An Architecture for the Control of a Mechatronic Head used in Rescue Robotics Applications

Angelos Amanatiadis, Antonios Gasteratos and Spyridon Mouroutsos
Democritus University of Thrace
Vas. Sophias 12, GR-671 00 Xanthi, Greece
agaster@pme.duth.gr

Abstract

In this paper we present a control architecture for a semiautonomous tele-operated robotic head, suitable for emergency situations. The proposed architecture controls in parallel the 4 dof of the head in a twofold fashion including automatic mechanical image stabilization and remote head posing. The first process is realized via inertial feedback to the pan and tilt motors. The second process duplicates the movement of the operator's head to control its robotic counterpart. Both the operations require the concurrent use of pan and tilt servo motors and, thus, all the processes are performed with Linux-based Real-time Operating Systems and open source libraries under GPL license. Detailed behavior is executed through priorities to prevent the computation waste and simultaneous and non-synchronized accesses to servo motors. Experimental results showed that the architecture is capable of satisfying the hard-real time requirements for the servo control, with great precision.

Keywords

Human-Machine Interaction, Real-Time Control, Rescue Robotics

1. Introduction

Remotely operated robots possess significant requirements for control and visual information, since they perform in hazardous and time critical environments [1, 2]. Their control systems should guarantee a precise timely response in order to prevent fatal scenarios in bomb disposal operations and in life rescue missions [3]. Significant role to these operating scenarios play the concurrent visual information provided to the remote operators by the on-board cameras. Image stabilization is an important qualitative feature for such system, since it removes the unwanted motion from the frame sequences captured from the cameras. This image sequence improvement is necessary in order to enhance the perception of the scene as sensed by the human operator. Several stabilization system implementations that use visual and inertial information have been reported. An image stabilization system which compensates the walking oscillations of a biped robot is described in [4]. A vision and inertial cooperation for stabilization have been also presented in [5] using a fusion model for the vertical

reference provided by the inertial sensor and vanishing points from images. A visuo-inertial stabilization for space variant binocular systems has been also developed in [6], where an inertial device measures angular velocities and linear accelerations, while image geometry facilitates the computation of first-order motion parameters. In [7], course stabilization and collision avoidance is achieved using a bio-inspired model of optic flow and inertial information applied to autonomous flying robots.

The stabilized image sequence is projected as visual feedback of the remote robot in front of operator eyes by means of a head mounted display (HMD). 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 stereo pair is used, in order to provide a stereo feedback to the operator's HMD, enhancing his visual perception and improving the sense of distance [8, 9]. Consequently, the operator 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. In this paper, we present a control architecture for parallel use of servo motors for two separate robot functions. The first process includes the automatic camera stabilization through pan and tilt motors. The second process includes the movement of the stereo vision head from head inertial sensor feedback. Both operations require the concurrent use of pan and tilt servo motors. In order to fulfill the requirements in such complex situation, the choice of hardware mechanism, software architecture and real-time operation were very crucial. The system is designed to fit into mobile rover platforms allowing the architecture to be modular and the whole system expandable. Special attention was paid to the real-time constraints, particularly for the control part of the system. Key feature is the real-time servo control system, written in C, using Open Source Software which includes a Linux-based Real-Time Operating

2. Proposed architecture

A. Hardware Architecture

Fig. 1 depicts the components of the proposed stabilization and manipulation system and how they interact. A head tracker inertial measurement unit was used to obtain high update rate measurements. The unit possesses an internal signal processor to provide accurate, drift-free signals with highly dynamic response and long-term stability. The data used for the head tracking are the pitch and yaw angle measurements, that commands the pan and tilt d.o.f. of the tele-operated stereo head, respectively. The chosen interface used for connecting the sensor to the Mobile Control Unit (MCU) on-board 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 attached on the stereo head mechanism is used for the stabilization of the stereo head [10]. The optimized filter outputs of pan and tilt are the subsequent feedback to the controllers for opposite movement of the pan and tilt axis, respectively, realizing the stabilization process. 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 sensor mounted on the HMD is initialized to orientate the tracker in such a way that the sensor axes point exactly towards the same direction as the axes of the operator's global coordinate frame. The sample frequency used is 100 Hz at a baud-rate of 115 Kbps. 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.

