VECIMS 2008 – IEEE International Conference on
Virtual Environments, Human-Computer Interfaces, and Measurement Systems
Istambul, Turkey, 14-16 July 2008

Development of a Stereo Vision System for Remotely Operated Robots: A Control and Video Streaming Architecture

A. Amanatiadis[1], A. Gasteratos[2], C. Georgoulas[1], L. Kotoulas[1], I. Andreadis[1]

[1]Laboratory of Electronics, Department of Electrical and Computer Engineering,
Democritus University of Thrace, 12 Vas. Sofias Str, GR-67100, Xanthi, Greece
E–mail: {aamanat, cgeorg, lkotoula, iandread} @ee.duth.gr

[2]Laboratory of Robotics and Automation, Department of Production and Management Engineering,
Democritus University of Thrace, University Campus Kimmeria, GR-67100, Xanthi, Greece
E–mail: agaster@pme.duth.gr

Abstract – *This paper describes the open and flexible architecture of a stereo vision system prototype, using open source software. The system is designed for teleoperated robots and includes a four degrees of freedom stereo head mechanism, a pair of high performance digital cameras, a head tracker and a head mounted display. All processes for the head control and video streaming are performed in Linux-based Real-time Operating Systems using open source libraries under GPL license. Experimental results showed that our system is capable of satisfying the hard-real time requirements for the head control, with great precision, and a low latency for the stereo video streaming. The video streaming management is particularly sophisticated resulting in a flexible, efficient and reliable service.*

Keywords – *Telerobotic Vision , Human-Machine Interaction, Real-Time Remote Control, Open-Source Programming.*

I. INTRODUCTION

Remotely operated robots, functioning in hazardous and time critical environments, have significant requirements for control and visual information [1], [2]. The control systems are supposed to guarantee a precise timely response in order to prevent fatal scenarios in bomb disposal operations or in life rescue missions. Significant role to these operating scenarios play the concurrent visual information provided to the remote operators by the on-board mounted cameras.

Visual information [3], [4] is often displayed in one or more monitors depending on the number of on-board mounted cameras. In sophisticated and multi-tasking robots more than one operators are performing certain actions. Especially, in case of robots with grippers and robotic arms, one operator might be dedicated only with the maneuvering and controlling of the robotic arm or gripper. In these operation scenarios, the dedicated user must be focused only on this task and furthermore should have the best visual understanding of the working field. The widely used equipment and gear for these assignments is a Head Mounted Display (HMD) and an attached head tracker.

The HMD projects visual feedback of the remote robot in front of operator eyes. A single camera feedback projection in both eyes is not so significant since the result in operator's perception is the same as being watched from a single monitor. Thus, a pair of cameras are used instead, in order to provide a stereo feedback to the operator's HMD, enhancing his visual perception and improving the sense of distance [5]. Consequently, operators can judge situations and perform actions more efficiently based on the qualitative information of the synchronized stereo video streams. The use of a head tracker expands the operator's functions while it offers a hands-free ability to remotely control the pose of the robot head. The inertial measurement devices used for the head tracking usually contain rate gyroscopes (gyros) and accelerometers. The measurements of the inertial sensor can be processed and transmitted as control signals to the remote robot.

Many different interfaces have been proposed in literature recently. A method of robot teleoperation that allows a human operator to control a robot manipulator in presented in [6]. It uses a non-contacting vision-based humanrobot interface for both the communication of the human motion to the robot and for feedback of the robot motion to the human operator. However, this visual feedback does not give the operator the depth visual information that is necessary for this critical task. In [7], a sophisticated anthropomorphic robot is developed for space operations. It is comprised of a stereo head which transmits the video feedback to the operator through a HMD. The same human-machine interface is developed also in [8] for robot control. Both implementations however, require sophisticated and expensive equipment and are built with proprietary software. In [9], an on-line robot architecture is proposed. It enables the control of a robot by interacting with an advanced user interface with very promising results but the real-time constraint for control can not be guaranteed. In this paper, we propose a human-machine interface which guarantees the real-time control of a binocular robotic head. Furthermore, stereo video streaming transmission with low latency is implemented. This hands-free interface is implemented exclusively with open source software on a Linux-based Real-time Operating System. The proposed control and video architec-

