A Smart Camera for Traffic Surveillance

Michael Bramberger[1], Roman P. Pflugfelder[2], Arnold Maier[1]
Bernhard Rinner[1], Bernhard Strobl[3], Helmut Schwabach[3]

[1]Institute for Technical Informatics,
Graz University of Technology, A–8010 Graz, Austria
{bramberger,maier,rinner}@iti.tugraz.at

[2]Pattern Recognition and Image Processing Group,
Vienna University of Technology, A–1040 Vienna, Austria
romanp@prip.tuwien.ac.at

[3]Video & Safety Technology,
ARC Seibersdorf research, A–2444 Seibersdorf
{bernhard.strobl,helmut.schwabach}@arcs.ac.at

Abstract — *The integration of advanced CMOS image sensors with high-performance processors into an embedded system facilitates new application classes such as smart cameras. A smart camera combines video sensing, video processing and communication within a single device. This paper reports on the prototype implementation of a smart camera for traffic surveillance. It captures a video stream, computes traffic information and transfers the compressed video stream and the traffic information to a network node. The achieved experimental results of the implemented stationary vehicle detection demonstrate the feasibility of our approach.*

1 Introduction

Due to their logarithmic behavior, high dynamic range and high bit resolution the low-cost and low-power CMOS sensors acquire images with the necessary quality for further image processing under varying illumination conditions. The integration of these advanced image sensors with high-performance processors into an embedded system facilitates new application classes such as *smart cameras*. Smart cameras not only capture images or video sequences, they further perform high-level image processing such as motion analysis and face recognition on-board and transmit the (compressed) video data as well as the extracted video information via a network.

An important application area where smart cameras can potentially and advantageously replace most known cameras, frame grabbers and computer solutions is visual traffic surveillance [1]. CMOS image sensors can overcome problems like large intensity contrasts due to weather conditions or road lights and further blooming, which is an inherent weakness of existing CCD image sensors. Furthermore, noise in the video data is reduced

by the capability of video computation "close" to the CMOS sensor. Thus, the smart camera delivers a new video quality and better video analysis results, if it is compared to existing solutions. Beside these qualitative arguments and from a system architecture point of view, the smart camera is an important concept in future digital and heterogeneous third generation visual surveillance systems [2]. Not only image enhancement and image compression but also video computing algorithms for scene analysis and behavior understanding are becoming increasingly important. These algorithms have a high demand for real-time performance and memory. Fortunately, smart cameras can support these demand as low-power, low-cost embedded systems with sufficient computing performance and memory capacity. Furthermore, they offer flexible video transmission and computing in scalable networks with thousands of cameras through a fully digital interface.

The purpose of this paper is to present first results of an ongoing research project between ARC Seibersdorf research, the Institute for Technical Informatics at Graz University of Technology and the pattern recognition and image processing group at Vienna University of Technology. The primary goal of this project is the development of a smart camera for traffic surveillance. This paper presents the camera's prototype implementation and a case study of our smart camera concept with respect to stationary vehicle detection in tunnels. We chose this application, because it is by far the most important application (80 percent of all applications) in traffic surveillance.

The remainder of the paper is organized as follows: Section 2 presents the requirements of a smart camera and lists the related work. The hardware architecture is outlined in Section 3, while the software components required by the smart camera are depicted in Section 4. Experiments and a prototype description is described in Section 5. Section 6 concludes this paper with a short discussion.

2 Requirements and Related Work

2.1 Requirements of a Smart Camera

In general a smart camera is compromised of a *sensor*, a *processing* and a *communication* unit. In this section we briefly discuss the requirements for each of these units as well as some system wide requirements.

2.1.1 Sensor Requirements

The image sensor is the prime input for a smart camera. An appropriate image quality is, therefore, essential for the performance of the entire system.

Dynamic Range Traffic surveillance applications enforce high demands on the image sensor. Typical traffic situations may contain a high dynamics, e.g., when high-intensity areas, such as the high-beam of a vehicle, appear concurrently with low-intensity areas such as the car's silhouette at night. Image sensors with high dynamic range and little blur are preferred for these applications. Additionally, high dynamic-range sensors ease the design of the camera control and the control of the lens aperture in changing light conditions.

