The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Task Oriented Control of Smart Camera Systems in the Context of Mobile Service Robots

Hannes Bistry and Jianwei Zhang

Abstract— **In this paper we present our work on enabling smart cameras to act as autonomous vision systems for mobile service robots. The smart cameras will adapt their functionality to the current task of the robot system. Instead of transferring raw image data, higher level image information or regions-of-interest are transmitted. This way, the amount of data is reduced and computationally intensive image interpretation will not affect other tasks of the robot system. We are developing a flexible software solution to integrate smart camera systems to the architecture of our robot system. Each image processing task is constructed by a composition of modular functions that can be distributed over different systems. Timing aspects of the flow of data can be analyzed with different tools to evaluate the performance.**

For this work we are using a commercially available smart camera, but due to the use of a modular architecture the porting to other camera models is easy. We show the advantages of our approach in a setup where image regions containing a face are detected and extracted for further processing steps. This task is accomplished using different setups, where more or fewer subtasks are assigned to the camera system. The performance of the overall system is evaluated with respect to processor load, network load and latency of image data.

I. Introduction

Mobile robot systems will become more and more important in the coming years. Autonomous robots could be used in many applications like assembly, delivery and cleaning tasks or security services. While stationary robot systems are well established at industrial production sites, most of the mobile robot systems are still not suitable for practical everyday use, but are mostly limited to research or demonstration. One main challenge of mobile robot systems is the perception of the dynamically changing environment.

Vision is one of the most sophisticated perceptional capabilities of humans and animals. We are able to localize ourselves in the environment, recognize objects and other persons, control our movements and avoid collisions based on our sense of sight.

Artificial vision systems are quite hard to handle, as there are many drawbacks to deal with. These are mainly:

- implementation effort
- computational effort at runtime
- quality/robustness of image data
- the lack of (artificial) intelligence for scene interpretation

This work is supported by the German BMBF (Bundesministerium für Bildung und Forschung)

Hannes Bistry and Jianwei Zhang are with the Institute of Technical Aspects of Multimodal Systems, Department of Computer Science, University of Hamburg, Germany. {bistry, zhang}@informatik.uni-hamburg.de

Our research focuses on the first three aspects mentioned above. We are investigating a software framework for smart cameras that features distributed processing of image data and intelligent control of capturing parameters. It is highly extensible and portable to make the integration of this framework as easy as possible. Although the architecture is purposed for smart cameras, it can be used for systems with a conventional digital camera as well, yielding the advantage of a unified camera access. In this paper, we will focus on the task-oriented choice of regions-of-interest that are transmitted via Ethernet for further analysis. The robotic task by which we are exemplifying the capabilities of the system is face detection. This task has been chosen because one main challenge of service robots is human interaction. Having the image region containing the face is a prerequisite for further steps like gesture detection or face recognition. We show that our approach can reduce network load, latency and system load compared to conventional Ethernet cameras.

The remainder of this paper is organized as follows: In section II we present our research background and discuss the technical challenges that occurred with the camera systems. We introduce approaches of similar research projects and refer to our prior work in this research field. Section III introduces the developed framework for a wide range of freely programmable camera systems with its underlying part, the open source project GStreamer. We will describe the implementation of a face detection system with three alternative implementations in section IV. Experimental results are discussed in section V. A conclusion and an outlook to future research is given in section VI.

II. Research Background

The TAMS group is doing research on mobile robotic systems with different sensor and actuator systems. The main research platform is the service robot TASER. Descriptions of the whole architecture of TASER can be found in [1]. The purpose of this robot system is to effect delivery tasks in an office environment with subtasks like human interaction, localization, object detection and grasping.

In the following we will only focus on the camera systems of TASER. We will discuss the effect of the drawbacks of the camera systems in terms of connectivity issues, software development and system load. Multiple camera systems are installed (Firewire, USB-Framegrabber), serving different subsystems of the platform.

These different devices are currently controlled by a single industrial PC. This raises the problem that all types of interfaces need to be provided by a single PC. In addition to that,

the operating system must support all the camera devices as well as the other sensors and actuators. Bottlenecks in the Firewire bus systems prevent all cameras from operating at the same time at full capturing speed. These issues are discussed in [2].