978-1-4244-1928-9/08/$25.00 ©2008 IEEE

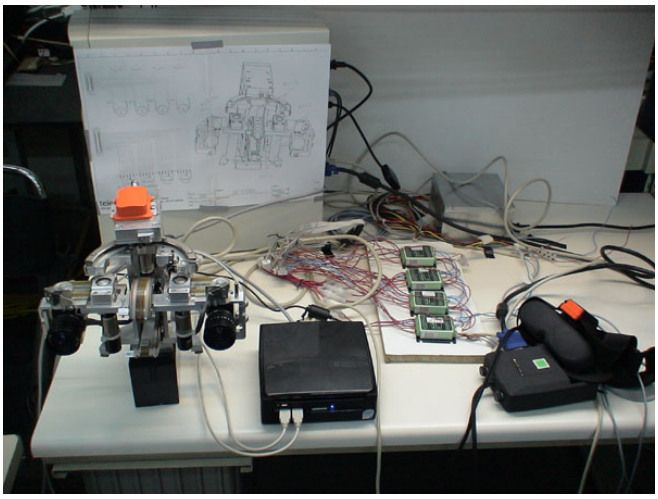


Fig. 1. The Stereo Vision System.

ture is realized with great consideration for flexibility and expandability in future additions and features.

II. SYSTEM DESIGN

The functions of the proposed system, as shown in Fig. 1, can be separated into video streaming and motion control and are both implemented with the use of two host computers. The first host computer is called Mobile Mechatronic Unit (MMU), and is placed on the mobile platform where the stereo vision head is operating. The second host computer comprises the Mobile Control Unit (MCU) which is placed on the remote control center. There, the remote operator wears the HMD with the attached head tracker.

The video streaming presents high computational burden and resource demand while it requires the full usage of certain instruction sets of a modern microprocessor. In contrast, motion control includes filtering and signal processing from the head tracker output and requires the operation system to be able to execute real-time tasks. This demand for high multimedia performance and real-time motion control has forced us to adopt a computer structure consisting of two high performance computers with RT-Linux operating system for the MMU and MCU host computers. However, the two main tasks of the video streaming and the motion control will be performed in different kernel layers due to their different requirements.

The two host computers are presently connected together with a high speed Ethernet network. Future development will replace the Ethernet network with a secure wireless connection for achieving better autonomous operation of the robot. The communication protocol between the computers uses a higher level abstraction, built on top of sockets, meeting the requirements for low latency and priority handling.

Fig. 2 shows the flowchart of the stereo vision software system. The right part shows the control flow from the head tracker starting from the MCU and ending to the motors of the

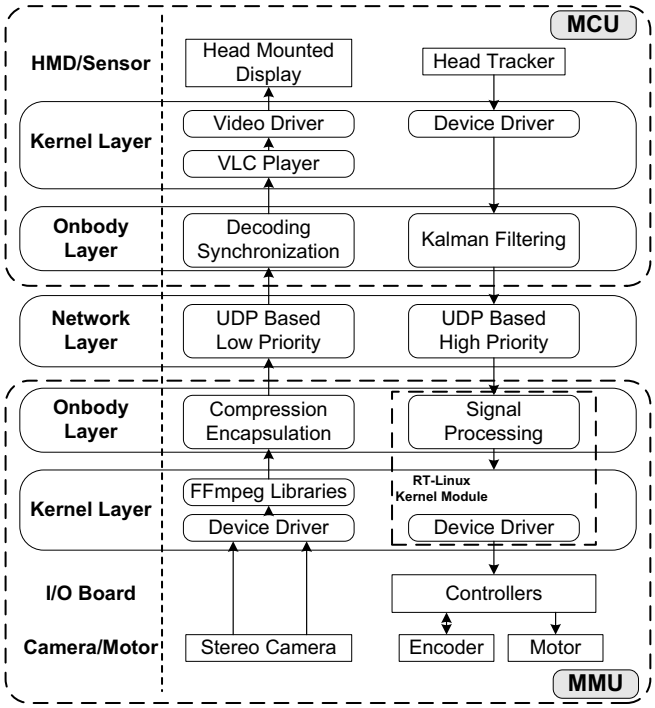


Fig. 2. Flowchart of Stereo Vision Software System.

stereo head. The left flow shows the video stream from the stereo head cameras of the MMU to the player in the MCU.