Resolution and Frame Rate Many available image sensors feature only small image formats such as CIF and QCIF. These formats are acceptable for cell phones. However, surveillance cameras require a larger resolution due the requirements of the image processing and the operators. Note that many currently available surveillance systems deliver images in PAL resolution (720x576 pixels). Most image processing algorithms for the smart camera are based on monochrome input, however, the operators prefer color images for manual surveillance.

The maximum frame rate (in fps) is another important parameter of the smart camera. It is determined by the image sensor and succeeding image processing stages. A frame rate of 15 fps is aimed for live video and fast response times of the image processing tasks.

Digital Interface In order to reduce the effect of temperature drift and aging as well as to avoid glue logic the image sensor has to deliver digital video output. Thus, the sensor has to include analog amplifiers and ADCs.

2.1.2 Processing Requirements

There are various tasks that must be executed by the processing subsystem including (i) video compression, (ii) video analysis, (iii) the computation of traffic statistics, and finally (iv) the camera control and firmware.

Video Compression To reduce the required transmission bandwidth the video stream must be compressed. The corresponding video encoder is therefore executed on the smart camera.

Video Analysis The on-board video analysis transforms the digital network image sensor into a smart camera. The video analysis extracts an abstract scene description out of the raw video data. This description is then exploited for example to detect stationary vehicles, to detect lost objects, or to detect wrong-way driver. If an extraordinary situation has been detected by the video analysis an alarm is triggered and sent to the central control station.

Computation of Traffic Statistics Finally, traffic statistics are computed out of the video stream. These statistics include the number of vehicles per time interval, the average speed of vehicles per vehicle class or the lane occupancy. These parameters are transmitted on demand or continuously to a network node.

Camera Control and Firmware Another important tasks of the processing subsystem are the camera control, i.e., aperture and flash control, and the firmware of the embedded software. The firmware includes the control of all peripherals, the management of all software tasks, and software reconfiguration via the network interface.

2.1.3 Communication Requirements

The compressed video stream and the output of the video analysis are transferred to the control station via the communication unit. For a flexible and fault-tolerant communication different network connections such as Ethernet, wireless-LAN and GSM/GPRS should be possible.

Beside the standard data upload the smart camera must also support data download to enable to change the configuration or firmware of the camera via the network.

2.1.4 System Requirements

Low-Power Power consumption is a major design constraint in recent embedded systems. High power consumption reduces the operation time in battery or solar-powered environments. Another important aspect is heat dissipation which must be low in order to avoid active cooling. Active cooling, e.g., fans, increases size and costs as well as limits the camera's area of application.

Real-Time The smart camera has various requirements concerning its firm and soft real-time performance. There are several timing constraints concerning the camera control and the peripherals, e.g., the flash trigger. There are also timing constraints for the image analysis algorithms, e.g., a stationary car has to be detected within 6 seconds.

2.2 Related Work

The smart camera presented in this paper is a critical step towards a high-performance embedded surveillance system. In [3], Wolf et al. presented a PC-based smart camera system using analog cameras. We decided to enhance the concept to design a fully-integrated embedded system to reduce power consumption and size as well as to improve processing performance and flexibility. Although smart cameras are already commercially available [4], the possibilities of embedded systems are not exploited there. High-level image analysis algorithms require processing power that current systems are not able to deliver.

Some commercial and scientific systems have been proposed for traffic surveillance in the past. These systems extract quantitative information, e.g. speed, volume, and qualitative information such as unusual events like stationary vehicles, from the image data. All these systems use either a background/foreground segmentation in combination with object tracking [5][6][7] or analyze long-term background changes in a statistical background model [8]. Background/foreground segmentation can be done by frame differencing which is simple and surprisingly robust to illumination changes, e.g. car lights. As this method has problems within the objects due to small illumination changes, more sophisticated, adaptive methods like adaptive background models [9] or pixel-wise background estimation techniques are used [10][11]. A difficult problem is always at which time the background model should be updated. We propose a background/foreground segmentation that is a combination of frame differencing and a Gaussian background model. The Gaussians are only updated under certain conditions with respect to the current observation distribution (last k frames) and the actual frame difference. The model is analyzed over time for long-term background changes. Pixel outliers are eliminated by morpholog-

ical voting. We also correct the background model after the voting. First results show that the method is robust with respect to noise and real-time capable.

