TRICam - An Embedded Platform for Remote Traffic Surveillance

Clemens Arth, Horst Bischof
Graz University of Technology
Institute for Computer Graphics and Vision
Inffeldgasse 16/2, 8010 Graz, Austria
{arth,bischof}@icg.tu-graz.ac.at

Christian Leistner
FREQUENTIS GmbH
Technology Center GRAZ
Plueddemanngasse 104/1, 8042 Graz , Austria
christian.leistner@frequentis.com

Abstract

In this paper we present a novel embedded platform, dedicated especially to the surveillance of remote locations under harsh environmental conditions, featuring various video and audio compression algorithms as well as support for local systems and devices. The presented solution follows a radically decentralized approach and is able to act as an autonomous video server. Using up to three Texas Instruments™ TMS320C6414 DSPs, it is possible to use high-level computer vision algorithms in real-time in order to extract the information from the video stream which is relevant to the surveillance task.

The focus of this paper is on the task of vehicle detection and tracking in images. In particular, we discuss the issues specific for embedded systems, and we describe how they influenced our work. We give a detailed description of several algorithms and justify their use in our implementation. The power of our approach is shown on two real-world applications, namely vehicle detection on highways and license plate detection on urban traffic videos.

1. Introduction

In recent years the state of the art in traffic surveillance has changed dramatically. Many systems have changed from analogue closed circuit television (CCTV) cameras to fully digital systems. While this delivers higher flexibility and quality, as well as decreases the high communication costs for analogue systems, the growing amount of video data becomes more and more unmanageable for a single human operator.

Several arguments stand for the implementation of a totally decentralized, flexible embedded vision platform:

- Analogue-digital conversion and video compression should take place directly on the camera spot, to reduce communication overhead and to make wide-area surveillance monitoring feasible. Image processing algorithms can be used to take over particular tasks of a human operator, or to alert the operator.

- Facing threads of terrorism, there is a growing industry demand for embedded vision systems, which can meet the demands on fault tolerance and proper performance, even under adverse environmental conditions.

- Improvements of low-power high-performance DSPs for video processing in recent years made the application of even high-sophisticated computer vision algorithms on embedded platforms possible; thus, embedded platforms are representing an alternative solution to conventional industrial PCs nowadays.

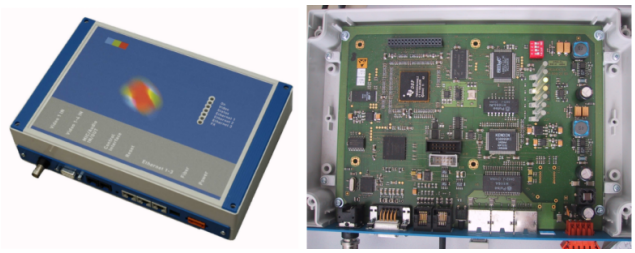

Figure 1. Prototype of the embedded DSP vision platform.

In this respect, our system has the following features: it is able to act as a stand alone video server, reduces the overall communication overhead, and has the objective to be applied in outdoor traffic scenes under harsh environmental conditions where standard industrial PCs are no longer applicable. As it is characteristic of embedded platforms, this robustness comes at the cost of very limited amount of memory and computational resources available for software applications. Moreover, floating point operations are very costly, which complicates algorithm design and makes additional usage of coding schemes necessary (for example floating point to fixed point conversions).

The goal of this paper is to implement a generic object detection and tracking system, which is able to perform in

