A DISTRIBUTED FRAMEWORK FOR RELAYING STEREO VISION FOR TELEROBOTICS

M. Al-Mouhamed, O. Toker, A. Iqbal, and M. Nazeeruddin

College of Computer Science and Engineering
King Fahd University of petroleum & Minerals
Dhahran 31261, Kingdom of Saudi Arabia.
mayez/onur/aiqbal/nazeer@ccse.kfupm.edu.sa

ABSTRACT

Stereo vision is one critical tool in minimally invasive surgery (MIS) for enhancing perception of depth of organs which greatly improves the operation efficiency. Augmented stereo views results from superimposing 3D anatomical models with real organ views. A reliable distributed framework for relaying stereoscopic visual feedbacks between a telerobotic server and a client station is proposed. The distributed components are based on DirectX, Visual C#, and Window sockets. We used a multi-threaded execution to promote concurrency in grabbing, transmitting, receiving, processing, and displaying image data using head-mounted display (HMD) technology. The client station provides components that support augmented reality (AR), i.e. superimposing animated graphic model with real views from the operating site. Design and performance issues of proposed multi-threaded execution for streaming of stereo data in a distributed and modular framework is presented.

Keywords: Augmented Reality, Distributed Application Framework, DirectX , Stereo Vision, Telerobotics.

1. INTRODUCTION

With the wide recognition [1] of importance of virtual reality tools in telerobotics, teleoperation, telesurgery, and telepresence, more researchers turned their attention to 3D video generation and visualization techniques. In robotic surgery stereo image techniques allows the surgeon to estimate the relative depth of organs which greatly enhance the operation efficiency [2]. There is a variety of 3D-video formats, interlaced, page flipping, sync doubling, and line blanking. However, the design of portable client-server systems capable of reliably streaming stereo vision, haptic feedback, and computer assisted tlerobotics is still in an R&D stage.

Parallel camera configurations [3] accurately capture 3D depth in objects at low computational costs but with limited perception in near stereoscopic viewing. Tilted camera configurations [4] are more accurate in the horizontal than in the vertical direction compared parallel cameras. Computational aspects are more complicated and demanding compared to the parallel setup. The NuView 3D consists of two LCD-shutters, a prismatic beam splitter and an adjustable mirror. The shutters allow the camera lens to get only one of the left or right views at a time. There are basically two major classes of 3D visualization techniques. These are shuttering glasses and head mounted displays.

Network transmission delays[5], limited bandwidth, and reliability issues in streaming of stereo images is a fundamental problem in telerobotics. Image/video transmission based methods [6] consists of sending static images or live video from the slave robot location to the display(s) at the master arm location. One way to overcome the delay is to issue high-level operator commands that are interpreted by a local controller at the remote slave. The model-based methods [7] consists of using graphical tools to superimpose a graphics of the slave robot scene with a generated background image at the remote display. The operator adjusts the 3D graphic representation with respect to the object that appear in the scene and sends the final setting for execution, which saves many iterations. In an interactive model-based environment the master station uses a model of the robotic scene to regenerate visual and haptic feedback that help regenerating environment reaction on the operator for a given task. The Predictive models [8] use a prediction filter that receive delayed slave arm parameters, and generate motion based predicted actual parameters.

Augmented reality [9], consists of overlaying virtual graphics over real images of remote scene to provide the operator an augmented view of the remote scene combined with his own local actions. The operator can see how his action fits into the scene before executing his commands. The applications of this technique [10, 11] are: 1) as a tool for probing the real remote environment visible on video, 2) for enhancing video images through real object overlays, thus compensating for

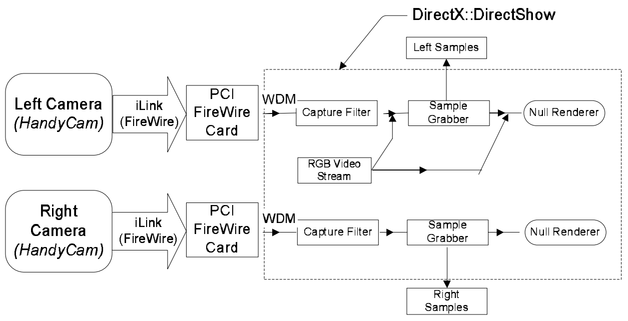

Figure 1. Sample grabber and camera interfacing

image degradation due to occlusion of objects, and 3) for introducing realistic looking but non-existent graphic objects so that they appear to be a part of the video scene.