3 Architecture of the Smart Camera

3.1 System Overview

For traffic surveillance the entire smart camera is packed into a single cabinet which is typically mounted in tunnels and aside highways. The electrical power is either supplied by a power socket or by solar panels. Thus, our smart camera is exposed to harsh environmental influences such as rapid changes in temperature and humidity as well as wind and rain. It must be implemented as an embedded system with tight operating constraints such as size, power consumption and temperature range.

3.2 Architecture

As depicted in Figure 2.1, the smart camera is divided into three major parts: (i) the video sensor, (ii) the processing unit, and (iii) the communication unit.

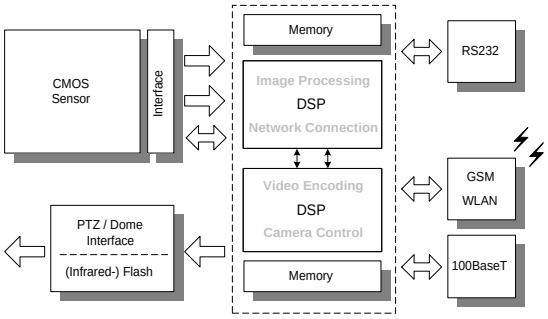

Figure 1: System architecture of the smart camera.

Video Sensor The video sensor represents the first stage in the smart camera's overall data flow. The sensor captures incoming light and transforms it into electrical signals that can be transferred to the processing unit.

A CMOS sensor best fulfills the requirements for a video sensor. These sensors feature a high dynamics due to their logarithmic characteristics and provide on-chip ADCs and amplifiers.

Our first prototype of the smart camera is equipped with the *LM-9618* CMOS sensor from *National Semiconductor*. Its specification is enlisted in Table 1.

Dynamic range	Type	Resolution	Max. fps	ADC-Resolution	Sensor control
100 dB	Monochrome	640×480	30	12 bit	I^2C

Table 1: Video sensor specification.

Processing Unit The second stage in the overall data flow is the processing unit. Due to the high-performance on-board image and video processing the requirements on the computing performance are very high. A rough estimation results in 10 GIPS computing performance. These performance requirements together with the various constraints of the embedded system solution are fulfilled with *digital signal processors (DSP)*. The smart camera is equipped with two TMS320DM642 DSPs from Texas Instruments running at 600 MHz. Both DSPs are loosely coupled via the *Multichannel Buffered Serial Ports (McBSP)*, and each processor is connected to its own local memory.

The video sensor is connected via a FIFO memory with one DSP to relax the timing between sensor and DSP. The image is then transferred into the DSP's external memory with a capacity between 8 MB and 256 MB.

Communication Unit The final stage of the overall data flow in our smart camera represents the communication unit. The processing unit transfers the data to the processing unit via a generic interface. This interface eases the implementation of the different network connections such as Ethernet, wireless LAN and GSM/GPRS. For the Ethernet network interface only the physical-layer has to be added because the media-access control layer is already implemented on the DSP.

A second class of interfaces is also managed by the communication unit. Flashes, pan-tilt-zoom heads (PTZ), and domes are controlled using the communication unit. The moving parts (PTZ, dome) are typically controlled using serial interfaces like RS232 and RS422. Additional in/outputs are also provided, e.g., to trigger flashes or snapshots.

3.3 Low-Power Considerations

The key power saving strategy used in our smart camera is based on system-level *Dynamic Power Management (DPM)* [12].

3.3.1 Dynamic Power Management

The basic idea behind DPM is that individual components can be switched to different *power modes* during runtime. Each power mode is characterized by a different functionality/performance of the component and the corresponding power consumption. For instance, if a specific component is not used during a certain time period it can be switched off.

The commands to change the components' power modes are issued by a central *Power Manager (PM)*. The commands are issued corresponding a *Power Managing Policy (PMP)*. The PMP is usually implemented in the operating system of the main processing component.

In order to decide which command to issue the PM must have knowledge about the system's workload. Note that switching the component's power mode requires also some time. Thus, the PM must include these transition time in its PMP in order to avoid malfunction of the system.