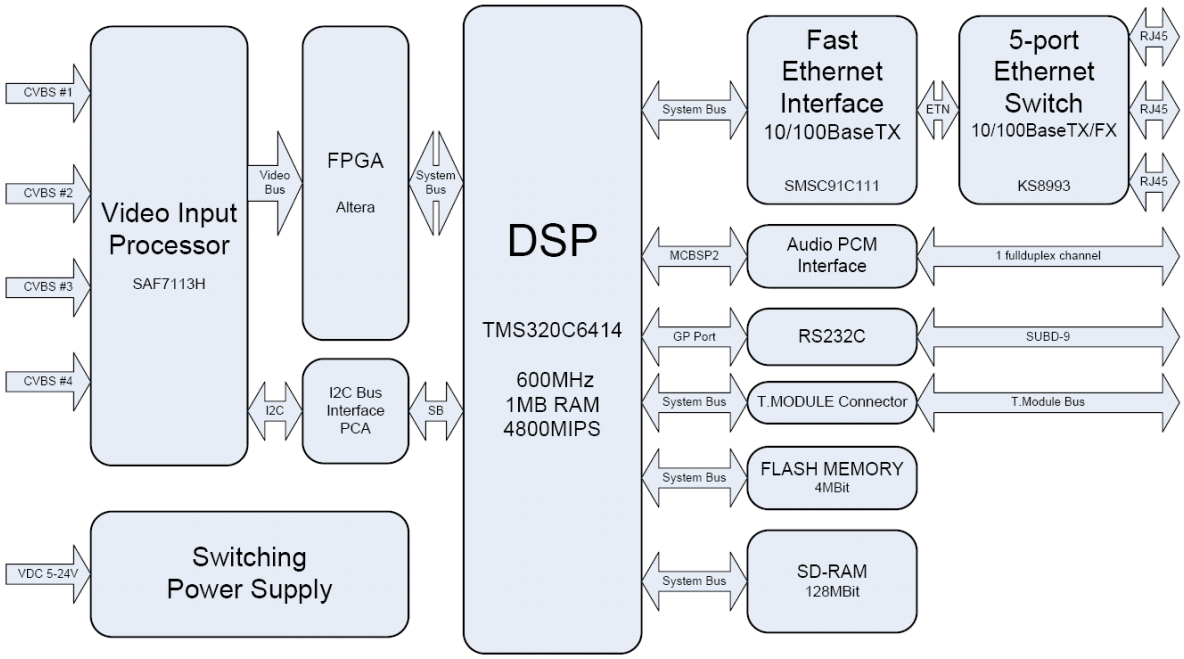

Figure 2. Block diagram of the hardware architecture.

real-time despite of these charges and restrictions. In the following section 2, we give a short overview about related work. A detailed description of our hardware platform is given in section 3. A description of the various software modules used in our system follows in section 4. In the experimental section 5, we show how the single modules are assembled in two real-world applications. The paper concludes in section 6, summarizing the main contributions of this work and giving an outlook on future work.

2. Related Work

Due to the lack of performance and memory ressources, visual object detection on embedded devices has not been used widely. Especially higher-level vision tasks are not treated in the embedded vision literature. Many work concentrates on special FPGA based architectures, their suitability for embedded vision systems and simple vision tasks, for example in the works of Sen *et al*. [17] and MacLean [11]. Yet, Mathew *et al*. presented a custom coprocessor which they call the *perception processor* to be used in mobile systems [12]. They compare several algorithms and show that their VLIW approach has significant advantages concerning performance and power to FPGAs, ASICs and CPU logic functions.

An embedded pedestrian detection system based on a clustered training set is presented by Shashua *et al*. [19]. Furthermore, they achieved a performance improvement from 10Hz to 25Hz frame rate by porting their algorithm from a standard 1GHz personal computer to a system-on-a-chip called ”EyeQ”.

An embedded vision system based on a CMOS imaging chip and a RISC processor able to perform real-time car counting is presented in the work of Chiu *et al*. [5]. The entire system consists of a network of 13 embedded vision systems and is applied to a parking facility.

Rowe *et al*. presented a low-cost embedded color vision system implemented in small mobile robots where traditional vision systems would not be practical [14]. They implemented basic color blob tracking and other low-level image processing tasks on a low-cost micro controller combined with a CMOS camera module and achieved 50 frames per second with a resolution of 352 x 288 pixels.

3. Hardware Architecture

The embedded DSP platform used in this paper can be qualified as being a small completely integrated video server, based on Texas Instruments^{TM} TMS320C6414 digital signal processor. The main features are:

- 1 TMS320C6414 DSP with 600 MHz and 1MB cache
- 128 MBit SDRAM for video compression, processing and storage of temporary data
- 4 MBit Flash Memory for firmware storage
- a 5-port ethernet switch with three 10/100 BaseTX ports and one 100 BaseFx fibre port

- 1 video input processor, featuring 4 analogue CVBS or 2 SVHS TDM video channels per system (PAL/NTSC)
- 1 FPGA for buffering video frames/scanlines between the video input processor and the DSP
- a full-duplex PCM audio interface using 8 kHz for voice communication and sound processing
- 2 RS232 serial ports
- a T-module expansion connector for two additional DSPs

The overall platform design is depicted in the block diagram in Figure 2.

