THE MOSAIC CAMERA: STREAMING, CODING AND COMPOSITING EXPERIMENTS

Mashhour Solh and Ghassan AlRegib

Georgia Institute of Technology
School of Electrical and Computer Engineering
{msolh,alregib}@gatech.edu

ABSTRACT

The HP FanCamera is a panoramic mosaicking camera that is a composite of 24-imager array system. Streaming the captured video is a challenging problem due to several factors such as the large bandwidth requirements, the limited capabilities of the client's machines, and our desire to provide independent viewing controls for users. In the process of developing an optimal rate controller for the HP FanCamera we developed a client-server framework for multi-camera streaming and performed a set of experiments using various bandwidth allocation schemes. From our preliminary research, we found that sending individual streams of the cameras over the network provides more interactivity to the end users and requires less bandwidth in case the behavior of the end users is aggressive in scene selection. In this paper we present this framework and share the results of our conducted experiments.

Index Terms— multi-camera, multi-viewpoint, camera array, multi-camera streaming, mosaicking, panorama

1. INTRODUCTION

With the rapid improvement in electronic and computing technologies and dropping costs of cameras, researchers are rapidly discovering applications to use multi-imager camera systems to improve users experience beyond what can be offered by a single camera [1][2]. One of these applications is using multi-camera systems to produce an ultra-high resolution video experience [3] [4]. In [5] a high-performance multi-imager camera system was presented. This system was refered to as the HP FanCamera. The HP FanCamera (also shown in Figure 1) consists of 24-camera CMOS capture system built around a direct memory access (DMA) PCI interface that streams synchronized uncompressed Bayerformat video to PC memory through a three-layer tree structure. The captured streams are processed in real time and composited into a single ultra-high resolution mosaic video at the GPU level. To produce resolution that scales linearly with the number of imagers, the system utilizes native resolution by "tiling" their data which requires that imagers overlap only to ensure spatial continuity in the final mosaic [6].

Fig. 1. The HP FanCamera is a 24-imager array mosaicking camera.

The resulting mosaic is 8 megapixel panoramic video which is four times the resolution of a single HDTV video. The system allows joystick control in which the user can digitally manipulate a virtual camera with pan, tilt, zoom, roll and select whole or part of the video. With the unique imaging capabilities of the HP FanCamera, streaming the captured video is a challenging problem due to several factors such as the large bandwidth requirements, the limited capabilities of the client's machines, and our desire to provide independent viewing controls for users. Since the overlap between the camera views is very small the multi-view video coding (MVC) technicques [7] cannot be applied. Therefore, there are two approaches to stream video generated by the HP FanCamera: (1) compositing the images locally and sending them across the Internet to users and viewers, or (2) sending the individual 24 camera streams across the Internet and compositing at the receiver. We will refer to the former process as mosaicking-at-server (*MAS*) and to the latter as mosaicking-at-client (*MAC*).

Although the *MAS* approach is bandwidth efficient, it limits the resolution at receiver end to the images read back form the server's GPU. In contrast, the other approach above, *MAC*, emulates the HP FanCamera ultra-resolution experience remotely. This approach however is not bandwidth efficient ($> 3Gbps$ to stream 24 cameras uncompressed). Under limited total bandwidth constraint, the process of allocating bitrates to different camera streams becomes critical and influences the quality of received images. Driven by this observation, major questions need to be answered: (1) shall the bandwidth be split equally over all the cameras and how would that affect the quality of the composted image?, (2) if the bandwidth is not split equally, how it should be allocated?

In this paper we show how we implemented the second

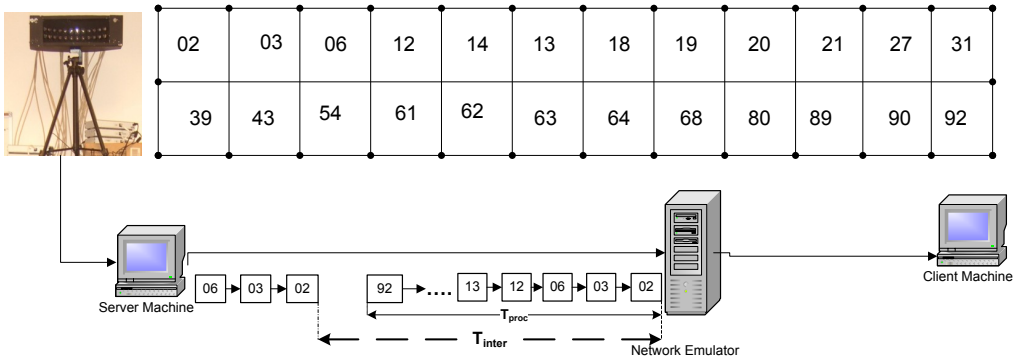


Fig. 2. Experimental Setup.