We present a portable client-server framework for grabbing and relaying of a stereo video stream using DirectX and Windows sockets to feed a visualization system at a remote master station. The proposed framework uses Microsoft Visual C# and DirectX which provides COM based interfaces for various graphic related functionalities such as DirectShow. DirectShow provides efficient interfaces for the capturing and playback of video data. We also describe an effective implementation of AR technique in a client-server telerobotic system. The client station provides components that support augmented reality such as superimposing of animated graphics of slave arm with real views.

The organization of the rest of the paper is as follows. Section 2 presents our distributed stereo vision framework at server and client. In Section 3 we evaluate our approach. We conclude in Section 4.

2. STEREO VISION FOR TELEROBOTICS

The use of stereo image techniques allows the operator to estimate the relative depth among the remote objects. These techniques greatly enhance the operator's efficiency during tele-manipulation [12, 2]. Stereo vision requires large bandwidth for real-time streaming of video data in a client-server environment. In addition it also requires the use of advanced technologies like DirectX and Windows Sockets to accomplish the capturing and relaying of video.

We present a portable client-server framework for grabbing and relaying of a stereo video stream targeted to the use of head-mounted display (HMD) at the client station. We designed a client-server framework using Microsoft Visual C# and Microsoft DirectX. The server tasks are: (1) capture stereo images from two cameras at the slave side simultaneously, (2) establish a reliable client-server connection, (3) upon a request from the client, send a stereo frame comprising of two pictures to the client through windows sockets. We use SampleGrabber, a component of DirectX, to capture the video frame stream coming from a stereo camera as shown in Figure 1. A 400 Mbps IEEE 1394 serial bus (FireWire) interfaces the DirectShow capture filters to transfer the video stream from the digital cameras.

The client tasks are: (1) establish a highly optimized fast graphic display system to show the pictures received from the server, (2) detect and establish the connection with server, and (3) display the incoming pictures using a HMD. The graphical component of the is

We use the graphics device interface (GDI) of Windows graphical environment to (1) communicate between the application and the device driver, (2) perform the hardware-specific functions that generate output, and(3) display the received stereo picture (see Figure 2).

2.1. Implementation

Our setup uses synchronous windows sockets for the vision client-server interface. Two thread schemes were implemented in the server: (1) a single buffer with serialized transfer, and (2) double buffer, concurrent transfer (see Figure 1).

In the single buffer with serialized transfer, The SampleGrabber component of DirectShow uses a callback function to inform the completion of one video frame. Two thread instances of SampleGrabber running at the same time to capture the video coming from two cameras. The data is copied by SampleGrabber to some global memory buffer to be sent to the client through sockets. After hooking of callback function onto SampleGrabber, the FilterGraph (DirectShow) starts the video capturing. The last step of server socket is to transfer the video data. The server waits for a request of picture from client to start video data transfer.

On the client the GDI is set to draw the received pictures. Following GDI initialization, the server flushes the previous bitmap buffers, grab left and right images using callback functions, create a Bitmap information header for these images

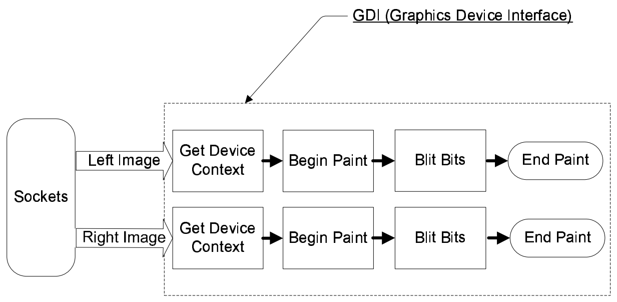

Figure 2. Stereo vision display on client side

and transfer to the client. The client gets the buffer size from the TCP stream, prepares the bitmap buffer, receives the bitmap information header, copies the bitmap data from the sockets into the buffer, requests for new picture, draws the stereo picture on the screen to be viewed in 3D.