The use of an analogue video interface allows for fast integration of the video server into an already existing network of pre-assembled cameras. However, this does not narrow the applicability of the platform in the domain of digital cameras, since the integrated ethernet interface can also be used to connect the platform to digital cameras sharing an ethernet port[1]. Furthermore, the audio interface can be used to augment the visual perception of any camera by audio information.

The computational resources of the platform may be scaled using the expansion module and additional DSPs, allowing for the application of more demanding algorithms from the field of computer vision. Any information, either generated or extracted, can be transmitted to a given destination in various ways, like sharing a conventional computer network or any other technology (modems or mobile phones may be interfaced using the integrated serial ports). Thereby, the amount of information to be transmitted may directly depend on the infrastructural resources available at site, thus, ranging from a few bytes for a simple text message to a fully encoded MPEG4 video stream [23] at 1.5 MBit/s.

In summary, the platform can be used in a lot of different ways, for example, as a simple compression device producing high-quality video for general surveillance tasks, or as a complete video telephone box using Voice over IP (VoIP). As a proof of concept, the applicability of our embedded platform as a smart vision device, utilizing various high-level computer vision algorithms, will be described in the following section.

4. Software Architecture

Due to the modular software design and architecture, the software packages can be composed of several single modules, depending on the task to be accomplished. We will describe these modules in the following and demonstrate how they can be assembled to fulfill their task in a real-world application. Furthermore, we justify their application and note special implementation issues.

[1] We are looking forward to integrate a USB 2.0 or Firewire interface in a further revision of the hardware platform.

4.1. Image Plane to Ground Plane Mapping

Because we are dealing with a stationary camera setup, it makes sense to bring in information about the scene geometry. This knowledge is used for narrowing the search space of our high-level object detector (section 4.2.2), and is used for estimating the average speed of the detected objects.

There are many possible algorithms known for camera calibration, also ones especially adapted to our problem of stationary traffic surveillance cameras (for example, refer to the work of Dailey *et al*. [16, 3]). Yet, we have not developed a camera calibration tool for our purposes, mainly due to the lack of time needed for a full-featured embedded implementation. Anyway, the information needed for the calculation of a simple plane-to-plane mapping, called a *Homography*, can easily be derived from the knowledge of road geometry and road painting measurements. Hence we preferred this solution over the camera calibration method.

At least four non-colinear point correspondences are needed to guarantee that the system of equations to be solved for the transformation between image-plane and real-world-plane is well-posed. A good explanation of the algorithm can be found in the book of Hartley and Zisserman [8]. Naturally, neither most objects are flat at all, nor the real-world ground is perfectly planar, thereby a lot of errors are introduced. Nevertheless, this solution performs sufficiently well for the purposes already mentioned above.

As the homography only has to be calculated once at startup of the system, it is done using *floating point* variables. The mapping of a single image point into real world coordinates reduces to a single matrix-vector calculation.

4.2. Generic Object Detection Algorithms

An important group of algorithms in computer vision is dealing with the detection of objects or humans, commonly known as the *generic object detection task*. As this task can be arbitrarily complicated, depending on the appearance of the objects to detect and the environment they are situated in, the algorithms used can be arbitrarily complex as well.

We discuss two algorithms for generic object detection in the following and list their advantages and shortcomings.

4.2.1 Background Modelling and Subtraction

Background modelling and subtraction is a very popular approach for motion detection and is our first algorithm to be described. The *Approximated Median Filter*, originally proposed by McFarlane and Schofield, is used for background modelling [13].

For each pixel $p[x, y]$, the median of a sequence of values is approximated by consecutively incrementing and decrementing the estimator $m[x, y]$. The estimator is decremented by one, if the input pixel value is smaller than the estimate, and incremented by one, if it is greater,

$$m_c[x,y] = \begin{cases} m_c[x,y]-1 & \text{if} \quad v_c[x,y] < m_c[x,y] \\ m_c[x,y]+1 & \text{if} \quad v_c[x,y] > m_c[x,y] \end{cases} \tag{1}$$

with $c \in \{R, G, B\}$. Because we are using 24-bit RGB-images, we apply the background modelling approach to all three color channels and threshold over the sum of the three differences to create our difference image. This makes the model a little more robust, introducing only little additional cost. Image subtraction of the background model from a new frame and thresholding over the difference gives a binary image with white blobs representing areas of motion in the actual frame.

$$\text{diffimage}[x,y] = \begin{cases} 1 & \text{if} \quad \text{sum} > \text{Threshold} \\ 0 & \text{else} \end{cases} \tag{2}$$

$$\text{sum} = \sum_{c \in \{R,G,B\}} abs(v_c[x,y] - m_c[x,y]) \tag{3}$$

After morphological opening and closing operations, applied on the difference image, it is passed to the second module, where the connected white blobs are labeled and their dimensions and positions are recorded using a standard region labelling algorithm [22].