For the development of new standalone applications, it is necessary to integrate low-level libraries of the specific camera system. This leads to a high implementation effort, as standard program parts and algorithms need to be implemented again and again with each application. If changes are applied to the camera system, the applications need to be ported (i.e. a Matrox Meteor PCI board for grabbing was replaced by the above-mentioned USB solution).

The camera systems of TASER cause a high system load on the control PC. As this PC is also used for time-critical control tasks on the robot actuators, they cannot be used while image analysis is being done. Most current image processing algorithms have an execution time greater than the reciprocal of the frame rate of digital cameras. If those image processing functions are run continuously, the system load will always be approximately 100 %.

A. Desired Solution

We are investigating a solution where different kinds of sensors are replaced by intelligent sensor systems that communicate over ethernet. This principle has also been successfully applied to laser range finders, as described in [3]. Communication via standard network technology yields the advantage that this technology is supported by a wide range of devices, so there is no need for specialized hardware inside the control PC. Preprocessing of sensor data will reduce the load on the control PC of the robot and therefore ensure that time-critical tasks are carried out properly. Thus the principle of intelligent sensors has also been applied to camera systems. In [4] we proposed a distributed software architecture to integrate smart cameras into a robotic system, taking advantage of the computing power of the devices. The camera can act as an autonomous system. Whenever the main task of the robot system implicates image processing tasks, these can be sourced out to the camera hardware. The main principles of the software will be presented in section III with respect to the scenario we use for evaluation.

B. Related Research

We will look at some research work from different research groups with the focus on intelligent camera systems and active vision. Intelligent image sensors combine photo detectors and processing elements on one image sensor. An application of these kinds of sensors is shown in [5]. At the moment, the resolution is very low and due to the fixed processing strategy the flexibility is strongly limited. Three-dimensional image analysis carried out on dedicated hardware is shown in [6]. In [7] the authors propose an active vision system that can apply various preprocessing functions implemented as dedicated hardware. They introduce an algorithm to track an object based on a template and color segmentation based region-of-interest selection. In [8] a smart camera system intended for tracking applications is shown that can be configured to read out only a partial area of the image sensor and reaches up to 1000 frames per second. An application of face recognition running on a smart camera system is shown in [9]. Due to a DSP-based hardware and an efficient implementation the process of face recognition runs in realtime. Compared to these projects, the innovative feature of our system is that the image processing algorithms are not fixed and can be exchanged at runtime.

III. Control Architecture

In order to integrate smart camera systems into a robot system, we are developing a flexible software architecture that is capable of controlling the camera hardware and carrying out software-based preprocessing.

The basic principles of the developed software layer are summarized below:

- **Processing Chain:** The operations are arranged in a pipeline of processing functions. This pipeline can be modified at runtime.
- **Platform-Independence:** The code should be transferable to other common platforms.
- **Re-usability of code:** We want to build on a huge set of proved library functions and be able to reuse code in different contexts.
- **Timing analysis:** The execution time of each operation should be measurable to carry out performance tests and to achieve flow control.
- **Extensibility:** New image processing functions and support for a new type of cameras should be easy to integrate.

Our distributed software system can control the processing function of many cameras from a single instance. The software is written for Linux, but could be transferred to other operating systems, too. Transfer and processing of image data is done within the GStreamer [10] framework. This open-source multimedia framework is supposed to be an equivalent of DirectShow [11] that is running under most common operating systems including Linux. Many functions needed for this application are already implemented in GStreamer, like format conversion, image resizing, encoding, decoding, timing issues and network data transmission. The GStreamer framework is plug-in-based, so the functionality can be expanded by new plugins that also can define their own data types. The elements are connected to a processing pipeline, so that many operations can manipulate image data consecutively. Each Elements can act as a data source (i.e. camera driver, network-receive, read file), as a filter that runs processing functions on image data or as an output plugin (i.e. network-send, show video data, write files). There is also the possibility to set up non-scalar pipelines where data of one image is processed by many elements in parallel. In this case, copies of image data are only made if any element wants to manipulate image data.