In the double buffer, concurrent transfer, we try to optimize the transfer of video data by overlapping execution among capture and sending thread when using double buffers on the server.

Here we allocate two buffers, one for each stereo frame on the server. When a picture is received, the camera callback function is invoked which accesses a variable shared by multiple threads that indicates whether the buffer was copied to in the previous callback of this very camera. The camera status is used to synchronize the stereo frames for the left and right pictures. It is updated after copying the data to the buffer. If both cameras are ready, updating the buffer status enables the sending thread to transfer the content of this buffer over to the client. In case the second camera has not finished copying the picture to the buffer, buffer status is not updated.

The sending thread is responsible for receiving requests from the client. It checks the buffer status to determine which buffer should be sent. Next, it creates Bitmap headers and retrieves the buffer size. If the information has not already been sent to the client, it is sent. Otherwise, the server continues with the sending of buffer data only. The client proceeds in the same manner as with single buffer approach except that it does not receives the Bitmap information header and buffer size with each stereo frame. It retains the Bitmap information header and buffer size to properly display and read the required number of bytes from windows sockets.

2.2. Support to Augmented Reality at the Client

The image data retrieved from the *StereoSocketClient* component comes in a memory stream according to bitmap format. This stereo image is then displayed to the tele-operator using the *DXInterface* component, and HMD(Head Mounted Display) controller. Because the *DXInterface* heavily depends on *DirectX* API (Application Programming Interface), so a brief overview of it will be helpful in understanding the subject matter.

The *DirectX Surface* is used to superimpose different objects and switching from one surface to another using Page Flipping. In the following subsections, we present the components providing stereo vision and augmented reality functionality.

StereoSocketClient Component

StereoSocketClient component provides a mechanism to regenerate compatible bitmaps from the binary video data received from the stereo video server on the remote side (see Figure 3).

An instance of the StereoClientComponent connects to any remote computer running the vision server by specifying DNS address of remote computer & vision server port. Once the connection is established, it creates a separate thread for receiving the images. The new thread receives a fresh copy of both left and right images, informs the master (calling) thread and waits until current images are copied to the master thread to avoid the image data being overwritten during the copy operation.

IdentifyCamera Component

IdentifyCamera component can be used to find out the camera projection matrices for both left and right cameras based on a given set of four non-coplanar points constituting the basis vector. The component uses images available through the video stream provided by StereoSocketClient to update its GUI. On the closure of the GUI, both the left and right projection matrices (M_l) and M_r, respectively, are available to the calling thread based on user input.

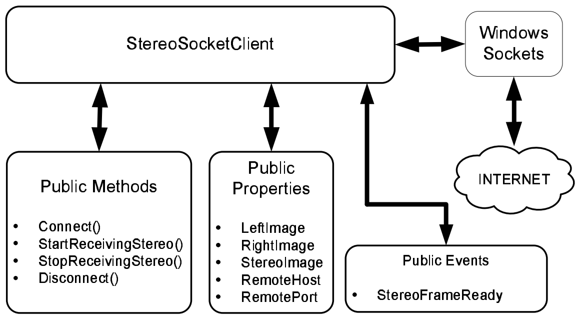

Figure 3. StereoSocketClient Component

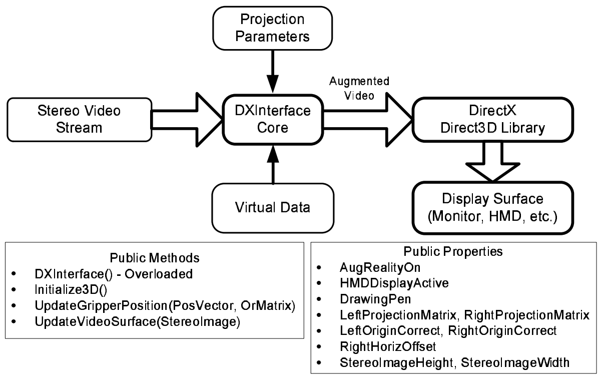
Projection Parameters
Augmented Video
Stereo Video Stream
DXInterface Core
DirectX Direct3D Library
Virtual Data
Display Surface (Monitor, HMD, etc.)
Public Methods
DXInterface() - Overloaded
Initialize3D()
UpdateGripperPosition(PosVector, OrMatrix)
UpdateVideoSurface(StereoImage)
Public Properties
AugRealityOn
HMDDisplayActive
DrawingPen
LeftProjectionMatrix, RightProjectionMatrix
LeftOriginCorrect, RightOriginCorrect
RightHorizOffset
StereoImageHeight, StereoImageWidth