The approximated median filter is very well suited for implementation on DSPs, because its computational costs are relatively low and it simultaneously provides stunningly good performance (an overview of various algorithms for background modelling and their properties is given in the work of Cheung and Kamath [4]). Furthermore an implementation does not require any floating point operations and only a moderate amount of memory resources.

The described algorithm does perform reasonably well, even under inclement weather conditions like heavy rain or snow. The majority of the detected regions are more or less correctly corresponding to real objects. Nevertheless, errors occur due to shadows or occlusions, and the algorithm is sensitive to camera motion. Another drawback is, that that objects, which become stationary, tend to ”fade” into the background model.

Nevertheless, for most scenarios this approach is sufficient for extracting regions of interest where moving objects are located. This approach is also used to collect training examples for the algorithm described in the subsequent section 4.2.2.

4.2.2 Viola-Jones Detector

This popular detection algorithm is mainly based on the AdaBoost approach presented by Viola and Jones [24]. Although, meanwhile a large number of new boosting algorithms has emerged, most of them are slight variations of the classical Viola-Jones approach such as WaldBoost [21] or WeightBoost [9], for embedded applications still the original approach has turned out to be the best choice.

Viola and Jones achieved real-time performance as well as high accuracy by using a boosted cascade of weak classifiers or perceptrons which enabled early rejection of simple samples and by representing images as summed tables or integral images which allowed to calculate simple linear Haar-features in constant time. Additionally, the entire classifier can be implemented on a platform with only fixed point numbers. After a new stage has been built, all negative samples are filtered and only wrong classified samples are kept which leads to increasing classifier stage complexity. The tedious training process can be performed on a standard PC and the trained classifier is loaded onto the embedded platform where the image is scanned with a search window in different scales and locations.

In our current implementation, the integral images are calculated in *integer* units and thresholding inside the weak classifiers is done using *floats*. On this account, the number of weak classifiers and the shape of the cascades has significant influence to system performance on our embedded platform. Thus, it is crucial to minimize the number of classifiers simultaneously keeping a good level of performance.

We acchieved superior system accuracy over the discrete boosting approach by using real-valued or confidence-rated predictions as proposed by Shapire and Singer [18]. In practice, this has the advantage that during training the system converges faster than discrete boosting if the training process is stopped after a few iterations. Moreover, we reduced the overall number of weak classifiers in the long term, while there is no large accuracy difference to the discrete approach of Viola and Jones.

We, additionally, applied the ideas of Sochman and Matas [20] to cascade building. In Viola and Jones' cascade training process all features are thrown away after a stage has been built which totally ignores the additional information on confidence provided by the stage output. Each new stage is trained from scratch and independently from its predecessor. This has turned out to be sometimes a waste of already well performing features.

In contrast, Sochman and Matas proposed a cascade method where previous-stage weak classifiers propagate from one stage into the next. This works at zero evaluation cost because the propagated classifiers have already been evaluated in the foregoing stage but leads to stronger stages due to the additional information. We applied this inter-stage feature propagation method to RealBoost which

led to significant fewer weak classifers and, hence, faster detectors while keeping almost the same accuracy as in the original approach.

Furthermore, we achieved a significant reduction of weak classifiers by carefully selecting the negative training samples. Hence, we randomly selected background traffic scences rather than arbitrary examples. The training of the detector still has to be done on a usual desktop computer, because of the memory resources needed by the training algorithm. We use the approach described in the previous section 4.2.1 to accelerate the process of acquiring positive training samples, and we use a method similar to the one presented in [1].

4.3. Post-processing Methods

The detector described in the previous section 4.2.2, as well as many other object detection algorithms, uses exhaustive search techniques. Thus, a subwindow scans the entire image in different scales and locations and evaluates its content. This search method, however, owing to the invariance of the classifier to small translations of an object, has the side-effect that the same object is detected more than once. These multiple detections then have to be post-processed to form up one single correct detection.