III. CONTROL SYSTEM

A. Hardware Architecture

A head tracker inertial measurement unit [10] was used to obtain high update rate measurements. Its internal low-power signal processor provides drift-free 3D orientation as well as kinematic data of 3D acceleration and 3D rate of turn (rate gyro). The data used for the head tracking is the pitch and yaw in order to send the pan and tilt commands to the teleoperated stereo head, respectively. The chosen interface used for connecting the sensor to the MCU computer is the RS-232, in order to have full access to the basic level of the sensor unit and a full compatibility for the drivers, since no serial-to-USB converter drivers are needed. The second sensor [10] attached on the stereo head mechanism, as shown in Fig. 1, is used for the stabilization of the stereo head [11] which is not in the scope of this paper. However, special attention was paid for the placement of this inertial sensor. Possible errors and distortions from the strong currents of the servo motors can be quite large enough in order to deteriorate the inertial measurements. A global reset is performed each time the HMD sensor is initialized to orientate the tracker in such a way that the sensor axes point in exactly the same direction as the axes of the operator's global coordinate frame. The sample frequency used is 100 Hz with a baudrate of 115 Kbps.

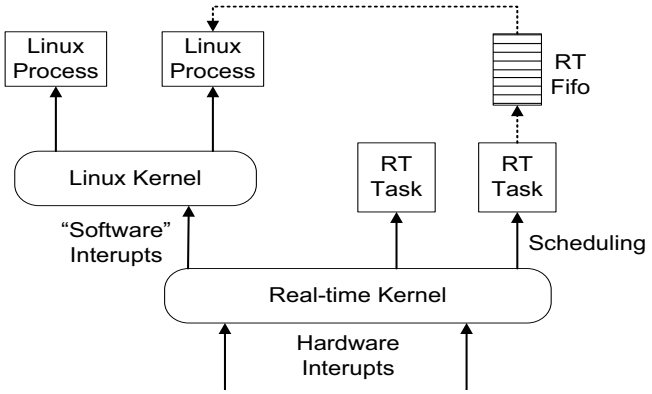


Fig. 3. The chosen operating system combines the use of two kernels, RT-Linux and Linux to provide support for critical tasks and soft real-time applications, respectively.

Two harmonic drive actuators are used to move the pan and tilt axis of the stereo head, based on feedback from incremental position encoders. The chosen high precision encoders guarantee a specification 0.01 degree resolution and a maximum frequency response of 100 KHz. The DC servo motors have a maximum output speed of 110 rpm and maximum radial load 59 N, which is adequate for the two cameras load.

Each servo is connected to a controller [12] which sends low-level commands to the actuators for executing the trajectories received by the head tracker. A very precise calibration of the controllers was performed so that we could utilize the great degree precision of the position encoders. Position control strategy was chosen while position is the most important aspect of a high performance head tracking control. The Proportional Integral Derivative (PID) controller values were calibrated in discrete-time through the use of real-time processes running with fixed time steps. The use of a simple, and easy to tune control strategy across the pan and tilt axis helped to ensure the reliability and robustness of the whole system. The following equation represents the general PID controller [13].

$$u = K_p e + K_i \int e dt + K_d \frac{d(-PV)}{dt} \qquad (1)$$

B. Software Architecture

Key feature for the implementation of the real-time control was the operating system we used. Since critical applications such as control need low response times, RT-Linux operating system was chosen for both MMU and MCU host computers. The distribution used is called OCERA [14] and is an Open Source project which provides an integrated execution environment for embedded real-time applications. It is based on components and incorporates the latest techniques for building embedded systems. The architecture is designed to develop hybrid systems with hard and soft real-time activities as shown in Fig. 3. The Linux kernel is treated as the lowest priority task under the RT kernel. In this case, we allocated the critical task of control at the RT-Linux level and the less critical tasks, such as inertial data filtering, at the Linux level. The real-time tasks need to communicate with user-space processes for things like file access, network communication or user interface. Thus, it provides FIFOs and shared memory implementations that provide communication with this user-space processes. The interface for both kinds of activities is a POSIX based interface.