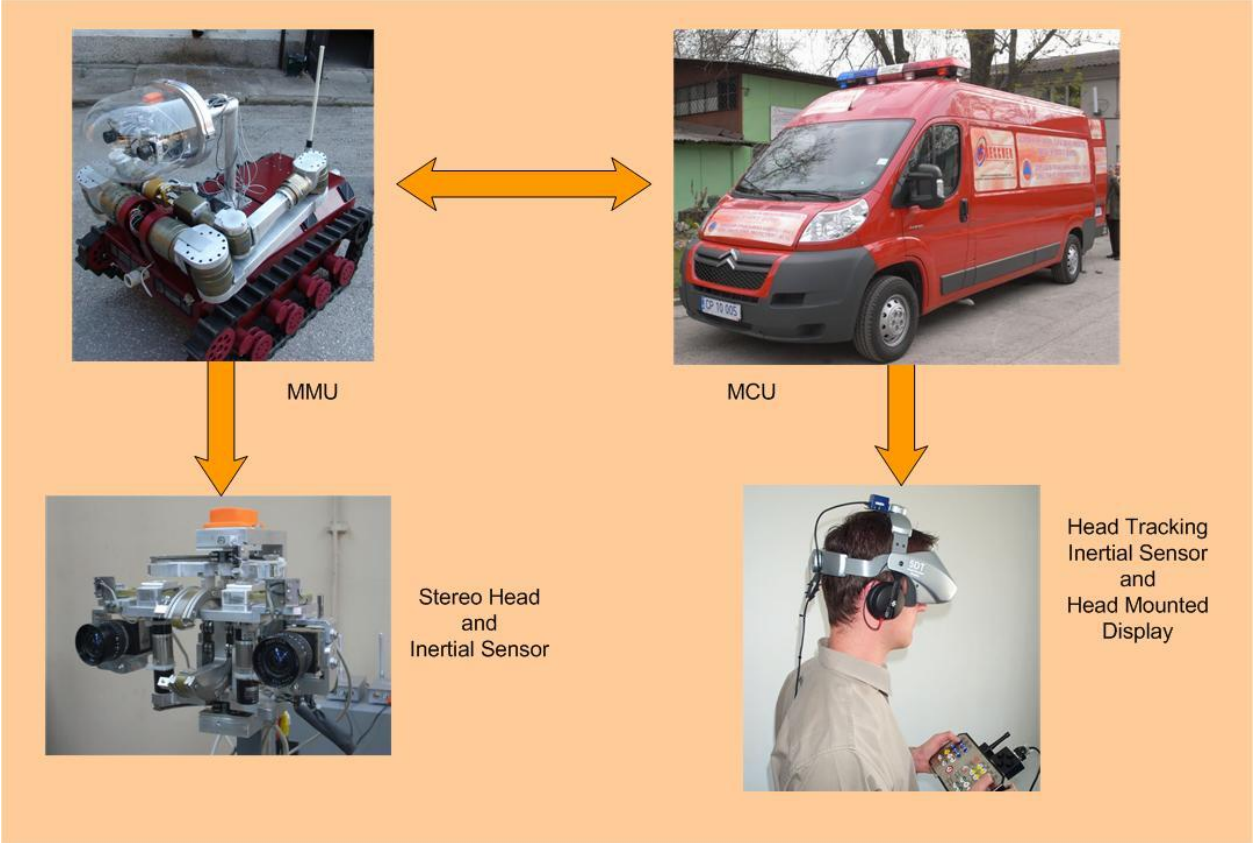

Figure 1: The Stereo Vision System Components.

Each servo is connected to a controller which transmits the 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.

B. Software Architecture

Key feature for the implementation of the real-time control was the operating system used. Since critical applications need low response times, RT-Linux operating system was chosen for both Mobile Mechatronic Unit (MMU) and MCU host computers. The distribution used is called OCERA [11] 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 presented in Fig. 2. 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 several tasks such as file access, network communication and 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.