Generally, there exist various simple approaches to solve this problem, most of them simply combine overlapping bounding boxes as in [24] and [15], where the latter one uses the number of detections in a small environment as a measure for confidence.

If detections are able to deliver additional information such as values or margins for safety or probability, more intelligent algorithms can be used. One such method is the *non-maximum suppression* where all detections are sorted by their safety values or filter responses in descending order. If two or more boxes overlap to a certain extent, the weaker detection is eliminated.

Additionally, when having confidence values, it is also possible to calculate a *classifier activation map* where all positive results are weighted by their confidence responses. While this matrix can then be analyzed in various ways we use a variant of the MeanShift algorithm [6], the CAMShift as proposed by Bradski [2] originally used for face tracking. An application to post-processing can be found in [7].

The Mean Shift algorithm requires that all acquired detection responses are represented as probability density function estimations which is calculated as

$$\hat{f}_k(x) = \sum_{i=k}^{n} Y_k(X_i) K_k\Big(\frac{x - X_i}{W_k}\Big), \tag{4}$$

where $\{\mathcal{X}_i\}_{1\dots n}$ are the image locations where classification has been performed and $K_k(\cdot)$ is the two-dimensional *Gaussian* kernel with a size equivalent to the object size W_k. The Mean Shift then climbs the gradients of this distribution to find dominant clusters (modes or peaks) which is also known as mode seeking.

Unlike to the fixed window size of the Mean Shift, the CAMShift (**C**ontinuously **A**daptive **M**ean **Shift**) adaptively adjusts the window size in order to find proper modes. The starting points of the algorithm are the local maxima determined from the probability distribution function D. In each CAMShift iteration the initially very small window size (e.g. 20% of the original detection window size) and/or the position of the current window are adjusted. The local covariance within the window is estimated by computing local statistical moments of the zeroth- (M_{00}), first- (M_{10}, M_{01}) and second-order (M_{20}, M_{02}, M_{22}) as shown in equation 5 which can be performed very efficiently by representing these modes as *integral images* [7]. The width and height of the new window size are represented as elliptic estimations of the underlying probability distribution. Equations 6 and 7 show how the new adapted size is calculated.

$$\begin{aligned} M_{00} &= \textstyle\sum_x \sum_y D(x,y) \\ M_{01} &= \textstyle\sum_x \sum_y y \cdot D(x,y) \\ M_{10} &= \textstyle\sum_x \sum_y x \cdot D(x,y) \\ M_{02} &= \textstyle\sum_x \sum_y y^2 \cdot D(x,y) \\ M_{20} &= \textstyle\sum_x \sum_y x^2 \cdot D(x,y) \\ M_{22} &= \textstyle\sum_x \sum_y x \cdot y \cdot D(x,y) \end{aligned} \tag{5}$$

$$\begin{aligned} a &= \tfrac{M_{20}}{M_{00}} - x'^2 \quad b = 2\Big(\tfrac{M_{11}}{M_{00}} - x'y'\Big) \\ c &= \tfrac{M_{02}}{M_{00}} - y'^2 \end{aligned} \tag{6}$$

$$\begin{aligned} width &= \sqrt{\tfrac{(a+c)+\sqrt{b^2+(a-c^2)}}{2}} \\ height &= \sqrt{\tfrac{(a+c)-\sqrt{b^2+(a-c^2)}}{2}} \end{aligned} \tag{7}$$

4.4. Tracking

Tracking algorithms are used to bring loose observations, taken from single frames, together into connected sequences - in our case, this means, following an object in motion. For that purpose, the well-known Kalman Filter is a good choice ([10]). The movement of an object in x and y direction is modelled separately by a second order equation of motion. A list of fixed length is used to store the state information for a whole tracking sequence, from which vehicle speed information is deduced later on.

At the moment we have only an implementation using *floating point* variables, therefore only a limited number of objects can be tracked simultaneously. As the algorithm involves a lot of matrix multiplications and matrix inversions, tracking a lot of objects gets computationally expensive. We leave a portation of our implementation to a fixed-point version as an open issue.