In the MCU, the data received by the head tracker is first filtered by a Kalman filter. The software strategy dilemma of whether to use polling or events was considered in our implementation. Apart from the fact that the choice is mostly dependent on the user programming environment several other considerations were examined. When using the polling method, the user continuously or at a certain interval, queries the head tracker if new orientation data has been calculated. When queried, the sensor will immediately return the most recently calculated data. The polling method is useful when the query function runs in a loop at a certain update rate and each time orientation data is needed, the user just needs the latest data and not necessarily every single sample. When using the events method, instead of continuously querying the sensor, the event notifies the user when new data has been calculated and is available for retrieval with the appropriate functions. We chose the polling method since it ensures that we always get the latest available orientation data when we ask for it. The polling method allows that the other processes in our software to be asynchronous with the sampling rate of the head tracker itself, and we can synchronize the data with our processes. Furthermore, polling is slightly more straightforward to implement.

The errors in the force measurements introduced by our accelerometer and the errors in the measurement of angular change in the orientation with respect to the inertial space introduced by gyroscopes, were the two fundamental error sources which affected the error behavior of the operators head trajectory. Furthermore, all inertial measurements were corrupted by additive noise [15]. The Kalman filter [16], [17] was used while is a form of optimal estimator, characterized by recursive evaluation using an estimated internal model of the dynamics of the system. The filtering was implemented on the MCU computer where the inertial sensor is attached, using the soft-real time kernel. The control data received from the MMU, should be translated into motor commands for the equivalent axis. This operation was considered time critical while fast and accurate position commands to a remote robot can guarantee its safe operation. This strategy of considering the head tracker commands time critical and their implementation in the hard real-time kernel, gives a flexibility to the overall system in a way that additional future motor commands and even more crucial, like the operation of a gripper, can be implemented easily while satisfying the hard real-time constraints.

The concurrency and parallelism was considered in the programming of the robotic system by using a multi-thread model. The motor run time models are not using the $wait.until.done()$ function, while a change in the operator's field of view indicates that the previous movement should not

be completed but a new motion position command should be addressed. The following runtime model was chosen for the motor class:

```
Thread 1 (control)
motor.change_position()
do other things

Thread 2 (monitor)
periodically wake up
read.new_sensor_position()
if (new_sensor_position !=...
...old_sensor_position)
old_sensor_position=motor.change_...
...position.(new_sensor_position)
```

Simultaneous and non-synchronized accesses to the same resources, such as servo motors, was not a set of problems for our implementation while the the pitch and yaw commands would move separately the tilt and pan axis, respectively. However, in case of a future additional operation, such as motor stabilization, the sharing of the same resources would be a great problem. Thus, the software programming infrastructure considered the shared resources and critical sections in order to guarantee the expandability and flexibility of the stereo vision system. The critical sections were easily implemented since the protected operations were limited. However, special attention was paid since critical sections can disable system interrupts and can impact the responsiveness of the operating system.

IV. VIDEO STREAMING SYSTEM

A. Hardware Architecture

Each of the stereo head cameras on MMU is capable of outputting progressively images of 640×480 pixel resolution at maximum 30 frames per second. The digital cameras transmit the images over the fast USB 2.0 interface directly to the host's memory without the usage of frame grabbers. In order to determine the internal camera geometric and optical characteristics, camera calibration was necessary. A variety of methods have been reported in the bibliography. The method we used is described in [18] using its available C Open Source code. The method is a non self-calibrating thus, we used a projected chessboard pattern to estimate the camera intrinsics and plane poses. Finally, the calibration results were used to rectify the images taken from cameras in order to have the best results in possible subsequent image processing algorithms. The video processing requires high computational burden and resources while it makes a full usage of certain instruction sets of a modern microprocessor. Thus, a high performance processor was chosen for the MMU computer.

The operator in the MCU will receive the stereo pair of images in the stereo HMD. The chosen HMD has the same input resolution like cameras 640×480 and a refresh rate of $70Hz$. Two separate 15 pin D-Sub (VGA) interfaces are used for the stereo image input to the HMD. Thus, the MCU computer is equipped with a double output high performance graphic card in order to display in different outputs each video stream.

B. Software Architecture

Vision systems of mobile robots must unify the requirements and demands of both computer vision and image processing disciplines and robotic and embedded system disciplines. While the state of the art in computer vision algorithms is advanced, many computer vision processes are computationally expensive. Therefore, the resource demands of computer vision applications are in conflict with the requirements posed by robotics and embedded systems. For our case, a compression scheme must be implemented in order to transmit the stereo image stream. The high input data rate from the cameras of $2(stereo) \times 640 \times 480(resolution) \times 3(color) \times 8(bit\,per\,pixel) \times 25(fps) \cong 351$ Mbps demands a compression algorithm with high compression ratio, low computational complexity and good output quality. Furthermore, the compressed video should be packetized and streamed over the communication network.