approach (*MAC*) and performed a set of experiments to answer these questions. In the first set of experiments we split the bandwidth equally over the 24 cameras and in the second set we assumed the user select the six middle cameras as center of focus and we allocate 80% of the bandwidth to these cameras and 20% for the rest. Note that the $80-20\%$ ratio and the six-camera selection were chosen to demonstrate a concept and were not derived according to a solid or scientific basis. The motivation here is to give remote clients the ability of scene selection and provide them with the best experience and most efficient bandwidth utilization. In both cases, the PSNR of the individual cameras and the composted images were calculated.

Over total bandwidths of 168Mbps, 72Mbps, and 12Mbps, the results show an average PSNR improvement up to 7 dB for the selected middle cameras after 80-20% bitrate redistribution. The PSNR of the overall mosaic image after bandwidth redistribution, on the other hand, decreases in the 168 Mbps case while increases in the 72 Mbps and the 12 Mbps cases. Furthermore, we show some results that suggest some relationship between the PSNR of individual cameras and the PSNR of the total mosaic image. This relationship, if found analytically, could help us in developing a rate controller for the HP FanCamera. The rest of the paper is organized as follows. In Section 2 we will present the client-server framework for streaming the HP FanCamera videos. The video coding experiments and the results are shown in Section 3. Finally, a discussion of the results is provided in Section 4.

2. CLIENT-SERVER FRAMEWORK

Figure 2 shows the setup for the HP FanCamera streaming experiments. In order to simulate real network characteristics we installed a network emulator [8]. The emulator we used is *WANem*, which is a wide area network emulator that runs from a bootable CD with Linux Knoppix O/S. *WANem* can be used to simulate WAN characteristics such as network delay, packet loss, packet corruption, disconnections, packet re-ordering, jitter etc . A two 1 Giga-bit network cards are installed on the emulator machines and the emulator is connected to a server and a client machines. The two machines are equipped with Nvidia 8800 ultra with 768 MB memory graphics card . The order in which the camera frames are streamed follows the serial numbers of the cameras as shown in Figure 2 where the time separating two consecutive frames of the same camera is $T_{inter} = \frac{1}{30} secs$ (each individual camera runs at 30 Frames/sec) and the time to send a whole set of 24 streams is $T_{proc} < T_{inter}$. Where T_{proc} and T_{inter} are as shown in Figure 2.

In order to test the behavior of the HP FanCamera and the remote clients' experience in the mosaicking-at-client (*MAC*) case, we designed a framework at both the server and the client that would operate in real time on the HP FanCamera. This section explains this framework on both sides.

2.1. Server Side

In order to send the camera frames in the right order such that $T_{proc} < T_{inter}$, we have implemented a smart application at the server side called the scheduler. The scheduler makes use of two important features, namely the input groups and timestamps. As shown in Figure 3 the camera array is connected to the scheduler through a series of *forward units*. Each *forward unit* has two inputs and two outputs. Every $T_{inter} = \frac{1}{30} secs$ a new set of 24 frames will arrive over inputs connected to the camera array. Each unit has one output connected to timestamp update where the frames are forwarded and another output to the next forward unit where a `FORWARD_OK` flag is passed.