5. Experiments

Two applications are presented which demonstrate the power of our approach, both from the area of traffic surveillance. Especially, we try to point out the real-time capabilities of our system and the universal applicability of the hardware platform in combination with selected algorithms. Furthermore, we should mention, that we have put absolutely no effort into optimization of our algorithms. The code used for the experiments mostly conforms to ANSI C standard without any special usage of intrinsics or linear assembly code. Thus, the time measurements given could definitely be improved by introducing these types of code optimization techniques.

5.1. Single Module Time Consumption

In table 5 we list the average time consumption of our module implementations using special parameters. We apply our algorithms on a full CIF frame and a special subregion of the frame, depending on the application.

For the background modelling and subtraction module, we use a threshold of 45, which gives good results for our scenarios. The time for processing is constant because every pixel has to be processed twice (update and subtraction), irrespective of its content.

The region labelling process does not require any parameters to be set. The labelling process passes the image a first time maintaining a list of blob correspondences. Because this list has to be taken into account in a second pass, the length of this list slightly influences the average processing time of the algorithm.

The Viola-Jones detector used for vehicle detection uses only 27 features and 7 stages. To speed up the search process, the exhaustive search is limited to three scales using the scene geometry information.

For license plate detection, the search space is narrowed to 5 scales, while the detection algorithm used only requires 37 features in 8 stages.

Due to the simplicity of the non-maximum suppression, the number of detections to be processed does only slightly influence its time consumption. On average, the time needed for post-processing is negligible.

The computational complexity of the implementation of the CamShift algorithm does not make it suitable for real-time processing. Because the number of iterations necessary for convergence are not known beforehand, the average time needed for processing is quite too long. Anyway, using an optimized version of this algorithm is a clear alternative to using the non-maximum suppression algorithm, especially for complex scenarios, as shown in the computer vision literature [7].

As can easily be seen from the last two rows of table 5, the time consumption of the a single Kalman tracker is significant and should be further improved to make tracking of multiple objects feasible. In contrast, the time needed for mapping an image point to a real-world point using a homography is negligible.

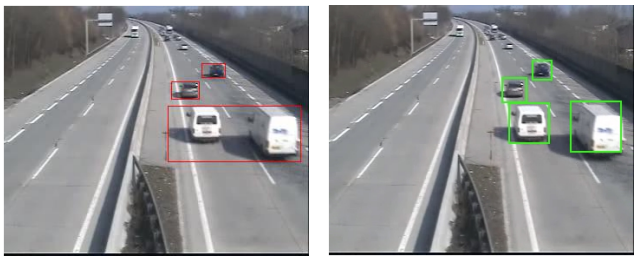

Figure 3. A sample frame taken from a 30 minutes demonstration video and the results of both algorithms. The simple algorithm is not capable of detecting both vehicles separately (left). The enhanced algorithm can easily overcome this problem and is able to correctly separate the vehicle detections (right).

Figure 4. A frame taken from a different video stream and the results of both algorithms. The simple system simply fails because objects are “fading” into the background model (left). The upgraded system can still detect all objects correctly because the algorithm simply scans over the whole image if no region of interest has been found by the previous module.

5.2. Simple Vehicle Detection System

For our first system, we use the background modelling module described in section 4.2.1 and the region labelling algorithm for object detection. Referring to table 5, the average time needed for the algorithm applied on a region of interest is calculated to about $2.27ms$.

As can easily be seen in Figure 3 on the left, this approach is not capable of detecting two vehicles separately if shadows are merging them into a single blob in the difference image. The performance of the system is also unsatisfactory if objects have to stop for some reason, for example due to a jamming situation (see Figure 4).

Even if there are a lot of problems, we can still use this setup to gather images for the training stage of our Viola-Jones detector. We carefully select good detections and add them to our positive training set.

Module	Full Frame 352x288		Region Of Interest 192x220	
	avg. time [ms]	std. dev.	avg. time [ms]	std. dev.
Background Modelling and Subtraction	1.768	0.0	0.75	0.0
Region Labelling	2.7	0.87	1.52	0.68
Viola-Jones Detector (5 scales) (Vehicles)	92.38	10.58	44.14	5.67
Viola-Jones Detector (3 scales) (License Plates)	141.62	3.27	-	-
Non-Maximum Suppression (max 150 detections)	0.18	0.16	0.034	0.041
CamShift (max 40 iterations)	208.53	222.43	139.7	93.86
	Single Execution			
KalmanTracker (Estimate/Update)	0.346	0.01		
Homography	0.008	0.0		

Table 1. Average time taken to execute each module. For simplicity, we have listed all time measurements for all experiments in this single table.