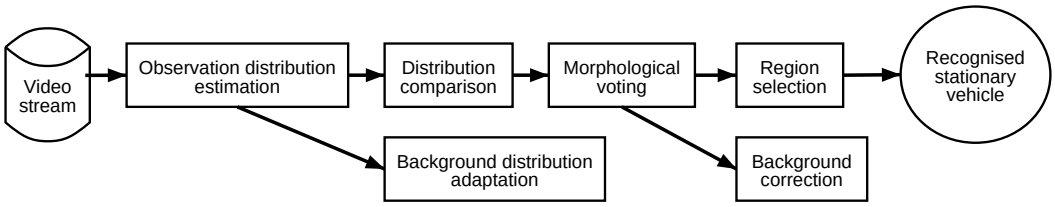

Figure 2: A sketch of the video processing algorithm for stationary vehicle detection

3.3.2 DPM in the Smart Camera

In the smart camera the PM is located in the OS kernel of the host DSP. The power modes of the individual components are controlled by sending individual commands via the I^2C bus. Each component has its specific power modes comprising different power consumption, speeds and wake-up times. These characteristics are stored in look-up-tables in the PM and are used as input for the PMP.

If the smart camera is running in *Normal Mode* (Section 4.4), for instance it is not necessary to run the corresponding DSPs in full power mode. In this case, the PM sets the DSPs into a lower power mode in a way that real time requirements are still met. If the camera changes its operating mode to the *Alarm Mode* (Section 4.4) the PM sets all components back to their high-performance modes.

4 Software

4.1 Image Processing

To demonstrate the video processing capabilities of the smart camera we chose the area of stationary vehicle detection, which is an important application in traffic surveillance. The qualitative decision is based on long-term intensity changes of background pixels. We focused on the tunnel environment, because background modelling is simpler compared to an outdoor scene with for example swaying trees. As more work will be done, especially for outdoor scenes, we paid attention to design a smart camera for future algorithms as well as to incorporate the video processing algorithms for this application.

It was assumed that the camera is static and the ambient light conditions are constant. Thus, intensity changes (foreground) are only caused by the motion of vehicles or by noise, e.g. reflections, lights of cars. Figure 3(c) shows a sample forground. Intensity values are grey values between 0 and 255. Pixels $(x\ y)^T$ of an image are semantically background pixels, if the difference $I_{t-1}(x, y) - I_t(x, y)$ of two consecutive images $I_{t-1}(x, y)$ and $I_t(x, y)$ is smaller than a threshold (stationary case) or the intensity value of a pixel is supported by a distribution over background intensity values (statistical case). Because we assume a static background, the intensity values of background pixels can be described by a Gaussian distribution.

Figure 2 sketches the stationary vehicle detection algorithm. Each pixel's background model is initialized with a Gaussian, which covers the whole intensity range. Then, an observation distribution is updated for each pixel with every new available image of the video stream. The mean and the variance of the observation distribution are estimated by the sample mean and sample variance over the last k images. To make the parameters