No frame is forwarded by the units 2 through 24 unless media arrives on both inputs. This is achieved by grouping each unit inputs together except for the first *forward unit*. *Forward unit 1* will behave similarly except for the first frame it receives. On startup *forward unit 1* will forward the frame to the timestamp update and a `FORWARD_OK` request to the next unit without waiting for a `FORWARD_OK` flag on its other input. After that the unit will wait for the next flag. The *forward units* will ensure that the frames are streamed

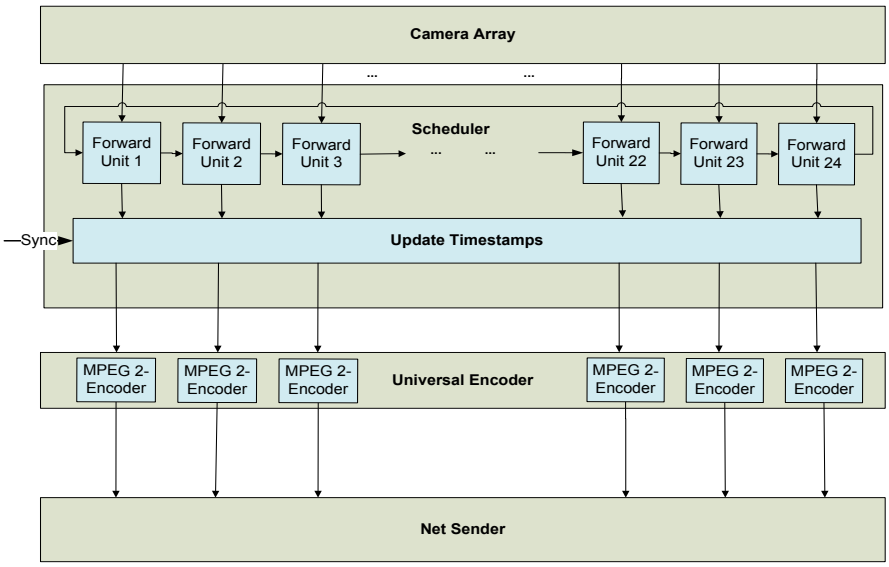


Fig. 3. Server side framework.

in the same order as the camera sequence numbers such that $T_{proc} < T_{inter}$. The next step is to update the timestamps to reflect new streaming process. The very first frame timestamp we call it *Sync* is shared with the stream identifier (offline) at the client side as shown in Figure 3 and Figure 4. The timestamps are updated as follows:

```
nextTimestamp = prevTimestamp + OFFSET;
```

Where the *prevTimestamp* is the timestamp assigned to the last received frame and the `OFFSET` is a constant value that is an estimate of the time between two consecutive frames . This value reflects the server speed and it should not exceed $1.39msec$. Any larger value will result in buffer overflows and loss of data. After updating the timestamps the media are compressed using MPEG2 as shown in Figure 3 and streamed through a UDP port (Net Sender).

2.2. Client Side

At the client side (Figure 4), the Net Receiver listens to the UDP port and captures the stream and forward it to a stream identifier task. The stream identifier uses timestamps to match each media (frame) with the corresponding camera. However, this process needs to be fast and operates properly, especially when the stream experience network losses. For this purpose we have came up with a simple algorithm that does not include any loops or comparisons (see Algorithm 1).

Each camera is identified by an index value *curCamera* and *prevCamera* reflecting its sequence number order. The index for each camera is selected to correspond to the camera sequence number. For example, the top-left camera has the lowest sequence number and, therefore, its index is 1. Similarly the next camera with the next higher sequence number is 03 and, therefore, its camera index is 2 and so on. The initial timestamp and *OFFSET* values are shared with the server application using Sync. The other variables are *curTimestamp*, which is the timestamp of the currently received media, and *prevtimestamp*, which is the timestamp of the previously received media. This algorithm is efficient and accounts for lost frames when there are packet losses in the network.The other tasks shown in Figure 4 are the array generator and the update timestamps. The array generator simply forwards each frame on the corresponding camera pin using the index value. The update timestamps undoes the server timestamp update by subtracting the *OFFSET* value. Each camera stream is then decompressed and the compositor tasks will composite the streams by performing a panoramic mosaicking [9]. The composted stream is then sent to a sink. The sink can be either a display or save-to-disk application.

