# A REAL-TIME SYSTEM FOR HDTV VIDEO TRANSMISSION OVER IP NETWORKS

Konstantia Moirogiorgou[1], Grigorios Th. Tsagkatakis[2], Michalis Zervakis[1], Euripides G.M. Petrakis[1]

[1] Department of Electronic and Computer Engineering Technical University of Crete (TUC), Chania, GR-73100, Greece
{dina, michalis}@display.tuc.gr, petrakis@intelligence.tuc.gr
[2] Center for Imaging Science, Rochester Institute of Technology, NY, 14623
gxt6260@rit.edu

## ABSTRACT

We present an end-to-end server–client system for high-resolution video transmission over IP networks. The system supports communication of compressed information, user operations through a fully functional user interface as well as detection and tracking of moving objects/people. It relies on a communication subsystem which has been developed for real time transmission of MJPEG / MJPEG2000 compressed video information over erroneous out-of-order packet-based local and wide area networks. The images are at the HDTV standard resolution. The server grabs frames from a camera-link sensor, encodes the raw video, places the encoded bit stream into packets and transmits them over network. The client receives and reorders the packets and then displays the video data. Each user can request different focus windows from the server according to the viewpoint of interest. Each encoded frame is filled with restart markers in order to face up the case of a frame been lost.

## KEY WORDS

Surveillance, HDTV transmission, real-time compression, video sharing, window specification.

## 1. INTRODUCTION

Whilst there have been considerable developments in the area of image based system for the semi-automatic identification of individuals and the recognition of unusual patterns of movement these have, to date, been limited to analysis within the field of view of a single camera. The proposed architecture seeks to extend the coverage of such systems by extending the field of view of the individual camera systems and link multiple camera systems together.

The real-time transmission of image and video data over the Internet has been identified as one of the most demanding applications imposing certain constraints on processing power, minimum delay, minimum package loss, and high bandwidth for transmission up to 20 frames per sec (fps) of high resolution HDTV (1280x780x24bit/pixel or even higher) color image and video [1]. It is becoming an important building block of numerous applications, such as Internet television, video conferencing, distance learning, digital libraries and video-on-demand. However, the Internet itself does not provide any mechanism for Quality of Service (QoS) guarantee. In the case of multicast video transmission, the heterogeneity inherited in the Internet creates more problems in terms of bandwidth efficiency and service flexibility [2].

In this study we introduce a fully operational, end-to-end server – client system for video transmission over network. The system allows for fast transmission of high resolution compressed image and video information over the Internet. It also performs motion tracking, aiming at surveillance applications. Such an autonomous system is able to identify anomalous or unexpected situations and can help a human operator to observe an open space in real-time conditions.

As the proposed system can be used for surveillance purposes, certain constraints for minimum information loss and zero latency reconstruction of transmitted video information must be followed (so that the video can be viewed at the monitor in real-time). The delay constraint favors the choice of MJPEG for the encoding of video, attached with a proposed error concealment algorithm [3].

The need for simultaneous broadcast of a full resolution images and multicast of zoomed parts of the images, according to the client request [4], reinforced the implementation of a new protocol. Our system is build upon the UDP transfer protocol for image and video broadcasting and TCP for communication of control information.

Each client can view the video data requested from one or more servers simultaneously on a user interface implemented in C#. The interface can display one or more camera (server) data and has appropriate menus/buttons in order to zoom in to a region of interest and to archive the video on user's demand.

The proposed communication protocol, in context with our system architecture, is discussed in the next section, along with our error concealment strategy and the motion tracking algorithm. Test results are presented in section 3 and the paper concludes in section 4.