Figure 4. An overview of DXInterface Component

RobotModel Component

This component plays an important role in the realization of the augmented reality system. *RobotModel* is needed to generate a graphical model of the slave arm that is driven by the user commands. It can be thought of as a thin copy of slave arm component with interfacing to visualization system.

DXInterface Component

DXInterface Component is the central part of augmented reality framework. All the video related tasks such as 1) augmentation of real video, 2) synchronization of real and virtual data, 3) projection on video surface, 4) page flipping for HMD stereo visualization, are handled by *DXInterface* (see Figure 4).

The component receives video stream in the form of stereo bitmap images from the *StereoSocketClient* component. The two other inputs of the *DXInterface* component are the projection matrices for the two cameras as well as the virtual data to be augmented with the real video stream. Based on the received inputs, *DXInterface* component uses the *DirectX* libraries to create new augmented image that will be displayed on HMD.

2.2.1. The Integrated AR System

AR functionality is supported through the following steps: (1) user inputs are read using *MasterArm*, (2) the incremental motion is feeds back to *RobotModel* and communicated to slave arm using *DecisionServer*, (3) *RobotModel* drives a graphical representation of slave arm using *DXInterface*, (4) *DXInterface* acquires a frame of remote scene from *StereoSocketClient* as well as left and right projection matrices from *IdentifyCamera* at the system initialization, (5) *DXInterface* generates the graphical output on each camera and sends the stereo image to HMD, (6) When the *IDecisionServer* receives the *OnMove* event from the client and simultaneously the slave position is sent to the *RobotModel* to update the slave arm graphical representation (see Figure 5).

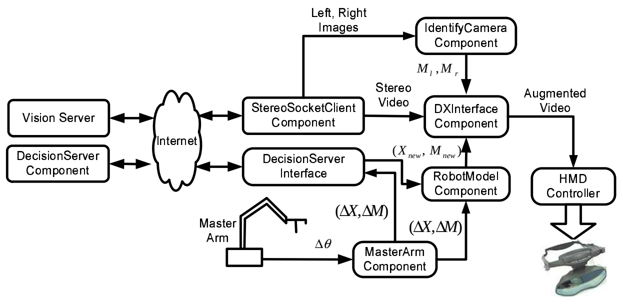

Figure 5. Block diagram of complete AR system on client side

3. EVALUATION

The server is 2.0 GHz P-IV which is connected to 100 Mbps LAN. Each force packet is 48 bytes. Each video picture is 288×360 pixels and each pixel is 3 bytes. Each stereo frame (two pictures) is 0.6 MB and requires a bandwidth of 5 Mbps/Frame.

The server throughput on network of streaming only force packets is about 1 KHz. Multistreaming of Force and video leads to a force packet rate of 250 Hz which drops to 76 Hz during active video intervals.

The time to copy one stereo frame from the SampleGrabber to the DRAM is 24.025 ms. The video copying time is increased to 60.48 ms if we enable a thread to only read force information without transfer during the above copying. If the network transfer of force packets is enabled the video copying drops to 33.46 ms because when force packets are transferred on the network the internal processor resources are exclusively used by the stereo copying thread.

Here the server enables the stereo copying thread as well as stereo transfer over the network. In the case of a single buffer with serialized transfer, the sending thread waits for the two SampleGrabbers to write stereo frame data to global buffer before transfer. The server throughput is 86.5 ms per stereo frame (11.6 fps). In the case of double buffer with concurrent transfer, the server throughput drops to 58.94 ms per stereo frame (17 fps).