Algorithm 1 Stream identifier aglorithm

```
DIFF = (curTimestamp-prevTimestamp-OFFSET)/OFFSET;
curCamera = mod(prevCamera+DIFF, 24);
prevTimestamp = curTimestamp;
prevCamera = curCamera;
```

3. CODING EXPERIMENTS

We have performed two different sets of experiments over three bandwidth values 168Mbps, 72Mbps and 12Mpbs. In the first set of experiments, we distribute the bandwidth equally over the 24 cameras. In the second set, we assume the user selects the six middle cameras as center of focus and

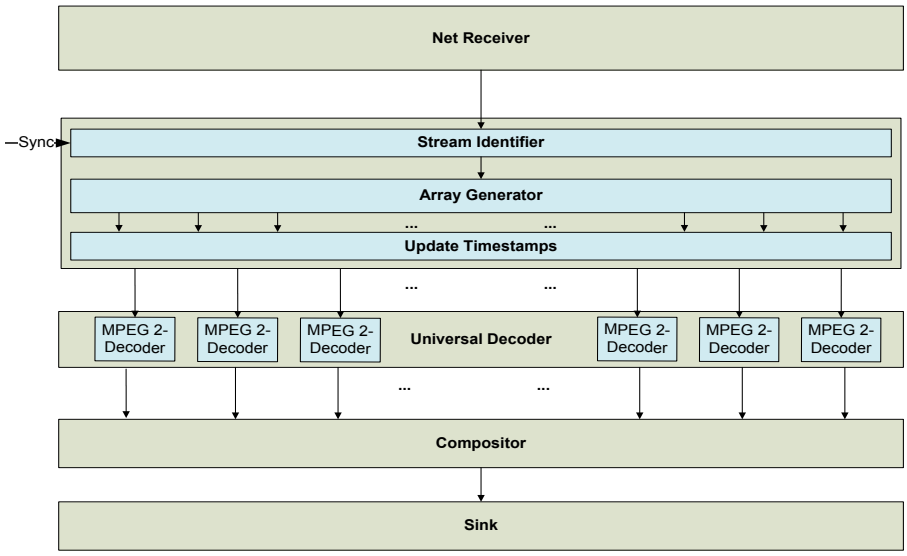


Fig. 4. Client side framework.

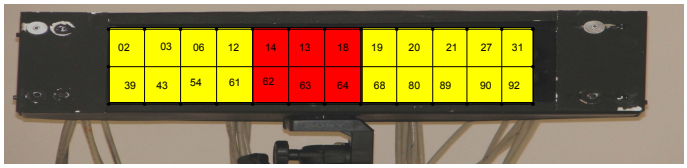


Fig. 5. The bit rate allocation in the uneven distribution coding experiments. The six middle cameras (in red) are considered the center of focus are allocated 80% of the bandwidth and 20% for the other cameras (in yellow).

we allocated 80% of the bandwidth to these cameras and 20% for the rest. The middle cameras are the ones shown in red in the Figure 5. The rationale behind allocating 20% to the other 18 cameras is to provide smooth transition as the user select another scene or zoom out of the 6-camera focus.

We have conducted the experiments with various video streams, including different kinds of motion content and different backgrounds. In this paper we have chose to include results for the two extreme cases, i.e. for scenes with no motion and scenes with a lot of motion. The complete set of results however, is consistent with the ones are presented in the next two subsections.

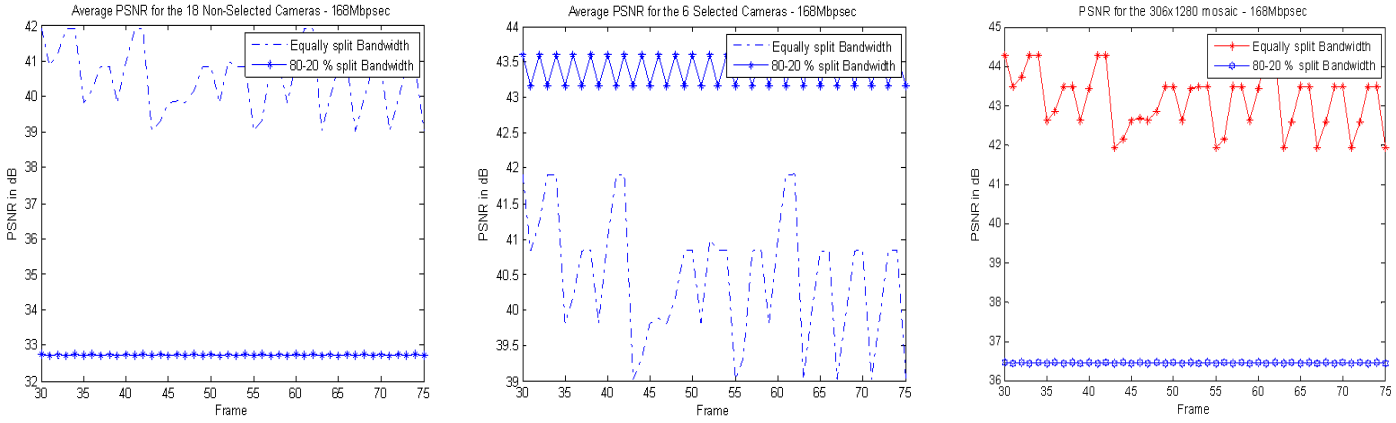
3.1. Uniform bitrate distribution vs. weighted bitrate distribution (no motion scenes)