Image processing functions can also be run on dedicated systems without camera hardware, so the tasks can run on many systems in parallel. This can be useful when the

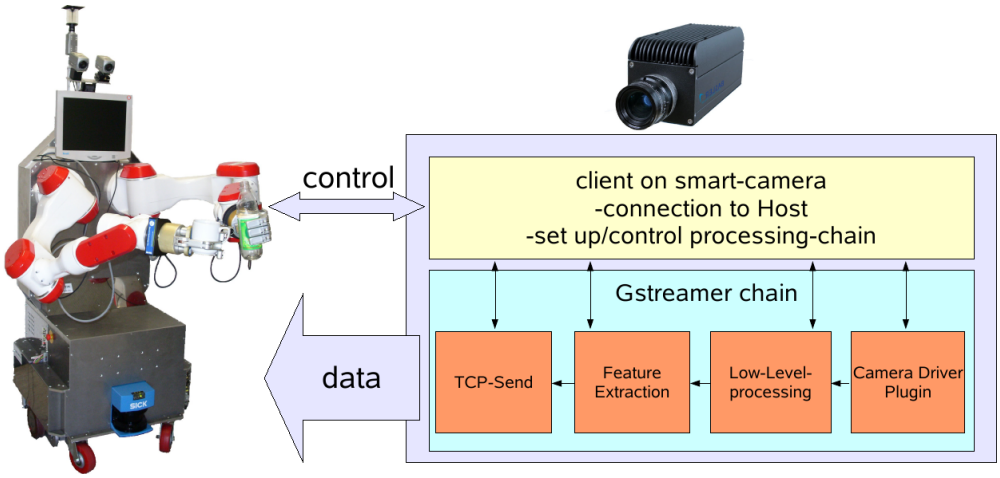

Fig. 1. Connection between TASER and the smart camera

computational effort is large compared to the additional overhead of transferring image data over a network.

Timing issues can be analyzed by the so-called timestamps that every unit of data (i.e. one image of a videostream) provides. Normally, this field of data is used for the streamtime of a video or audio file, starting with zero. In our case, we exchange this value with the current NTP [12] timestamp directly after the frame was captured. Therefore, we synchronize all systems including the smart camera by an NTP timeserver. The achievable accuracy is better than 1 ms in a local area network. Considering the usual processing times of image processing functions and jitter caused by the scheduler of the operating system, this accuracy is sufficient.

Our software-framework is tested with the commercially available smart-camera Basler "eXcite exA1390-19c" [13]. The device features an MIPS 1 GHz processor and has a resolution of 1388 x 1038 pixels. In the following setups the camera will be configured to output grayscale images with 8-bit resolution. The maximum framerate depends on the shutter time and is between 18 and 19 fps. A plug-in integrating the camera in the GStreamer framework has already been developed, tests concerning the performance of the camera are in progress. The setup of the connection is shown in figure 1.

IV. Application of the Smart Camera System

In this section, we compare three possible setups of the camera system in the use case of face detection. This task has been chosen both because of its relevance for mobile service robots and because it shows the capability of our framework to adapt to the performance of current smart camera systems. The camera system is intended to transmit only those image regions at native resolution where faces occur. Thus network bandwidth will be saved because only the data needed for further steps like face recognition will be transmitted. A face detection element has been set up using Haar classifiers [14] that are implemented in the OpenCV framework [15]. Prior tests have shown that running the face detection algorithms with a reduced image of 320x240 pixels is sufficient for a robust detection. Therefore, image data is scaled down before face detection, yielding the advantage of lower computing time and implicit noise reduction by subsampling. Nevertheless having the high resolution data is sufficient for further steps. In [16] it is shown that the accuracy of face recognition depends on a sufficient resolution of image data.

The three different setups differ in the allocation of tasks. For each configuration, we set up two processing pipelines, one on the smart camera system, and one on the control PC of the service robot. For these tests an Intel Pentium D with 3 GHz was used. The flow of data is established via TCP connections. In the first setup the smart camera system is configured to act as a conventional Ethernet camera. Image data is transferred to the control PC at full resolution. The control PC performs face detection and extraction of the image regions. Within the second setup, the camera system transmits a scaled-down video stream (320x240) to the control PC which performs the face detection and transmits coordinate information back to the smart camera system, where these regions are extracted and sent to the control PC. In the third setup the smart camera system accomplishes all tasks mentioned above and transmits only regions-of-interest and coordinate information. Detailed information about the setup is shown in figure 2.