## 2. SYSTEM ARCHITECTURE

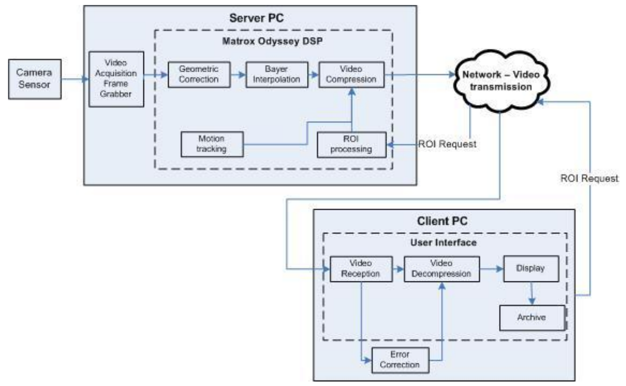
The overall system is a multiple server – multiple client system [5]. The server transmits the video data to the clients that requested the available video information and each client may request video data from multiple servers on the network. Each server acquires high resolution HDTV image data at a maximum of 25fps. The communication between the acquisition system and the processing system of the server is implemented through a frame grabber communicating synchronously with the camera sensor using camera link protocol. Video processing takes place on a Matrox Odyssey XPRO DSP board. This board is comprised of an array processing node (pixel accelerator), a Motorola G4 PowerPC microprocessor (MPC7455 featuring RISK technology) and a frame grabber module for image acquisition. The Matrox Odyssey can exchange data with the Host computer at up to 1 Gbyte/sec via a PCI-X slot.

The software of the camera system is implemented in C using the Matrox Imaging Library (MIL) and the Odyssey Native Library (ONL). MIL functions run on the Matrox Odyssey and are fully optimized for the G4 PowerPC. ONL is also a C-callable programming language more dedicated to the Matrox Odyssey board. ONL is designed to run solely on this type of processor.

### 2.1. The Server – Client Model for Video Communication

The system architecture is presented in Figure 1.

**Figure 1: Block diagram of the system architecture**



After each frame is grabbed from the server's camera sensor, geometric correction is performed, i.e. a warping by associating each pixel position to real-world coordinates. Bayer interpolation can also be performed for color generation, as to avoid the use of a high-cost color RGB camera sensor. Finally, the image is compressed and broadcasted to the clients with a frame rate close to the camera sensor's frame rate, so that we can talk of a real-time application.

Each client can request video data from any server that is available on the network. If a packet of the requested image data is lost during transmission the client uses an error correction method to estimate the missing packet and create the lost frame.

### 2.2. Window Specification

The client user has access to the entire image acquired from the server. In addition, the user is able to focus on a specific portion of the image, simply by selecting the desired region of interest (ROI). This is done simply by making two successive mouse clicks on the panoramic (whole) image. In that case, the User Interface is divided in two parts: the entire, panoramic image acquired by the camera server and the user-selected window. The two mouse clicks the user performs on the client's screen are translated in point coordinates, in the (start_x, start_y) and (end_x, end_y) pairs of coordinates that imply the ROI position in the initial server image. These coordinates are transmitted back to the server as a separate image request. The difference of this operation from just making a simple zoom in the image that he/she observes concerns image analysis. Zoom "degrades" partly the whole image, while, if the user request from the server a smaller portion of it, server sends the ROI in the same analysis with the one of the initial, entire image. The downside of this approach is that the continuous use of this operation by the user, with different sizes of image each time will cause a reduction of the transmission rate of data from the server to the client, because each time server will be supposed to commit different size buffer in order to processes the ROI image.

### 2.3. Data Compression

Data compression on camera system (server) prior to transmission (and decompression on client) is considered as an option for reducing bandwidth requirements. Since compression should not impose latencies, we adopt the MJPEG standard [6, 7]. MJPEG is just a necessary compromise offering real-time viewing at the expense of compression efficiency (MPEG-2, 4 support much higher compression rates) at the cost of latency.