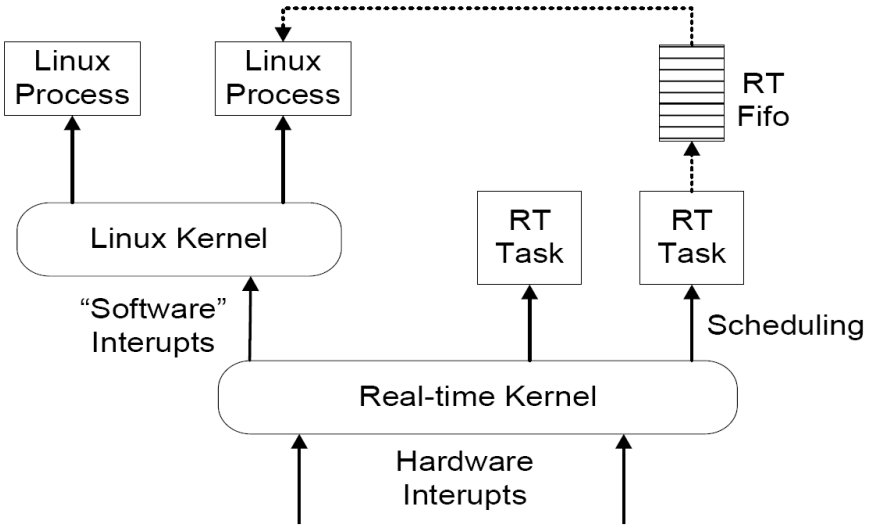

Figure 2. The chosen operating system combines the use of two kernels, RT-Linux and Linux to provide support for critical tasks and soft real-time applications, respectively.

In the MCU, the data received by the head tracker are 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 it is requested. 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 [12]. The Kalman filter [13, 14] was used as it 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 mechatronic 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, is a critical aspect since both stabilization and head tracking movement are performed at the same time. Thus, a priority scheduling feedback loop [15] was implemented. The priority feedback scheduler is implemented as an additional real-time periodic task, as shown in Fig. 2. The inputs are the measured response times of the control tasks and the inputs from both sensors. Priority was given to head posing tracker since we are interested firstly in giving the operator the desired view and then an optimized view by mechanical stabilization. The RT-Linux kernel keeps track of the real time tasks execution cycles, thus it allows to recover a precise measure of the control tasks execution times from the scheduling regulator. As the feedback scheduler is a simple feedback algorithm running at a slow rate, its computing cost is quite low. 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.

3. Simulation results

During tuning of the controllers the chosen PID gains were Kp = 58.6, Ki = 2000 and Kd = 340.2, respectively. The aim of the controlling architecture was to guarantee the fine response and accurate axis movement. Fig. 3 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. The latency measured was 12 msec. In order 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 demonstrate 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.