The architecture chosen aims to make the MMU computer a video server which will perform the following primary tasks: i) capture video from both cameras, ii) compress video using a codec, iii) packetize the compressed video and attach time stamps within the packets and iv) stream the packets over the communication network. For all the previous tasks we selected the FFmpeg video open source libraries [19]. The video server allows multicast transmission while it sends each video stream to a fixed-destination multicast address. The services dealing with each stream, like the video player in the MCU, only have to listen to the appropriate multicast address, so several services can receive the same video stream without increasing bandwidth consumption. The compression was done using MPEG-4 codec, and the transmission of the video streams using the MPEG Transport Stream [20]. MPEG-TS provides many features found in data link layers, such as packet identification, synchronization, timing (clock references and timestamps), multiplexing and sequencing information. In the architecture chosen, each processing tree is executed within its own thread and is processed in parallel with other source nodes, like the control loop. This framework ensures appropriate synchronization between the image streams. With this framework, the developers do not need to worry about locking issues and synchronization primitives. The UDP communication protocol was used between the two computers while it uses a higher level abstraction, it is built on top of sockets, and meets the requirements for low latency.

In the MCU computer, the VLC player [21] was chosen for the playback service of the video streams. VLC is an open source cross-platform media player which supports a large number of multimedia formats and it is based on the FFmpeg libraries. The same FFmpeg libraries are now decoding and synchronize the received UDP packets. Two different instances of the player are functioning in different network ports.

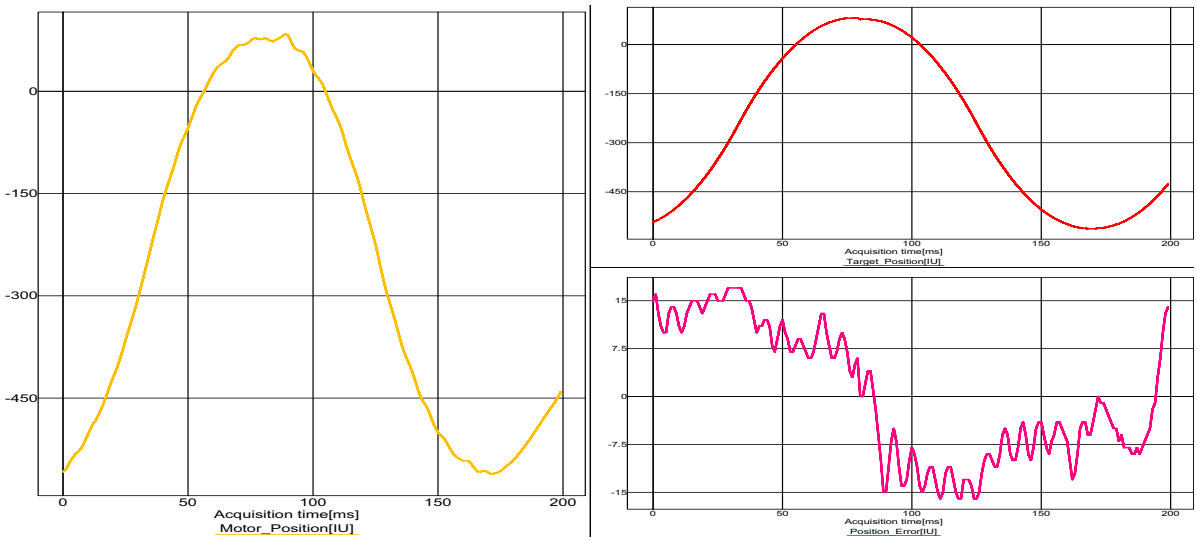


Fig. 4. A plot of position controller performance. Left: The motor position, Up Right: The target motor position, Down Right: The position error.

Each stream from the video server is transmitted to the same network address, the MCU network address, but in different ports. Thus, each player receives the right stream and with the help of the MCU on board graphic card capabilities, each stream is directed to one of the two available VGA inputs of the HMD.