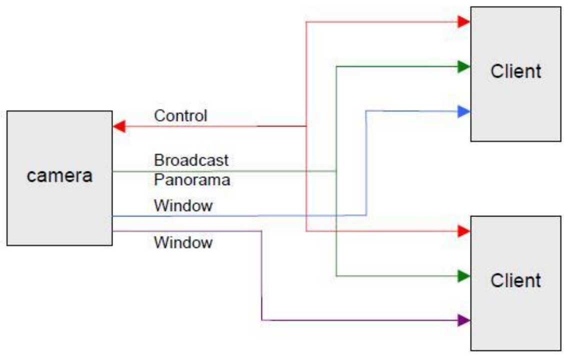
In our implementation, the image is divided into 8 x 8 blocks called segments. Each UDP packet can contain a number of segments. These segments are distinguished using Restart Markers, as mentioned later. Although the number of blocks, segments and Restart Markers are fixed, none of them have a constant size in terms of bytes. This variation is a consequence of the entropy encoding stage in the JPEG compression algorithm [8].

In the case of MJPEG, when a newly received image raises an exception such an error code, the reconstruction function at the host takes over. When a packet is lost during transmission, a portion of the image is lost. Due to the entropy encoding of the image, a missing part of the bit stream can result in total destruction of the image. There are three kinds of concealment algorithms namely the spatial, the temporal and the hybrid as described in [3, 9], that can be used for concealing the missing information. Hybrid techniques are based on an effort to combine both spatial and temporal concealment algorithms, in order to improve their performance. In our implementation, we follow a simple temporal based technique because of its efficiency related to surveillance video (little change over successive frames) and its low computational complexity. This choice was made upon examining the video in question and the real-time constraints set by our model. The key feature used in implementing our concealment technique was the use of Restart Markers. A restart marker is a special code that signifies that the encoded bit-stream has been padded to the next byte boundary before the encoding process restarts. Restart markers are useful for transmission the compressed images over a medium that is susceptible to errors, such as a network. If an error occurs and there are no restart markers, the error will propagate and effect subsequent data. With the use of restart markers, the error will be confined to the data between markers [10]. In our case, having located the first missing segment and the number of missing segments, we can reconstruct the frame based on previously received packets. When a packet is missing we search the packets received from the previous frame and we locate the missing segment in the previous frame. By knowing the first missing segment and the segments missing, we just copy the corresponding image section from the previous packets.

## 2.4. Communication System

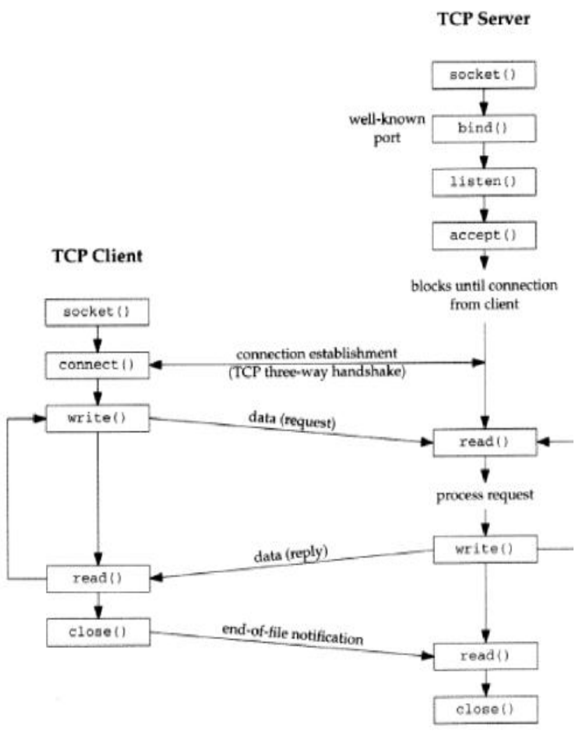
The communication system consists of the transmission protocol for the server's raw data, sends the packets over network and re-orders the packets as they arrive to the client [11]. It also supports an error – correction algorithm that is used in the case of missing packets.