5.3. Enhanced Vehicle Detection System

To overcome the shortcomings of the system described in the previous section, we introduce the Viola-Jones detector module, already explained in section 4.2.2.

The simple system is kept as it is and is now used to pre-select regions of interest in the actual frame. On all regions found, the *Viola-Jones detector* is applied to verify the presence or absence of a vehicle. Alternatively, if there were no blobs found, the detector is applied on the whole frame to correct for errors, which have occurred due to the shortcomings of the background modelling approach. To correct for multiple overlapping detections of the same vehicle, the *Non-maximum suppression* algorithm is applied (section 4.3).

In Figures 3 and 4, the results of our extension are shown. While the problems of our previous approach can be overcome using this new module, this comes at additional computational costs. The worst case time consumption now rises to about $46.6ms$, if the detector is applied to the whole region of interest and non-maximum suppression is used as post-processing method. Note that this is only the case if the detection process by the overlying simple module has failed. The time consumption for verification of a blob is negligible, thus the average time needed for processing a sequence of frames is much less than $40ms$ per frame.

5.4. Complete Framework

For our final system, we additionally integrate the Kalman tracker module (section 4.4) and the homography module (section 4.1) into our framework. After the detection algorithm has been applied, for all verified vehicles a single tracker is initialized to record its movements in the following frames. Using the homography module, we can estimate the velocity of the detected vehicles and the lane used. Therefore we simply calculate the spatial distance between the start and the end of the tracking sequence in the real world and divide it by the temporal period needed to cover this distance.

Figure 5. Two frames taken from a 30 minutes demonstration video and the results of our algorithm. As can easily be seen, in both frames, both vehicles were successfully tracked. The corners of the yellow rectangle represent the four reference points used for homography calculation.

Figure 5 shows the tracking results on two frames, taken from a 30 minutes demonstration video. The estimated speed of the vehicles is also noted. Unfortunately there are no reference measurements available at this site. Anyway, reference measurements at another site using laser speed sensors have shown that our approach delivers plausible values, and that only a constant offset error might occur due to erroneous reference point measurements in the homography calculation.

5.5. Licence Plate Detection

To justify the application of the Viola-Jones detector, we apply it on the very popular task of License Plate Detection. Because we want to detect a subregion of an object in motion, a simple motion detection algorithm is not sufficient to accomplish this task. Thus the usage of the Viola-Jones detector is justified. Its application is shown in searching licence plates on cars in a urban traffic surveillance video. Our intention is to further perform OCR (Optical Charac-

Figure 6. Two frames taken from a 15 minutes demonstration video and the results our algorithm. Note, that the region inside the bounding box should now be passed to an OCR module for further processing.

#plates	#plates detected	dr	#fp
100	96	96%	28

Table 2. Overall detection results for 100 license plates in a 15 minute demonstration video.

ter Recognition) on the license plates detected, thus there are some minimum requirements for the size of the detections. The description of the algorithms used for OCR is out of this paper's scope, but thereby this experimental setup is motivated.

The detection algorithm was applied to a full 352x288 frame using a classifier already explained in 5.1. Due to the high accuracy of the detector, a simple non-maximum suppression algorithm (section 4.3) served sufficiently for post-processing. Referring to the time consumption table 5, the average time for processing a single frame is about $140ms$. The detection results are listed in table 5.5.

Because the scale search process was not perfectly tuned to the experimental setup, the time taken by the detection algorithm is much longer than in the previous example. Furthermore, the additional amount of features used intensifies this effect. Using a loopback setup of tracking and detection, the amount of time necessary could be reduced significantly. Up to this time, this was left for implementation in a complete license plate recognition framework.

6. Conclusion and Future Work

In this work we presented an embedded platform capable of performing high-level computer vision tasks such as vehicle and license plate detection in real-time. We have shown that it is possible to build a flexible and robust embedded vision system to be applied on a wide area of applications by using a modular and flexible software design for embedded devices.

In future work, we will concentrate on further optimizing both the hardware and software architecture. We believe that it is possible to additional gain performance and accuracy by especially concentrate on further algorithm quality improvement and a better mapping to the underlying DSP architecture. For that purpose we will introduce the usage of both intrinsics and linear assembly code. We believe that this might further speed up our algorithms by up to a factor of 20 for some parts of our code.