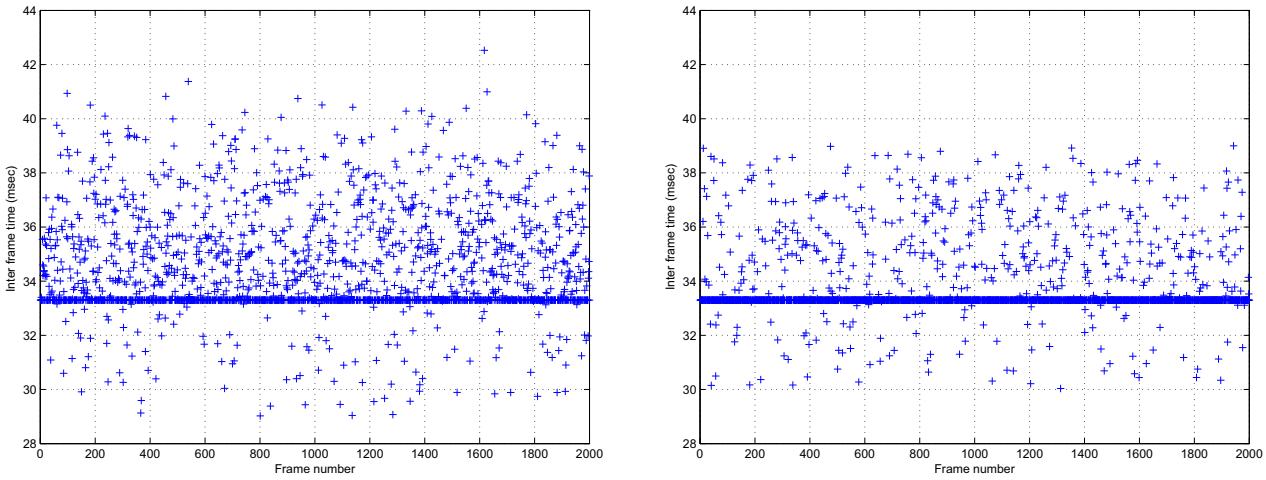
The above chosen architecture offers a great flexibility and expandability in many different aspects. In the MMU, additional video camera devices can be easily added and be attached to the video server. Image processing algorithms and effects can be implemented using the open source video libraries like filtering, scaling and overlaying. Furthermore, in the MCU, additional video clients can be added easily and controlled separately.

V. EXPERIMENTAL RESULTS

During calibration of the PIDs the chosen values of (1) were $K_p = 58.6$, $K_i = 2000$ and $K_d = 340.2$. The aim of the controlling architecture was to guarantee the fine response and accurate axis movement. Fig. 4 shows the response of the position controller in internal units (IU). One degree equals to 640 IU of the encoder. As we can see, the position controller has a very good response and follows the target position. From the position error plot we can determine that the maximum error is 17 IU which equals to 0.026 degrees.

To confirm the validity of the proposed architecture scheme, of selecting RT-Linux kernel operating system for the control commands, interrupt latency was measured on a PC which has an Athlon 1.2GHz processor. To show the effects of the operating system latency, we ran an I/O stress test as a competing background load while running the control commands. With this background running, a thread got the CPU clock-count and issued a control command, which caused the interrupt. Triggered by the interrupt, an interrupt handler (another thread) got the CPU clock-count again and cleared the interrupt. Iterating the above steps, the latency, the difference of the two clock-count values, was measured. On standard Linux kernel, the maximum latency was more than 400 msec, with a large variance in the measures. In the proposed implementation in RT-Linux kernel the latency was significantly lower with maximum latency less than 30 msec and very low variation.

The third set of results show the inter-frame times, the difference between the display times of a video frame and the previous frame. The expected inter-frame time is the process period $1/f$ where f is the video frame rate. In our experiments, we used the VLC player for the playback in the MMU host computer. We chose to make the measurements on the MMU and not on the MCU computer in order to calculate only the operating system latency avoiding overheads from communication protocol latencies and priorities. The selected video frame rate was 30 frames per second. Thus, the expected inter-frame time was 33.3 msec. Fig. 5(a) shows the inter-frame times obtained using only the standard Linux kernel for both control and video process. The measurements were taken with heavy control commands running in the background. The inter-frame time due to the control process load introduces additional variation in the inter-frame times and increases these times to more than 40ms. In contrast, Fig. 5(b) shows the inter-frame times obtained using the RT-Linux kernel with high resolution timers for the control process and the standard Linux kernel for the video process. The measurements were taken with the same heavy control commands running in the background. As we can see, the inter-frame times are clustered more around the correct value of 33.3 msec and their variation is lower.