**Figure 2: Block diagram of TCP server-client communication**



A TCP connection is first established between the server and the client prior to video transmission. Towards that end, the client listens to a TCP port for incoming connection. When the client sends a request, a TCP and UDP connections are established. The former is used for control purposes, where the latter is used for the transmission of the video stream. The communication of the video is performed using UDP sockets. The packets may arrive out of order, which implies that care must be taken to support the re-ordering of the packets as they arrive. In order to facilitate these requirements, the image is partitioned according to the restart markers appropriately placed in it.

**Figure 3: Block diagram of TCP server-client communication**



## 2.5 Motion Tracking

Object detection in our system is based on subtracting frames from an "empty background", which has the advantage of detecting moving objects, objects that temporarily stopped moving, or still objects which do not belong to the predefined background. Due to similarities between the object and background intensities, objects are often split into pieces at regions of such similarities. In order to alleviate this problem, we use detailed edge (color edges) information from such regions in the frame under consideration, as to enhance local region contrast and enable the discrimination between object and background.

For the initialization of the algorithms, the background is estimated by using a large number of pictures and the median filter operating in the temporal axis. The background image is then updated in regular time intervals. Using the knowledge of the detected objects in a scene, it is possible to define a selective update, which computes a new background value only if a point is not marked as an object pixel. The new background is the weighted average between the current background objects and the old background.

$$B_j = (1-\alpha)B_j + \alpha I_i$$

where Bj is the jth updated background image (j = i mod K), Ii is the current image, and α (0 ≤ α ≤ 1) is a scalar representing the importance of the current data and the learning rate of the model. K controls the update frequency of the background to avoid the updating for every frame (for the tests K=20 was chosen).

Two different types of background updating are considered:

- The *blobs algorithm* uses the blobs of the detected objects and updates according to equation 1 the background around these blobs. The background covered by the blobs is not updated and remains equal to the last updated background.
- The *bounding box algorithm* behaves like the blob algorithm with the slight difference that the background around the computed bounding rectangle of each object is updated. In this case the updated area is smaller than ideally, excluding the entire bounding box rather than the object itself, but the implementation is simpler.

In both cases wider boundaries are chosen in order to avoid calculating shadows or similar background colored objects in the updated image. The blobs algorithm appears to be the better, as more background can be updated in each step. Moreover, it is not dependent on object orientation compared to the bounding box algorithm.

### 2.6 User Interface

Via the user interface the user of each station of control (client) can select and check the camera (server) he/she desires. The user interface allows operations of shift (pan/tilt), zoom in/out or even motion tracking operations, i.e. watches the movement of an object or person.

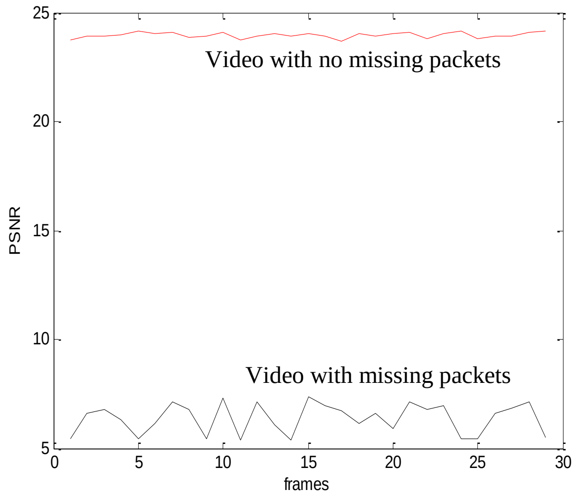
The main role of the user interface is to build a total set of low level operations that should support the communication between servers and clients. It is developed to communicate (connection/disconnection) with one or even more servers (that are each time available in the network), to send the demands of clients for the region of interest which they want to have the supervision, to receive the data of the entire image that it recorded by each server but also to store the image information in file.