We are looking forward to integrate a traffic jam detection module into our framework to add the functionality of traffic jam detection and operator alertion. Additionally to traffic surveillance, we will generalize our approach to other fields of out-door object detection tasks, such as pedestrian detection on public places.

References

[1] Y. Abramson and Y. Freund. SEVILLE, Semi-automatic Visual Learning. In *International Conference on Pattern Recognition*, 2005. 5

[2] G. R. Bradski. Computer Vision Face Tracking For Use In Perceptual User Interface. In *Intel Technology Journal*, pages 123–140, 1998. 5

[3] F. W. Cathey and D. J. Dailey. A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras. In *IEEE Intelligent Vehicles Symposium*, pages 777–782, 2005. 3

[4] S. S. Cheung and C. Kamath. Robust Techniques for Background Subtraction in Urban Traffic Video. *Video Communications and Image Processing, SPIE Electronic Imaging, San Jose*, January 2004. 4

[5] M.-Y. Chiu, R. Depommier, and T. Spindler. An Embedded Real-Time Vision System For 24-Hour Indoor/Outdoor Car Counting Applications. In *International Conference on Pattern Recognition*, pages 338–341, 2004. 2

[6] D. Comaniciou and P. Meer. Mean Shift Analysis and Applications. In *International Conference on Computer Vision and Pattern Recognition*, pages 1194–1233, 1999. 5

[7] H. Grabner, C. Belezani, and H. Bischof. Improving Adaboost detection rate by Wobble and Mean Shift. In *Proceedings of Computer Vision Winter Workshop,*, 2005. 5, 6

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 3

[9] R. Jin, Y. Liu, L. Si, J. Carbonell, and A. G. Hauptmann. A New Boosting Algorithm Using Input-dependent Regularizer. In *International Conference on Machine Learning*, 2003. 4

[10] R. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME–Journal of Basic Engineering*, pages 35–45, 1960. 5

[11] J. W. MacLean. An Evaluation of the Suitability of FPGAs for Embedded Vision Systems. In *Embedded Computer Vision Workshop, International Conference on Computer Vision and Pattern Recognition*, pages 131–138, 2005. 2

[12] B. Mathew, A. Davis, and M. Parker. A Low Power Architecture for Embedded Perception. In *International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 46–56, 2004. 2

[13] N. McFarlane and C. Schofield. Segmentation and Tracking of Piglets in Images. *Machine Vision and Applications*, 8(3):187–193, 1995. 3

[14] A. Rowe, C. Rosenberg, and I. Nourbakhsh. A Second Generation Low Cost Embedded Color Vision System. In *Embedded Computer Vision Workshop, International Conference on Computer Vision and Pattern Recognition*, 2005. 2

[15] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *IEEE Transactions on Pattern Recognition and Machine Intelligence*, pages 23–38, 1998. 5

[16] T. Schoepflin and D. Dailey. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):90–98, June 2003. 3

[17] M. Sen, I. Corretjer, F. H. S. Saha, J. Schlessman, S. S. Bhattacharyya, and W. Wolf. Computer Vision on FPGAs: Design Methodology and its Application to Gesture Recognition. In *Embedded Computer Vision Workshop, International Conference on Computer Vision and Pattern Recognition*, pages 133–141, 2005. 2

[18] R. E. Shapire and Y. Singer. Improved Boosting Algorithms Using Confidence Rated Predictions. In *Machine Learning*, pages 297–336, 1999. 4

[19] A. Shashua, Y. Gdalyahu, and G. Hayun. Pedestrian Detection for Driving Assistence Systems: Single-Frame Classification and System Level Performance. In *IEEE Intelligent Vehicles Symposium*, pages 1–6, 2004. 2

[20] J. Sochman and J. Matas. Inter-stage Feature Propagation In Cascade Building With AdaBoost. In *International Conference on Pattern Recognition*, pages 236–239, 2004. 4

[21] J. Sochman and J. Matas. WaldBoost - Learning For Time Constrained Sequential Detection. In *International Conference on Pattern Recognition*, pages 150–156, 2005. 4

[22] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision, Second Edition*. PWS Publishing, 1998. 4

[23] `Moving Picture Experts Group`. http://www.chiariglione.org/mpeg/index.htm. 3

[24] P. A. Viola and M. J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *International Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001. 4, 5