We ran the experiments over an error-free network and a delay of $150msec$. These parameters were set in the network emulator (Figure 2). In the results shown in Figure 6, Figure 7, and Figure 8 the same motion-free sequence is used in all the cameras which make it easier to draw comparisons and relations between individual images/video and the composed image/video. Figure 6, Figure 7, and Figure 8 show the PSNR of the composed frames at the client side for a subset of the frames, i.e., frames $30 - 75$.

The results in Figure 6 show an average PSNR improvement of $3dB$ for the selected cameras and an average loss of $7dB$ for the total mosaic image/video resulted from redistribution of bitrate by $80 - 20\%$ ratios. The average PSNR losses for the non-selected cameras are $10dB$ which explain the losses in the mosaic frames. Figure 7 shows the results for the $72Mbps$ case. These results show an average PSNR improvement of $7dB$ for the selected cameras and an average loss of $3dB$ for the total mosaic image/video. Comparing these results with Figure 6, we notice that the $80 - 20\%$ ratio redistribution yields better performance when the available bandwidth is $72Mbps$ compared to the 168Mbps case. This is true for both the selected cameras and the mosaic image/video.

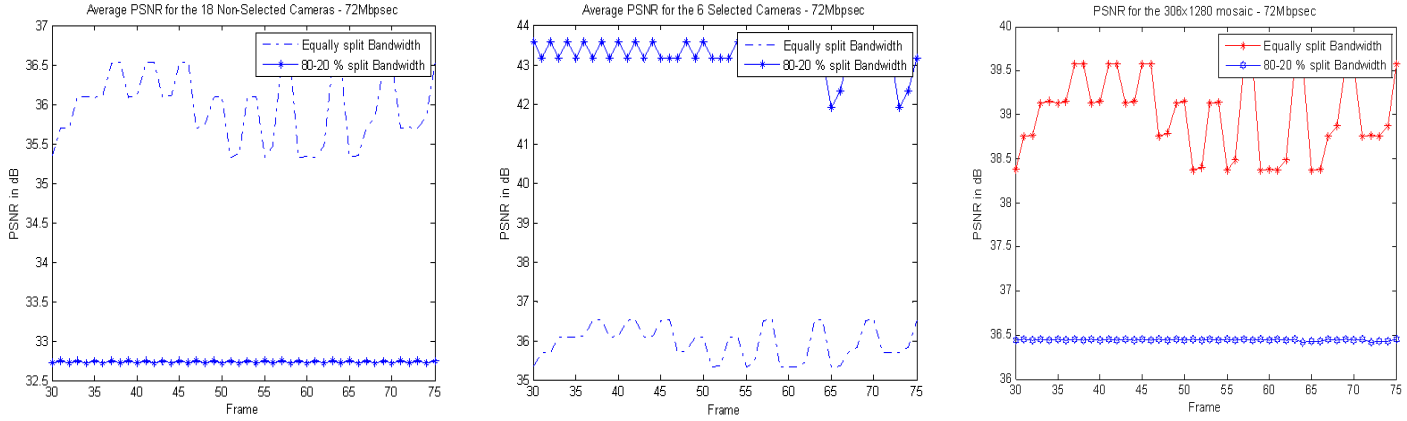
At $12Mbps$ (Figure 8), the losses due to bitrate redistribution are very insignificant in the non-selected eighteen cameras. The PSNR of the scenes captured by the selected six cameras, however, improves by an average of $3dB$. This results in a $2dB$ improvement of the mosaic image/video.