Since face detection cannot be accomplished for every image at full frame rate, measures have to be taken to deal with the problem that the queue of images waiting to be processed will permanently grow. Therefore, we set up a caching element preliminary to the face detection element that is configured to keep only the latest image and drop all older ones. This way, no processing time is wasted for old data, the average latency is reduced and the system will not run out of memory. The same dropping strategy is applied to all network connections.

The region-of-interest extraction can work in two different modes. It can either force synchronization between the position information and the full resolution image or not. If this element works unsynchronized, it will use the latest available

position information for every image, yielding the advantage of a higher frame rate with a good chance of gathering useful information. Image information will generally be available earlier on the host PC. The disadvantage of the unsynchronized strategy is that especially for moving persons and setups with a slow refresh rate of coordinate data, the extracted region will not match the actual position of the face anymore. In the synchronized mode only regions-of-interest are transmitted if coordinate information is available for exactly this frame. Otherwise, image data is dropped.

To simulate conditions like they are while the service robot is carrying out additional tasks, system load is caused by simple dummy tasks that run in parallel with image processing. These are three "busy-waiting" tasks and one task that receives data over Ethernet connection from a third system at a data rate of 20 MB/s.

Within each test period of approximately 30 seconds a person is walking along a path in front of the robot system. The latency of image and position data as well as system and network load is permanently measured and logged to a file. The three different pipeline-setups, the choice whether to force synchronization or not, and the possibility to add additional load lead to twelve possible combinations. For each of them the test is run once. Several aspects are analyzed to evaluate the performance of the overall system. Their impact will be discussed in the following:

- **System load on the control PC of the robot:** The system load has been evaluated by reading out the virtual file "/proc/stat". All tasks running on the control PC of the robot will influence this value. If no additional load is generated, it nearly represents the computational effort of the assigned tasks. Taking into account that many sensor and actuator systems have to be controlled by this PC and that several higher-level tasks need to be carried out, the load generated by image processing must not be too high. Within testing procedures with additional load this value is meaningless.
- **System load on the smart camera:** The evaluation method is the same as for the system load on the PC. This variable has been analyzed to determine how the different setups use the capacity of the CPU of the smart camera. It can also be derived whether the camera provides additional computing capabilities for further processing functions.
- **Network Load:** The network load has been measured in each of the TCP elements used in the setup. Only the payload has influence on this value. As more and more sensors are replaced by intelligent sensor systems with ethernet connection, it should be kept in mind that the maximum bandwidth of the network infrastructure must not be exceeded. A high network load causes higher latencies for the different devices that communicate via Ethernet.
- **Position updates per second:** Each time the face recognition of one image finishes, a dataset of coordinates is generated. The number of these packets per second is analyzed by the timing element. If the system is working in the synchronized mode, this number also represents the update frequency of the image regions (presuming that all could be transmitted in time).
- **Age of position data:** This value is also measured in the timing element within the processing pipeline. Due to latencies caused by data transfer and caching, this value can be greater than the reciprocal of the number of position updates. This value is important as the robot system should react to events in the environment in time.
- **Age of image data:** This value is also checked inside the pipeline. In the unsynchronized operation mode, it mainly represents the time needed for extracting the regions and data transmission. In the synchronized mode, it is nearly the same as the age of the position data, but due to the implementation of the region extraction element and the bigger data size, image data is available a short time later.