(a) (b)

Fig. 5. Inter-frame time measurements: (a) Both control and video process running in standard Linux kernel; (b) Control process running in RT-Linux kernel and video process in standard Linux kernel.

VI. CONCLUSION

This paper described a robust prototype stereo vision paradigm for real-time applications, based on open source libraries. The system was designed and implemented to serve as a binocular head for a remotely operated robot. The two main implemented processes were the remote control of the head via a head tracker and the stereo video streaming to the mobile control unit. The key features of the design of the proposed stereo vision system include: 1) a complete implementation with the use of open source libraries based on two RT-Linux operating systems, 2) a hard real-time implementation for the control commands, 3) a low latency implementation for the video streaming transmission and 4) a flexible and easily expandable control and video streaming architecture for future improvements and additions. All the aforementioned features make the proposed implementation appropriate for sophisticated remotely operated robots.

ACKNOWLEDGMENT

This work was supported by the E.C. under the FP6 research project for improvement of the emergency risk management through secure mobile mechatronic support for bomb disposal, "RESCUER", IST-2003-511492.

REFERENCES

[1] R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Trans. Syst., Man, Cybern., Part C,*, vol. 34, no. 2, pp. 138–153, 2004.

[2] A. Davids, "Urban search and rescue robots: from tragedy to technology," *IEEE Intell. Syst.*, vol. 17, no. 2, pp. 81–83, 2002.

[3] G. Desouza and A. Kak, "Vision for mobile robot navigation: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 237–267, 2002.

[4] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Autonomous Robots*, vol. 11, no. 1, pp. 9–18, 2001.

[5] P. Willemsen, M. Colton, S. Creem-Regehr, and W. Thompson, "The effects of head-mounted display mechanics on distance judgments in virtual environments," in *Proc. of the 1st Symposium on Applied perception in graphics and visualization*, 2004, pp. 35–38.

[6] J. Kofman, X. Wu, T. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Trans. Ind. Electron.*, vol. 52, no. 5, pp. 1206–1219, 2005.

[7] W. Bluethmann, R. Ambrose, M. Diftler, S. Askew, E. Huber, M. Goza, F. Rehnmark, C. Lovchik, and D. Magruder, "Robonaut: A robot designed to work with humans in space," *Autonomous Robots*, vol. 14, no. 2, pp. 179–197, 2003.

[8] S. Tachi, K. Komoriya, K. Sawada, T. Nishiyama, T. Itoko, M. Kobayashi, and K. Inoue, "Telexistence cockpit for humanoid robot control," *Advanced Robotics*, vol. 17, no. 3, pp. 199–217, 2003.

[9] R. Marin, P. Sanz, P. Nebot, and R. Wirz, "A multimodal interface to control a robot arm via the web: a case study on remote programming," *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1506–1520, 2005.

[10] Xsens Motion Technologies home page, *http://www.xens.com/*.

[11] A. Amanatiadis, I. Andreadis, A. Gasteratos, and N. Kyriakoulis, "A rotational and translational image stabilization system for remotely operated robots," in *Proc. of the IEEE Int. Workshop on Imaging Systems and Techniques*, 2007, pp. 1–5.

[12] Technosoft Servo Drives home page, *http://www.technosoftmotion.com/*.

[13] K. Astrom and T. Hagglund, *PID controllers: Theory, Design and Tuning*. Instrument Society of America, Research Triangle Park, 1995.

[14] OCERA project home page, *http//www.ocera.org*.

[15] S. Ovaska and S. Valiviita, "Angular acceleration measurement: A review," *IEEE Trans. Instrum. Meas.*, vol. 47, no. 5, pp. 1211–1217, 1998.

[16] G. Welch and G. Bishop, "An introduction to the Kalman filter," *ACM SIGGRAPH 2001 Course Notes*, 2001.

[17] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1998.

[18] J. Bouget, "Camera calibration toolbox for Matlab," *California Institute of Technology, http//www.vision.caltech.edu*, 2001.

[19] FFmpeg project home page, *http//ffmpeg.sourceforge.net*.

[20] S. Gringeri, B. Khasnabish, A. Lewis, K. Shuaib, R. Egorov, and B. Basch, "Transmission of MPEG-2 video streams over ATM," *IEEE Multimedia*, vol. 5, no. 1, pp. 58–71, 1998.

[21] VideoLAN project home page, *http://www.videolan.org/*.