Evaluation of relaying stereo vision for telerobotics is based on carruing out a set of telerobotic tasks. The operator view the remote scene using an HMD and convey his commands using a master arm with force feedback display. We successfully used the ability to scale-down the operator motion and camera zooming to scale-down operations into a very small space. This feature is the basis for robotic surgery. The tested tasks are: (1) pouring of water, (2) per-in-hole insertion with low tolerance, (3) assembly of a small water-pump, (4) operating drawers, and (5) carrying out wire-wrapping operations on an electronic circuit breadboard. The motion scalability and camera zooming allows to operate in a very small scale requiring high positioning accuracy like in the wire-wrapping tasks. The quality of stereo vision depends on setting a horizontal disparity of 6 cm and a fine vertical alignment of cameras. The operator's depth perception depends more on the image resolution than the refreshing rate. The AR component is useful in tasks involving positioning or path finding. AR does not help much in tasks involving intensive interaction with environment. Relaying Stereo Vision for Telerobotics is a useful tool for extending eye-hand motion coordination and human's manipulative capabilities to a remote work-station through the Internet. It contributes in allowing human to perform working tasks in hazardous, hostile, unaccessible, small, and extremely small environments.

4. ACKNOWLEDGEMENT

The authors acknowledge computing support and conference attendance support from King Fahd University of Petroleum and Minerals (KFUPM).

5. CONCLUSION

A reliable distributed framework based on DirectX, Visual C #, and Window sockets is proposed for streaming stereo vision in a client-server tlerobotic system. A modular and portable framework was implemented using a multi-threaded execution at both client and server. We engineered the thread granularity, synchronization, concurrency, and priority to promote frame rate for HMD visualization technology. Thread engineering proved to be effective in achieving a sampling rate of 17 Hz for stereo video opposed to 11 Hz without optimization. We also implemented client components to support the design of augmented reality functions like superimposing graphics with real stereo images, camera model identification, and slave robot components.

6. REFERENCES

[1] R. D. Howe; Y. Y. Matsuoka. Robotics for surgery. *Annual Review of Biomedical Engineering*, pages 211–240, 1999.

[2] X. Ning; T. J.Tarn. Action synchronization and control of internet based telerobotic systems. *IEEE Inter. Conf. on Robotics and Automation*, 1:219–224, 1999.

[3] S. Lee; S. Lakshmanan; S. Ro; J. Park; C. Lee. Optimal 3d viewing with adaptive stereo displays for advanced telemanipulation. *International Conference on Intelligent Robots and Systems*, pages 1007–1014, 1996.

[4] S. Lee; S. Ro; J. Park; C. Lee. Optimal 3d viewing with adaptive stereo displays: A case of tilted camera configuration. *ICAR '97*, pages 839–844, 1997.

[5] A. Bejczy. Space shuttle remote manipulator system force-torque system. *http://ranier.oact.hq.nasa.gov/telerobotics_page*.

[6] T. Suzuki; T. Fujii; K. Yokota. Teleoperation of multiple robots through the internet. *IEEE International Workshop on Robot and Human Communication*, pages 84–89, 1996.

[7] H. Friz; P. Elzer; B. Dalton; K. Taylor. Augmented reality in internet telerobotics using multiple monoscopic views. *IEEE Computer*, pages 354–359, 1998.

[8] M. Jägersand. Image based predictive display for tele-manipulation. pages 550–556, 1999.

[9] R. Marin; P.J. Sanz; J.S. Sanchez. A very high level interface to teleoperate a robot vis web including augmented reality. *Inter. Conf. on Robotics and Automation*, pages 2725–2730, 2002.

[10] P. Milgram; A. Rastogi; J. Grodski. Telerobotic control using augmented reality. *IEEE Inter. Workshop on Robot and Human Communication*, pages 21–29, 1995.

[11] P. Cohen J. Gu; P. Augirre. Augmented reality interface for telerobotic application. *6th IEEE Workshop on Apps. of Computer Vision*, 2002.

[12] F. Paolucci; M. Andrenucci. Teleoperation using computer networks: Prototype realization and performance analysis. *Electrotechnical Conference, MELECON '96., 8th Mediterranean*, 2:1156–1159, 1996.