The User interface is also capable of archiving the transmitted video upon user demand. This operation stores the individual frames of the video on the hard disk of the client computer as jpeg images, so that they occupy less disk space. The file names contain extra information, i.e. which server sent the frame.

## 3. EVALUATION AND RESULTS

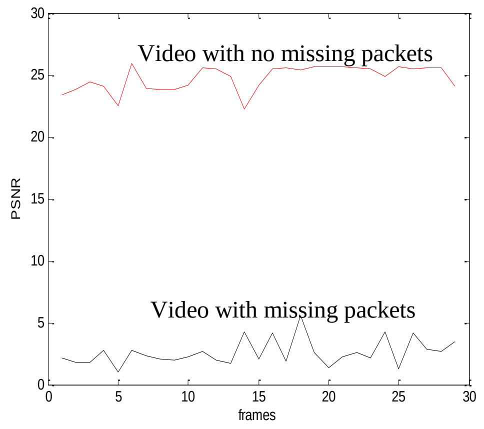
First we present the results of the error correcting mechanism providing figures relative to the algorithmic performance of the video transmission system. We examine two types of video, i.e. low motion and high motion content. The videos are encoded with MJPEG compression with the option of restart marker enabled. In both cases we examine the robustness of the transmission and error correction algorithm in terms of increase in the received video SNR, when 6% of the packets are randomly lost. This test reveals how packet loss can affect the quality of the received video. The scenario involves 6% missing packets in low motion video and Figure 2 presents error measurements on the received and reconstructed video. There is an obvious improvement of the video quality achieved by the correction scheme in the case of missing packets. Similar effects have also been observed in the case of 3% and 10% packet loss.

**Figure 4: Performance of reconstruction with little motion**



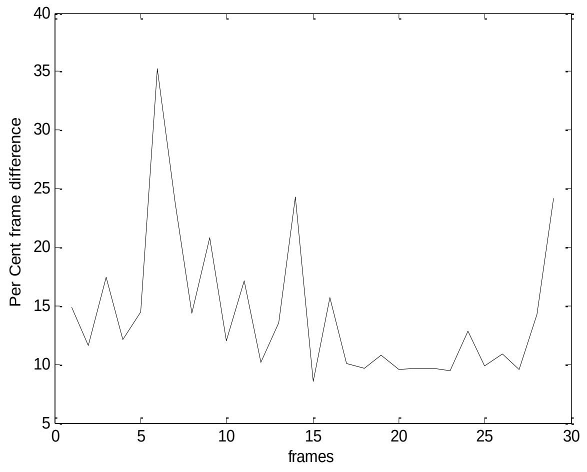
In Figure 3 we examine the quality of the received video when there is significant motion. The amount of motion is measured in terms of frame-to-frame bit difference (per cent), as shown in Figure 4. The implemented error-concealment algorithm performs well, even in the case of increased motion. The motion of the video is not heavily reflected on the quality of the reconstructed image in terms of the peak signal to noise ration (PSNR). The PSNR values tend to "follow" the motion activity, but not to the same extend. The reason for this effect is that when the error concealment algorithm attempts to reconstruct the missing frame, it uses as reference the last correctly received frame. Differences caused by motion lead to lower quality of the reconstructed frame.

**Figure 5: Performance of reconstruction with motion**



In terms of processing the video (grab, correct, compress and transmit), the system was able to support broadcasting of full size images, i.e.1280x780x24bit/pixel at 10 fps. In addition, two users can request zoomed parts of the full camera image in 1024x768 x24bit/pixel resolution transmitted at 20 fps frame rate.

**Figure 6: Percent changes in the bit stream of 30 successive frames from Fig. 4**



The performance of the system needs to be carefully checked with the option of object/motion detection and tracking, since the tracking algorithm can slow down its operation. More specifically, the execution times on the DSP and the PC are reported in Table 1. The following results are preliminary, we are working towards the improvement of the overall system performance. Our main focus is to reduce the processing time of the object detection algorithm, since this is the section that has the greatest impact on the entire system performance.