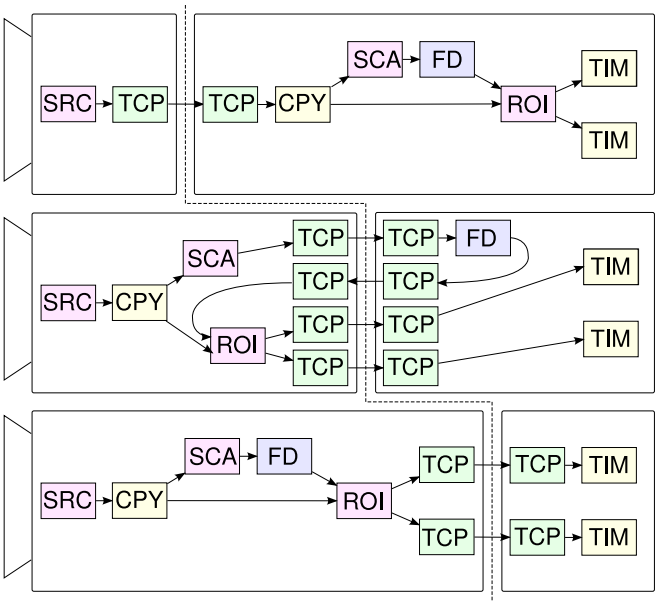

Fig. 2. These figure shows the three different setups of the processing pipelines. The smart camera system (left side) communicates with the control PC of the service robot (right side) via TCP connections. At the end of the pipeline the timestamp of the extracted regions and the coordinate data is compared to the current time.
Within the first setup (top) the smart camera acts like a conventional Ethernet camera. All relevant tasks are accomplished by the PC. The second configuration (middle) divides the tasks in a way that the smart camera system generate a low resolution video stream, that is analyzed for faces on the control PC. Coordinate data is sent back to the smart camera, where these regions are extracted and transmitted. In the third setup (bottom) all processing functions run on the smart camera and only the image regions and coordinate information are transmitted.
Elements that are used in this setup:
SRC: access to the camera driver libraries, generating raw image data
CPY: duplicates data (mostly pointer based)
SCA: scale element, modifies the resolution of images
FD : face detection using Haar classifiers, generates list of coordinates
ROI: extracts the regions-of-interest and sends them sequentially, sends current position information via the second stream
TCP: transmits data through a TCP connection
TIM: compares the timestamp of data to current NTP time

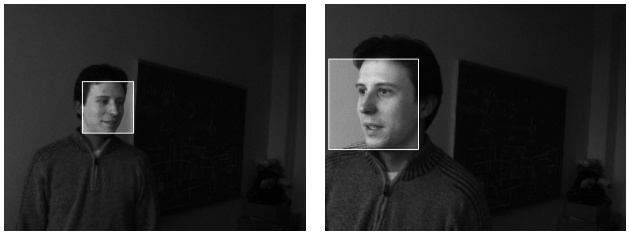

Fig. 3. Visualization of the extracted region, superimposed above the low resolution video stream (grayed out)

V. Experimental results

The detection of faces worked reliably with every configuration. For visualization purposes, the extracted regions are superimposed above the grayed out low resolution video stream. Figure 3 shows two arbitrary images of the testing procedure.

The most significant results are visualized in figure 4. A complete overview of all evaluated values is shown in table 5. Using smart camera systems yields a major advantage compared to the use of traditional camera systems. Especially the second configuration clearly offers the best overall performance. The amount of data is reduced to about 10 % of the initial value, the latency of image data is reduced noticeably. The continuous stream (18.5 fps) of 320x240 pixel generates only 1.4 MB/s compared to the 27 MB/s needed to transfer all native resolution images. Due to the fact that this amount of data does not need to be handled by the control PC, the reduction of the system load is also noticeable. The actual data rate strongly depends on the number of faces in the scene and the distance of each face towards the camera.

The third configuration shows that the complexity of face recognition exceeds the capabilities of current smart camera systems. If the occurring latencies of over 500 ms are acceptable, it can still be useful, because it barely generates load on the control PC and on the network. Especially if image processing is considered as a background task and the main task is something different, it is still possible to detect faces and trigger events. Due to the technical progress future types of smart camera systems will be able to carry out the task at higher speed. The basic setup of the image processing pipelines can be retained for different criteria for the region-of-interest detection, like motion detection or color histogram matching. Using these less computationally intensive functions, setups where these functions run directly on the camera system will perform even better than a distributed processing like in the second setup.