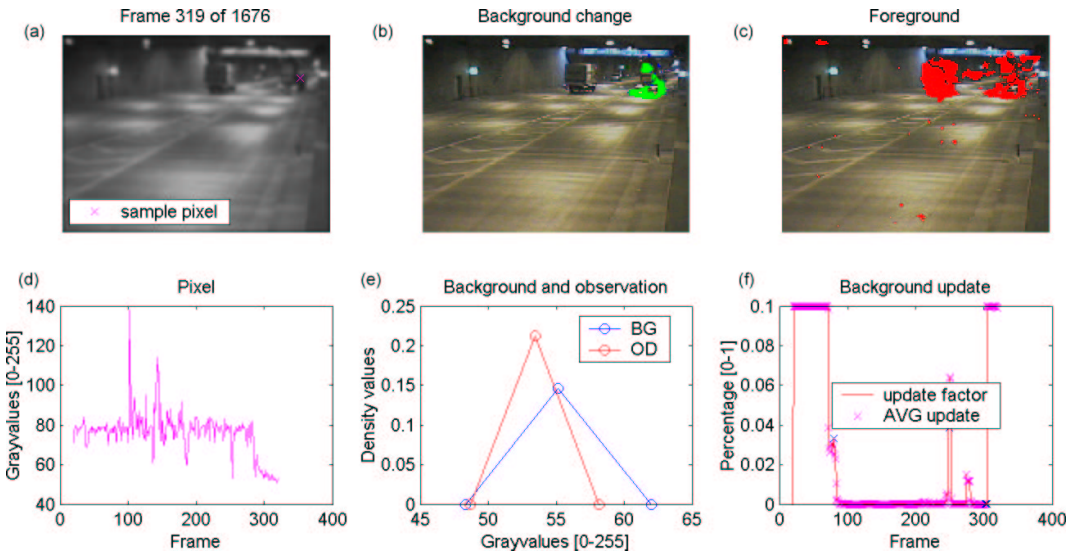

Figure 3: Stepping through the algorithm: The first row shows (a) a specific frame with a stationary truck and a sample pixel, (b) the selected long-term background change regions and (c) the foreground regions. The second row shows analysis results for the sample pixel: (d) the intensity profile, (e) the significant part of the background and observation distribution and (f) the update factor over time.

more stable over time, the image is spatially convolved with a Gaussian filter with a standard deviation of 1.3 before parameter estimation (see figure 3(a)).

In each step, the observation and background distribution are compared as it is shown in figure 3(e). If the significant parts, i.e. between 25%-quantile and 75%-quantile, of both distributions do not intersect each other, then a statistical long-term intensity change in this particular pixel is detected. To make the algorithm robust, a further morphological voting step in the 8×8 vicinity of this pixel is done. If the majority of neighbored pixels do not show the same separation of background and observation distribution, then the gap between both distributions is closed by an updated broader background model (background correction).

Regions of pixels with robust changes are selected by a connected component algorithm (see figure 3(b)). Stationary vehicles are detected, if two events happen: (i) The area of a region lies between a minimal and maximal threshold and (ii) each intensity profile (see figure 3(d)) $I_{t-k}(x,y), \ldots, I_t(x,y)$ over the last k images of all pixels of a region shows a difference $I_{i-1}(x,y) - I_i(x,y)$, $i \in [t-k+1, t]$ greater than δ. Besides, (ii) takes into account that the statistical background change was triggered by an abrupt intensity change, i.e. through a vehicle.

The adaptation of the background model is realized with respect to the current observation distribution. Beside the case of the separation of both distributions, further three cases can be distinguished for adaptation:

(1) The observation distribution is inside the background model

(2) The background model is inside the observation distribution

(3) Background model and observation distribution intersect each other

In (1), the background model is too broad. Therefore, the parameters of the model are updated towards the parameters of the observation distribution. This is realized by exponential averaging[13]. The mean of the background model μ_{BG} is updated with the mean of the observation distribution μ_{OD} by $(1-\alpha)\mu_{BG} + \alpha\mu_{OD}$. α defines how quickly the background model is updated towards the current observation distribution. In (2) the background model is only updated by exponential averaging, if the difference of the inter-quantile ranges of both distributions is small. This is defined by a new adaptation factor

$$\frac{\alpha}{e^{a(iqr_{OD} - iqr_{BG})}}$$

a defines the sensitivity with respect to the inter-quantile range difference $iqr_{OD} - iqr_{BG}$. In (3) the adaptation rule is similar to the adaptation in (2). The background model intersects the observation distribution from the left or from the right side. Accordingly, the 75%-quantile or 25%-quantile of the background model is updated towards the 75%-quantile or 25%-quantile respectively of the observation distribution. The mean of the background distribution remains the same. Thus, the background model broadens and supports more of the current observation intensities. Figure 3(f) shows the update factor over time. The crosses show frames with background updates ("AVG update") according to adaptation case (1).

In case of global intensity changes due to an ambient light change, a significant number of pixel intensities will not be supported by the current background models. Then, all pixel background models are reset to the initialization models.

4.2 Video Compression

Video transmission at full PAL resolution and at 25 fps requires a bandwidth of approximately 20 MB/s[1]. State-of-the-art video compression reduces the bandwidth needs by factor of 100 down to 1.5 Mb/s.

The advanced simple profile MPEG-4 encoding method is well-suited for traffic surveillance, since the encoded quality, and therefore the required bandwidth can be adapted to different needs. This MPEG-4 compression module is supplied by a DSP- software provider. Performance data reports a required processing power of approximately 4000 MIPS at full PAL resolution running at 25 fps.