**Table 1: Motion Tracking algorithm results**

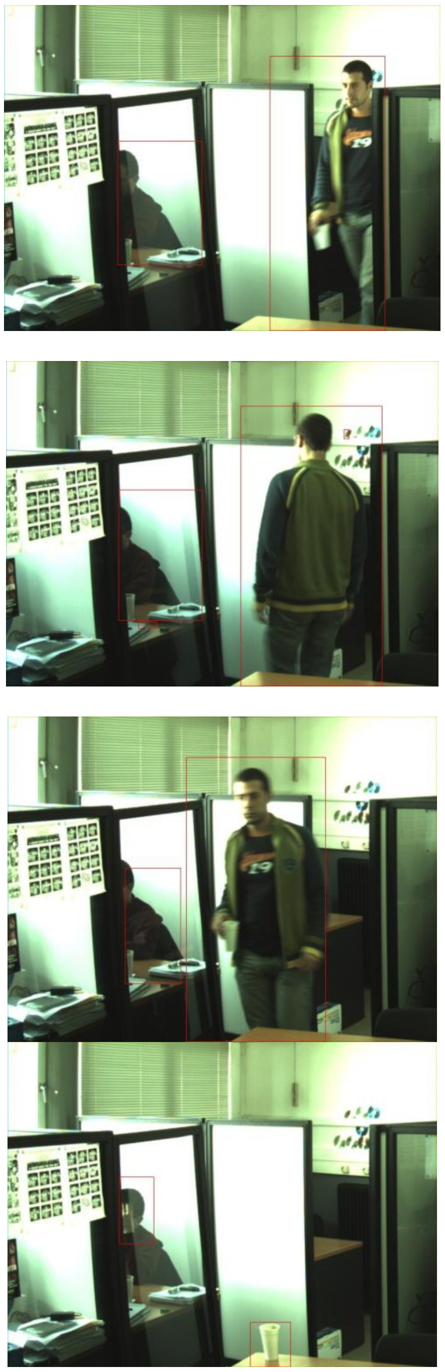
| **Object detection** | **On server (on Matrox DSP)** | **On client (on custom PC)** |
|---|---|---|
| Estimated time for full-sized images | 0.07s | 0.13s |
| Frame rate (for full-sized images 1280 x 780 x 24bit/pixel) | ~13fps | ~7fps |
| Frame rate (for ROIs 800x600x24bit/pixel) | ~20 fps | ~11 fps |

**Table 2: System operation results**

| | |
|---|---|
| **System test features:** | |
| - *'Panoramic' image: 1024x768* | |
| - *'ROI' – window image: 400x300* | |
| - *LAN type: intranet (100Mbps) / Internet* | |
| - *Motion: High* | |
| - *MJPEG compression rate: 30% - 60% - 90%* | |
| Server execution time: | 30 – 35ms |
| Ethernet load | 3-5Mbps |

Figure 7 illustrates detection and tracking of persons (in rectangles) in a frame sequence.

**Figure 7: Motion tracking and detection**



Finally, the overall system with one server – one client performed 95% compression ratio without obvious image quality reduction and 50% load on the pixel accelerator of the Matrox DSP.

The novelty of this work relies on the development of a complete surveillance system with full functionality. The proposed system generates and processes color images of high resolution, achieves high transmission speed due to high compression rate and can perform detection of movement and follow-up of objects along with selective follow-up and transmission of regions of interest.

## 4. CONCLUSIONS AND FUTURE WORK

In this work we introduce a fully operational, end-to-end system for video transmission over network. We examined a real-time end-to-end MJPEG video transmission mechanism for packet-based networks, such as the Internet.

