourth, a streaming mode and a broadcast mode are proposed for delivering multi-view video in different application scenarios while unique features of multi-view video are still preserved. Future directions include standardizing the system architecture and deploying this kind of systems in various environments.

7. ACKNOWLEDGMENTS

We thank the reviewers for their constructive comments in improving this paper, and our colleague Steve Lin for his proofreading of the paper.

8. REFERENCES

[1] Applications and requirements for 3dav. *ISO/IEC JTC1/SC29/WG11*, N5877, July 2003.

[2] Report on 3dav exploration. *ISO/IEC JTC1/SC29/WG11*, N5878, July 2003.

[3] V. Bhaskaran and K. Konstantinides. *Image and video compression standards – algorithms and architectures*. Kluwer Academic Publishers, second edition, 1997.

[4] R. Blahut. *Theory and practice of error control codes*. Addison-Wesley, 1993.

[5] I. J. Cox, S. Roy, and S. L. Hingoruni. Dynamic histogram warping of image pairs for constant image brightness. In *International Conference on Image Processing*, pages 366–369. IEEE, 1995.

[6] P. Debevec, Y. Yu, and G. Borshukov. Efficient view-dependent image-based rendering with projective texture mapping. *ACM SIGGRAPH*.

[7] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[8] R. Li, B. Zeng, and M. Liou. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4):438–442, Aug. 1994.

[9] J.-R. Ohm and K. Muller. Incomplete 3d representation of video objects for multiview applications. *PC97*.

[10] H. Tao, H. S. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *International Conferece of Computer Vision*. IEEE, 2001.

[11] R. Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *International Conferece of Computer Vision and Pattern Recognition*, pages 364–374. IEEE, September 1986.

[12] S. Wurmlin, E. Lamboray, O. G. Staadt, and M. H. Gross. 3d video recoder. In *Proc. of Pacific Graphics '02*, pages 325–334. ACM, October 2002.

[13] Z. Yang, H. Cai, and J. Li. A framework for fine-granular computational-complexity scalable motion estimation. *accepted by ISCAS 2005*, May 2005.

[14] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.

[15] S. Zhu and K. Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 9(2):287–290, Feb. 2000.

[16] C. L. Zitinick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *SIGGRAPH*, pages 600–608. ACM, August 2004.