4.3 Network Connection

Internet connections require the TCP/IP protocol implemented as an *IP stack*. This software module manages the network traffic with its various protocols like HTTP, FTP, and UDP. The smart camera uses HTTP to provide a flexible and user-friendly user interface for operators to adapt parameters, and to check the camera's vitality and log files. FTP is used to download data stored in files to the camera like firmware updates or new parameter sets. Finally, UDP is used for multicast streaming-video transmission.

The smart camera uses an implementation of an IP stack from Windmill Innovations / Traquair Data Systems, which is optimized for Texas Instruments' TMS320C6x series DSPs.

[1] Assuming 4:2:0 video format and 8 bits per pixel ADC resolution

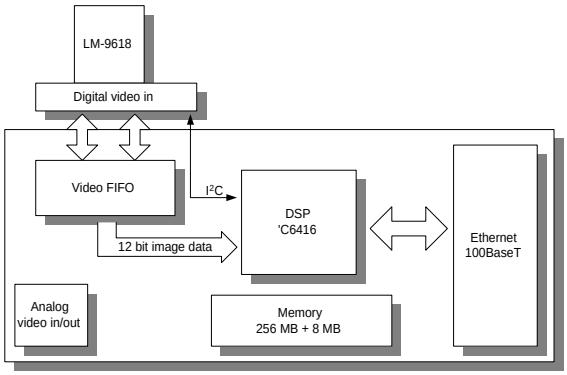

Figure 4: Testbed architecture of the smart camera.

4.4 Firmware

The smart camera's firmware controls the overall system behavior, and provides interfaces and methods for different tasks like task management, camera control, and control of peripherals.

Four basic runlevels are defined by the firmware: (i) The normal mode, (ii) the alarm mode, (iii) the full update mode, and finally (iv) the partial update mode.

In *Normal Mode*, the video compression is running with low quality, and therefore with low bandwidth requirements, while the video analysis tasks are running at full rate. The second mode, the *Alarm Mode*, is entered, if an alarm situation has been detected by the video analysis system. The quality of the video stream is increased and, if necessary, the video analysis tasks are throttled to gain more processing power for the network management. The calculation of the traffic parameters remains at full rate. The *Full-Update Mode* is used if the firmware is updated. Therefore, all analysis and compression tasks are halted. Finally, the *Partial-Update Mode* is activated each time a task has to be replaced or removed from the smart camera. The code for the new task is downloaded to the camera, however, all other tasks remain running.

5 Experiments

5.1 Prototype Architecture

In order to evaluate function and performance of our smart camera as soon as possible we developed a prototype architecture. This prototype is implemented as embedded system and includes all functional units of the final camera platform.

Heart of our prototype is the *Network Video Development Kit (NVDK)* from ATEME. The NVDK includes a TMS320C6416 DSP with 264 MB SDRAM as well as an Ethernet network extension card. We extended this prototype platform by an additional extension board that connects the LM-9618 CMOS sensor with the NVDK. Video data is transferred via FIFO memory.

The architecture of the prototype is very close to the final target platform of the smart camera. Only a single DSP is available at the prototype, but separate test runs deliver usable results concerning functionality and performance. Figure 5 presents a picture of our prototype.

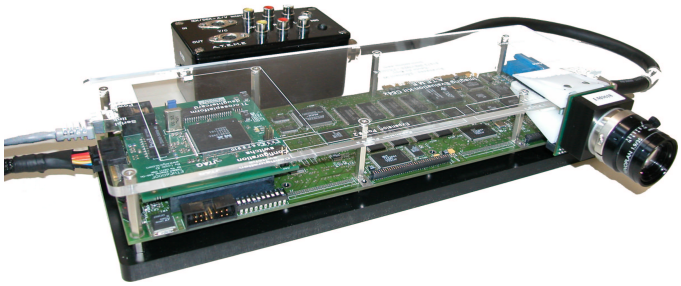

Figure 5: Prototype architecture of the smart camera including the CMOS image sensor, the DSP-based processing unit and the Ethernet network connection.

5.2 Video Analysis

Stationary vehicle detection as described in Section 4.1 was the first algorithm implemented on our prototype. After porting the original Matlab implementation to C++, we have done some performance experiments.