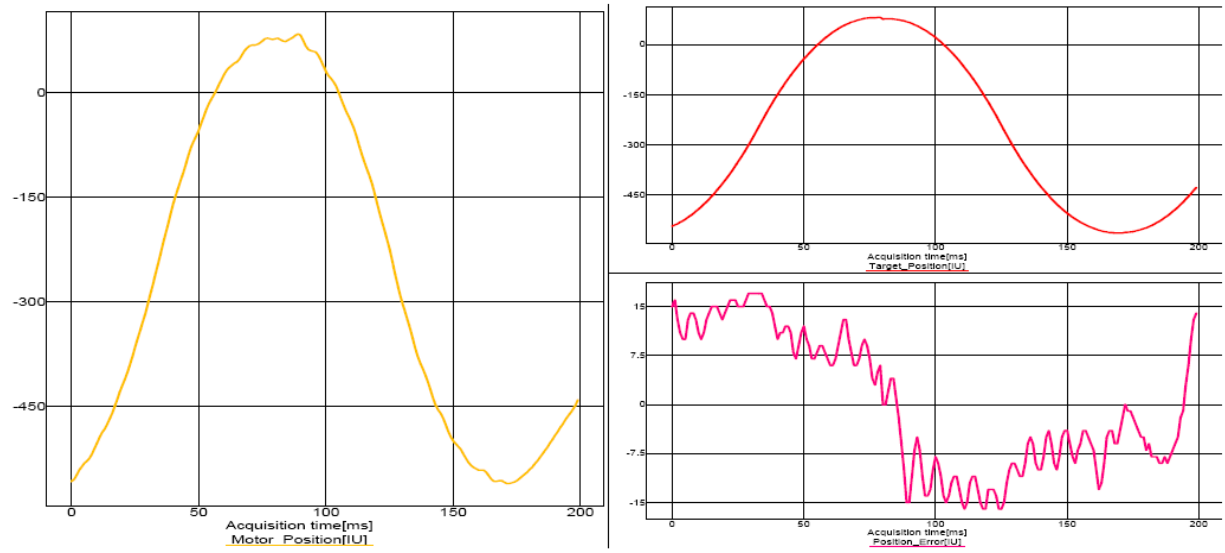

Figure 3. A plot of position controller performance. Left: The motor position, Up Right: The target motor position, Down Right: The position error.

4. Conclusions

In this paper, we presented a control architecture for parallel use of servo motors for two separate functions in a tele-operated robot vision system. The first process includes the automatic mechanical camera stabilization through pan and tilt motors. The second process includes the movement of the stereo vision head from head inertial sensor feedback. Detailed behavior is executed through priorities to prevent the computation waste and simultaneous and non-synchronized accesses to servo motors. The priority feedback scheduler was implemented as an additional real-time periodic task. Experimental results showed that our system is capable of satisfying the hard-real time requirements for the servo control, with great precision. The system has an open modular and scalable architecture with high performance and low cost.

Acknowledgments

This work was supported by the research project RESCUER, FP6-IST-2003-511492 from the European Committee.

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] R. Brega, N. Tomatis, and K. Arras, “The need for autonomy and realtime in mobile robotics: a casestudy of XO/2 and Pygmalion,” in Proc. Of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, vol. 2, 2000, pp. 1422–1427.

[4] R. Kurazume and S. Hirose, “Development of image stabilization system for remote operation of walking robots,” in Proc. IEEE Int. Conf. on Robotics and Automation, vol. 2, 2000, pp. 1856–1861.
[5] J. Lobo and J. Dias, “Vision and inertial sensor cooperation using gravity as a vertical reference,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 12, pp. 1597–1608, 2003.
[6] F. Panerai, G. Metta, and G. Sandini, “Visuo-inertial stabilization in space-variant binocular systems,” Robotics and Autonomous Systems, vol. 30, no. 1-2, pp. 195–214, 2000.
[7] J. Zufferey and D. Floreano, “Fly-inspired visual steering of an ultralight indoor aircraft,” IEEE Trans. Robot. Autom., vol. 22, no. 1, pp. 137–146, 2006.
[8] 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.
[9] J. Knapp and J. Loomis, “Limited field of view of Head-Mounted Displays is not the cause of distance underestimation in virtual environments,” Presence: Teleoperators and Virtual Environments, vol. 13, no. 5, pp. 572–577, 2004.
[10] 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.
[11] OCERA project home page, http//www.ocera.org.
[12] S. Ovaska and S. Valiviita, “Angular acceleration measurement: A review,” IEEE Trans. Instrum. Meas., vol. 47, no. 5, pp. 1211–1217, 1998.
[13] G. Welch and G. Bishop, “An introduction to the Kalman filter,” ACM SIGGRAPH 2001 Course Notes, 2001.
[14] E. Trucco and A. Verri, Introductory Techniques for 3-D Computer Vision. Prentice Hall PTR Upper Saddle River, NJ, USA, 1998.
[15] C. Locke, “Software architecture for hard real-time applications: Cyclic executives vs. fixed priority executives,” Real-Time Systems, vol. 4, no. 1, pp. 37–53, 1992.