We have conducted the experiments with H.264 as well as MPEG2. The conclusions and findings were quite similar in both cases. The main motivation behind using MPEG2 was the original implementation of this camera and system within a testbed to deliver multi-view-based distance learning across Georgia Tech campuses [10]. The testbed was based on MPEG2 and thus our empirical study in this paper is based on MPEG2.



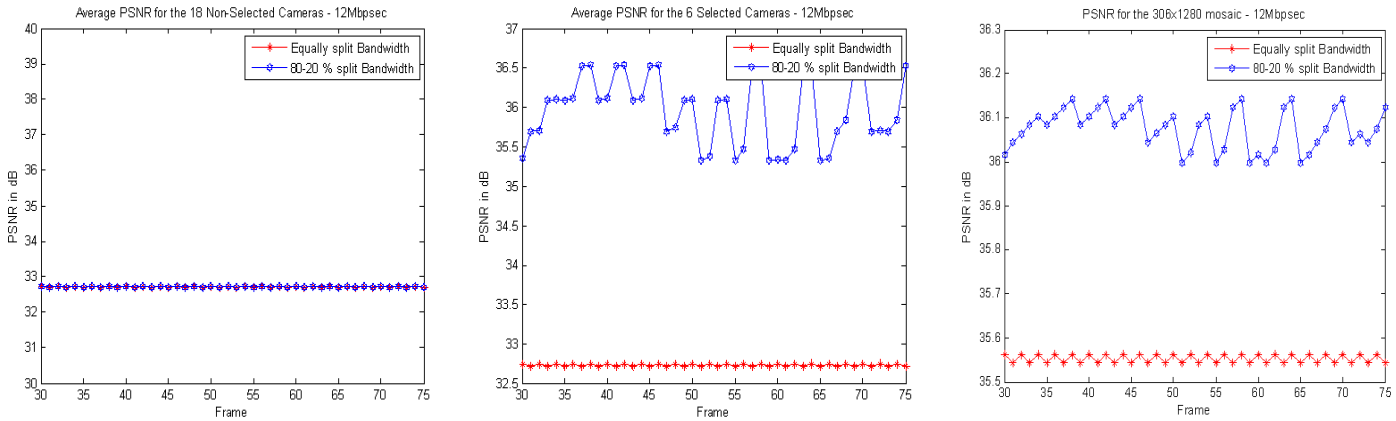
(a) (b) (c)

Fig. 6. PSNR for total Bandwidth -168Mbps :(a) Non-selected cameras (b) Selected cameras (c) 1280×306 mosaic video.



(a) (b) (c)

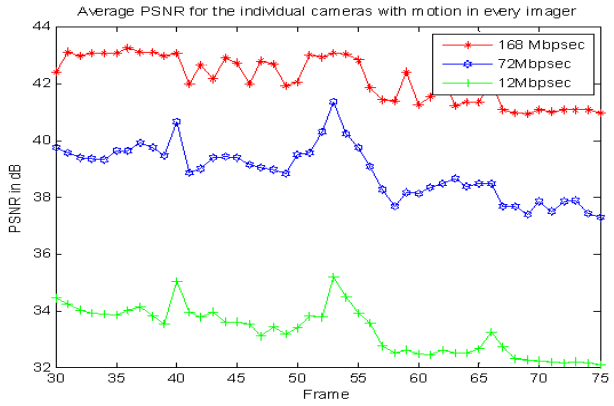
Fig. 7. PSNR for total Bandwidth -72Mbps :(a) Non-selected cameras (b) Selected cameras (c) 1280×306 mosaic video.



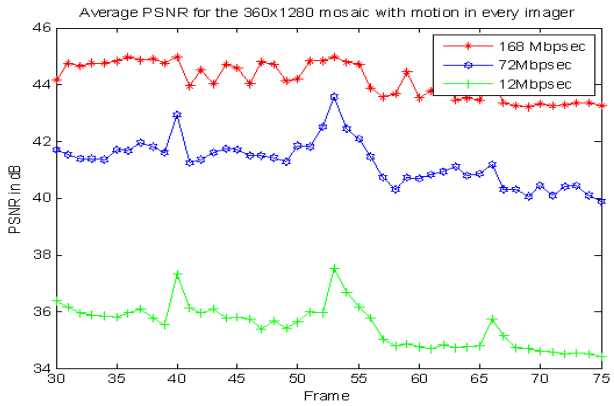
(a) (b) (c)

Fig. 8. PSNR for total Bandwidth -12Mbps :(a) Non-selected cameras (b) Selected cameras (c) 1280×306 mosaic video.