All measurement values are influenced by many factors, mainly the scheduling order of the operating system, additional tasks and dependency on the actual content of the scene. Therefore, these values can only give a rough estimation of the behavior. Nevertheless some unexpected measurement values occur in the table of the test results. One of theses issues is the high processor load on the smart camera systems in the first setup. This is caused by an inefficient implementation of the network interface and low memory bandwidth. In the tests with additional load the amount of data transmitted decreases. The generated network load leads to delays at the transfer of image data, thus images get dropped directly on the camera and do not need to be transferred. A slight decrease of the load in the smart camera is an effect of this circumstance. Also quite unexpected was the fact that the number of position updates decreased in the second setup compared to the first setup (both with additional load). A reason could be that sometimes two or more scaled down image buffers have already been received in the receive buffer when the corresponding thread gets called and as a result they get dropped. The native resolution image buffers could be less likely to cause drops. Thread scheduling and frame dropping can also explain why the number of position updates per second (setup 1 and 2, no additional load) is beneath the framerate of the camera, although the CPU is not fully loaded. A short test where the threshold for dropping was increased confirmed this: The update rate and the processor load increased, but so did the average latency due to the fact that image data is queued for a longer time.

VI. Conclusions and Outlook

In this paper, we presented our work on a framework for intelligent camera systems. The image processing functions on the camera systems can be adapted according to the current task of the robot. We showed different configurations where the camera is configured to detect regions of faces and transmits only these regions to reduce the network load and thus the amount of data that needs to be processed by further systems. Performace analysis has shown that current smart camera systems can reduce the system- and network load of the robot system and reduce the latency of image data. The innovative streaming concept makes it possible to take advantage of high-resolution cameras and handles the high amount of image data efficiently. These results form a basis for the integration of multiple intelligent cameras on one system and also make it possible to use cameras with even higher resolutions.

Future types of smart camera systems will feature more powerful hardware and will be able to carry out complex image processing algorithms in realtime. Due to the portability of our framework we could easily take advantage of additional computing power by a simple reconfiguration of processing elements. Having regions-of-interest detected opens up new possibilities in the control of the capturing parameters, that can be optimized for these regions. This way, the camera system will aim to generate robust and high-quality image data.

References

[1] T. Baier, M. Huser, D. Westhoff, and J. Zhang. A flexible software architecture for multi-modal service robots. *Computational Engineering in Systems Applications, IMACS Multiconference on*, 1, 2006.

[2] H. Bistry, S. Pohlsen, D. Westhoff, and Jianwei Zhang. Development of a smart laser range finder for an autonomous service robot. *Integration Technology, 2007. ICIT '07. IEEE International Conference on*, pages 799–804, March 2007.

strategy	sync	additional load	∅ load PC	∅ load eXcite	∅ network load	∅ position updates [1/s]	∅ age of position	∅ age of image content
1	no	no	60 %	74 %	27.5 MB/s	11.9	202 ms	92 ms
2	no	no	45 %	15 %	2.1 MB/s	12.1	134 ms	9 ms
3	no	no	3 %	95 %	0.7 MB/s	1.8	644 ms	11 ms
1	yes	no	62 %	73 %	27.3 MB/s	10.6	182 ms	184 ms
2	yes	no	40 %	16 %	1.7 MB/s	12.5	122 ms	138 ms
3	yes	no	2 %	94 %	0.2 MB/s	2.0	621 ms	626 ms
1	no	yes	99 %	60 %	23.5 MB/s	8.4	246 ms	115 ms
2	no	yes	99 %	16 %	2.0 MB/s	7.4	172 ms	13 ms
3	no	yes	99 %	97 %	0.6 MB/s	2.1	540 ms	13 ms
1	yes	yes	99 %	58 %	22.5 MB/s	8.0	262 ms	263 ms
2	yes	yes	99 %	16 %	1.6 MB/s	7.2	182 ms	220 ms
3	yes	yes	99 %	94 %	0.2 MB/s	2.0	662 ms	667 ms