The system is based on the client-server model and has been tested for HDTV video source. In order to minimize the effect of packet loss in the case of MJPEG, we implement a real-time low complexity error correcting mechanism. In the following, special attention is given to the video communication subsystem and its related protocol. The proposed error-concealment algorithm for the MJPEG performs well when there is little or no motion in the video. In videos with extensive motion, the low temporal redundancy quality of the video renders temporal based error-concealment inadequate.

The novelty of this work relies on the development of a complete, fully functional system which can be useful in surveillance applications. The proposed system generates and processes color images of high resolution, achieves high transmission speed due to high compression rate and can perform detection of movement and follow-up of objects along with selective follow-up and transmission of regions of interest.

In order to achieve better quality of the received video, spatial based error concealment techniques can also be introduced. Future implementations should support higher compression ratio and improved video quality. This need is more evident in the case of wireless networks where there is a considerable bandwidth limitation. Recent advances in hardware implementations of “state of the art” compression algorithms such a H.264 [12] should be also examined.

## 5. REFERENCES

[1] GA Jones, JM Defilippis, H Hoffmann, EA Williams, “Digital Television Station and Network Implementation”, - Proceedings of theIEEE, Vol. 94, Issue 1, pp 22- 36, January 2006.

[2] Dapen Wu, Yiwei Thomas and Ya-Qin Zhang,”Transporting Real-Time Video over the Internet: Challenges and Approaches”, Proceedings of the IEEE, Vol. 88, No 12, December 2000.

[3] C Sacchi, F Granelli, CS Regazzoni, F Oberti , “A real-time algorithm for error recovery in remote video-based surveillance application”, in Signal Processing: Image Communication, Vol. 17, Issue 2, February 2002, pp 165-186.

[4] Gian Luca Foresti, Christian Micheloni, Lauro Snidaro, Paolo Remagnino, and Tim Ellis, “Active Video-Based Surveillance System”, IEEE SIGNAL PROCESSING MAGAZINE MARCH 2005.

[5] Lucio Marcenaro, Franco Oberti, Gian Luca Foresti, Senior member IEEE and Carlo S. Regazzoni, Senior Member, IEEE, “Distributed Architectures and Logical-Task Decomposition in Multimedia Surveillance Systems”, PROCEEDINGS OF THE IEEE, VOL. 89, NO. 10, OCTOBER 2001.

[6] Gregory K. Wallance, “The JPEG Still Picture Standard”, Communications of the ACM archive, Vol 34, Issue 4, pp 30 - 44, (April 1991).

[7] Skodras, A. Christopoulos, C. Ebrahimi, T., “The JPEG2000 still image coding standard” , Signal Processing Magazine, IEEE, Vol 18, Issue: 5, pp 36-58, Sep 2001.

[8] Tien-Hsu Lee, Hsiu-Hua Hsu, and Pao-Chi Chang, “Restart Marker Regulation Technique for Progressive JPEG Image Coding in Mobile Communications”, IEEE COMMUNICATION LETTERS, VOL. 4, NO. 12, DECEMBER 2000.

[9] Yh Han, JJ Leou, “Detection and Correction of Transmission Errors in JPEG Images”, in Circuits and Systems for Video Technology, IEEE Transactions on, Volume: 8, Issue: 2, on page(s): 221-231, Apr 1998.

[10] Matrox Imaging Library, Version 7, User Guide, Matrox Imaging.

[11] Toufik Ahmed, Ahmed Mehaoua, Raouf Boutaba, Member, IEEE, and Youssef Iraqi, “Adaptive Packet Video Streaming Over IP Networks: A Cross-Layer Approach”, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 23, NO. 2, FEBRUARY 2005.

[12] Shao-Yi Chien Yu-Wen Huang Ching-Yeh Chen Chen, H.H. Liang-Gee Chen, “Hardware Architecture Design of Video Compression for Multimedia Communication Systems”, in Communications Magazine, IEEE , Vol 43, Issue: 8, pp 122- 131, Aug. 2005.