The unoptimized implementation required approximately 350 billion cycles per frame. This corresponds to a processing time of 7 seconds per image at full PAL resolution. The main bottlenecks of this implementation were excessive memory trashing and the intensive use of floating-point operations. Floating-point arithmetic is rather expensive on the fixed-point TMS320C64x DSP. Optimized floating-point runtime libraries improved performance, however, we decided to implement a fixed-point numerical format to represent fractional numbers in a 16 bit variable. This boosted the performance to 1 fps by reducing memory trashing and the implementation of the mentioned fixed-point number representation.

The stationary vehicle detection algorithm has been implemented by exploiting the compiler's class template support which made it easy to port the algorithm from Matlab to C++. However, the DSP's computing performance suffers significantly from this kind of implementation. A significant performance improvement is expected by a comprehensive re-implementation of the algorithm.

5.3 Power Estimation

The estimated power consumption of the smart camera is summarized in Table 2. It lists the *typical* power consumption of the main components of the camera. By applying DPM, a high power-saving potential can be exploited.

CMOS Image Sensor	2.470mW	RS232	950mW
DSPs	3.960mW	GSM Module	2.280mW
264MByte SDRAM	2.800mW	FIFO	990mW
Ethernet Interface	1.840mW	Other Components	1.500mW
		Total Power Consumption	16.790mW

Table 2: Estimated power consumption of the smart camera.

6 Conclusion

A smart camera realized as an embedded system has been presented in this paper. Our smart camera integrates a digital CMOS image sensor, a processing unit featuring two high-performance DSPs and a network interface and it clearly promotes the concept of Wolf et al. [3]. High-level video analysis algorithms in combination with state-of-the-art video compression transform this system from a network camera into a smart camera.

There is a rapidly increasing market for smart cameras. Advances in performance and integration will enable new and more functionality implemented in smart cameras. The next steps in our research project include (i) the development of the target architecture, (ii) the implementation of further image processing algorithms, and (iii) the real-world evaluation on highways.

References

[1] K. Borras. In life you must have vision. *Traffic Technology International*, pages 36–39, Oct/Nov 2002.

[2] C. S. Regazzoni, V. Ramesh, and G. L. Foresti. Introduction of the special issue. *Proceedings of the IEEE*, 89(10), Oct 2001.

[3] W. Wolf, B. Ozer, and T. Lv. Smart cameras as embedded systems. *IEEE Computer*, pages 48–53, 2002.

[4] Vision Components Germany. http://www.vision-comp.com, 2002.

[5] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Computer Vision Pattern Recognition*, 1997.

[6] D. R. Magee. Tracking multiple vehicles using foreground, background and motion models. Technical Report 2001.21, Univ. of Leeds, Dec 2001. School of Computer Studies Research Report Series.

[7] R. P. Pflugfelder. Visual traffic surveillance using real-time tracking. Master's thesis, Vienna University of Technology, Vienna, Austria, 2002.

[8] D. Gibbins, G. N. Newsam, and M. J. Brooks. Detecting suspicious background changes in video surveillance of busy scenes. In *Proceedings 3rd IEEE Workshop on Applications of Computer Vision*, pages 22–26, Sarasota, FL, USA, Dec 1996.

[9] S. Huwer and H. Niemann. Adaptive change detection for real-time surveillance applications. In *Third IEEE International Workshop on Visual Surveillance*, pages 37–45, Dublin, 2000. IEEE Computer Society, Los Alamitos.

[10] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision Pattern Recognition*, pages 246–252, Ft. Collins, CO, 1999.

[11] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *ICCV (1)*, pages 255–261, 1999.

[12] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on VLSI Systems*, 8:299–316, Jun 2000.

[13] R. Lyons. *Understanding Digital Signal Processing*. Addison-Wesley, 1997.

[14] R. P. Pflugfelder and H. Bischof. Learning spatiotemporal traffic behaviour and traffic patterns for unusual event detection. In *26th Workshop of the Austrian Association for Pattern Recognition*, pages 125–133.

[15] G. L. Foresti, P. Mähönen, and C. S. Regazzoni, editors. *Multimedia video-based surveillance systems*. Kluwer Academic Publishers, Boston, 2000.