Fig. 5. Evaluation results of the different test configurations

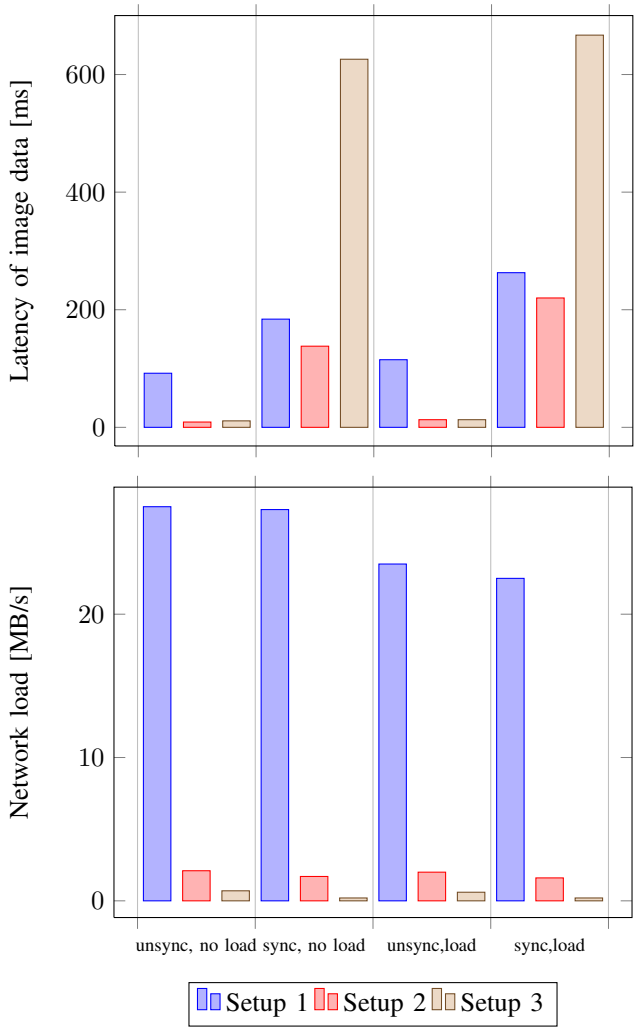

Fig. 4. Most significant test results. The latency and the network load are compared for the three different setups.
Setup 1: Classical Ethernet Camera
Setup 2: Distributed Processing
Setup 3: All Tasks on Smart Camera

[3] H. Bistry, D. Westhoff, and J. Zhang. A smart interface-unit for the integration of pre-processed laser range measurements into robotic systems and sensor networks. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 358–363, 29 2007-Nov. 2 2007.

[4] A. Maeder, H. Bistry, and J. Zhang. Intelligent Vision Systems for Robotic Applications. *International Journal of Information Acquisition (IJIA)*, 5(3):259 – 267, September 2008.

[5] Atsushi Konno, Ryo Uchikura, Toshiyuki Ishihara, Teppei Tsujita, Takeaki Sugimura, Jun Deguchi, Mitsumasa Koyanagi, and Masaru Uchiyama. Development of a high speed vision system for mobile robots. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1372–1377, Oct. 2006.

[6] Li Mingxiang and Jia Yunde. Stereo vision system on programmable chip (svsoc) for small robot navigation. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1359–1365, Oct. 2006.

[7] P. Chalimbaud and F. Berry. Embedded active vision system based on an FPGA architecture. *EURASIP Journal on Embedded Systems*, 2007(1):26–26, 2007.

[8] I. Ishii, K. Kato, S. Kurozumi, H. Nagai, A. Numata, and K. Tajima. Development of a mega-pixel and milli-second vision system using intelligent pixel selection. In *First IEEE Technical Exhibition Based Conference on Robotics and Automation, 2004. TExCRA'04*, pages 9–10, 2004.

[9] H. Fatemi, R. Kleihorst, H. Corporaal, and P. Jonker. Real time face recognition on a smart camera. *Proceedings of ACIVS 2003 (Advanced Concepts for Intelligent Vision Systems)*, 2003.

[10] W. Taymans, S. Baker, A. Wingo, R. Bultje, and S. Kost. *GStreamer Application Development Manual (0.10.21.3)*. http://gstreamer.freedesktop.org, October 2008.

[11] Mark D. Pesce. *Programming Microsoft DirectShow for Digital Video and Television (Pro-Developer)*. Microsoft Press, 5 2003.

[12] David L. Mills. *Computer Network Time Synchronization: The Network Time Protocol*. CRC, 1 edition, 3 2006.

[13] Basler AG. *Basler eXcite User's Manual*, May 2006.

[14] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. *IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION*, 1, 2001.

[15] G. Bradski. The openCV library. *DOCTOR DOBBS JOURNAL*, 25(11):120–126, 2000.

[16] Y. Utsumi, Y. Iwai, and M. Yachida. Performance evaluation of face recognition in the wavelet domain. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3344–3351, Oct. 2006.