3.2. Uniform bitrate distribution for motion scenes



(a)



(b)

Fig. 9. Average PSNR: (a) Individual cameras (b) 1280×306 mosaic video.

In this part of the experiments, we investigated the relationship between the PSNR of the video produced by each individual camera and the PSNR of the composed video. In these experiments, we feed each camera a video sequence with motion. The total bandwidth is evenly distributed over all the 24 cameras. We ran the experiments for three different bandwidths: $168Mpbs$, $72Mpbs$, and $12Mpbs$. The resulting PSNR values are shown in Figure 9(a) and Figure 9(b). Figure 9(a) and Figure 9(b) show that the mosaic image tends to follow the same trends of individual cameras with $2dB$ gain in the PSNR value for the total mosaic over individual camera. This $2dB$ difference is due to the fact that there is a minimal overlap between adjacent scenes.

4. DISCUSSION

In the process of developing an optimal rate controller for the HP FanCamera we developed a client-server framework for multi-camera streaming and performed a set of experiments using various bandwidth allocation schemes. The results have shown that there is always a gain in PSNR for the selected cameras in an $80-20\%$ split over an equally split bandwidth as the total bandwidth drops down from $168Mbps$ to $12Mbps$. The PSNR of the mosaic image/video, on the other hand, may increase for high total bandwidth values or decrease for lower total bandwidth values. This translates into concluding that the $80-20\%$ ratio is more efficient for certain bandwidth values such as $72Mbps$ and $12Mbps$ compared to $168Mbps$. This indicates the available bandwidth should also influence our choice of redistribution ratio. On the other hand,understanding the relationship between the PSNR of individual imagers and PSNR of the total mosaic may lead to a smarter allocation of the bandwdith in the future. Our current work focuses on cases where the selected area includes distant cameras. An example of that would be having an instructor writing on a white board and projecting slides that are at two distant areas of the wall. With multiple selected areas at a distance, a two-way split of the bandwidth as in $80-20\%$ may not be efficient and we might need an $n-way$ split $(n>2)$ such as $50-30-20\%$.

5. ACKNOWLEDGEMENTS

We would like to thank HP Labs on providing us with this camera and allowing us to access the codes to operate the camera. Their support throughout the operation of the camera is highly appreciated. Special thanks to Ronald W. Schafer, Harlyn Baker, and John Apostolopoulos at HP Labs.

6. REFERENCES

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multiview Imaging and 3DTV," *IEEE Signal Proc. Magazine*, vol. 24, no. 6, pp. 10–21, Nov 2007.

[2] "ISO/IEC JTC1/SC29/WG11 Applications and Requirements for 3DAV," Tech. Rep. N5877, Trondheim, Norway, July 2003.

[3] Bennett Wilburn, *High-performance imaging using arrays of inexpensive cameras*, Ph.D. thesis, Stanford, CA, USA, 2005.

[4] Donald Tanguay, *The mosaic camera: synthesizing a large-format camera from many smaller cameras*, Ph.D. thesis, Stanford, CA, USA, 2006.

[5] H. Harlyn Baker and Donald Tanguay, "Multiviewpoint uncompressed capture and mosaicking with a high-bandwidth pc camera array," in *Proc. Workshop on Omnidirectional Vision (OMNIVIS)*, 2005.

[6] H. Harlyn Baker and Donald Tanguay, "A multi-imager camera for variable-definition video (xdtv).," in *MRCS'06*, 2006, pp. 594–601.

[7] Yo-Sung Ho and Kwan-Jung Oh, "Overview of multi-view video coding," in *6th EURASIP Conf. on Speech and Image*, june 2007, pp. 5–12.

[8] "Wide area network emulator(wanem). http://wanem.sourceforge.net/," .

[9] Peter J. Burt and Edward H. Adelson, "A Multiresolution Spline with Application to Image Mosaics," *ACM Trans. on Graphics*, vol. 2, pp. 217–236, 1983.

[10] G. AlRegib, M.H. Hayes, E. Moore, and D.B. Williams, "Technology and tools to enhance distributed engineering education," *Proceedings of the IEEE*, vol. 96, no. 6, pp. 951 –969, june 2008.