Hardware Approach for Real Time Machine Stereo Vision

M. Tornow, J. Kaszubiak,
R.W. Kuhn, B. Michaelis and T. Schindler
Institute for Electronics, Signal Processing
and Communications
Magdeburg University, Germany
Email: michael.tornow@e-technik.uni-magdeburg.de

ABSTRACT

Image processing is an effective tool for the analysis of optical sensor information for driver assistance systems and controlling of autonomous robots. Algorithms for image processing are often very complex and costly in terms of computation. In robotics and driver assistance systems, real-time processing is necessary. Signal processing algorithms must often be drastically modified so they can be implemented in the hardware. This task is especially difficult for continuous real-time processing at high speeds. This article describes a hardware-software co-design for a multi-object position sensor based on a stereophotogrammetric measuring method. In order to cover a large measuring area, an optimized algorithm based on an image pyramid is implemented in an FPGA as a parallel hardware solution for depth map calculation. Object recognition and tracking are then executed in real-time in a processor with help of software. For this task a statistical cluster method is used. Stabilization of the tracking is realized through use of a Kalman filter.

Keywords: stereophotogrammetry, hardware-software co-design, FPGA, 3-d image analysis, real-time, clustering and tracking.

1. INTRODUCTION

Environmental sensing is important for autonomous robots and vehicles interaction with their surroundings. There are a multitude of sensor techniques that are available for environment sensing, such as laser scanning [10], radar, and ultrasound [23] etc., which can be implemented in combinations to balance their respective weaknesses. The majority of these techniques is based on active processes.

Photogrammetry is a passive position measurement technique, in which images from several cameras are analyzed. In time-critical applications like driver assistance systems [5], [13], or autonomic robots, real-time image processing systems with constant and known delay times are needed. This is not achievable using software based on current computational techniques and running at current processors. Therefore the hardware implementation of the image analysis is more suitable.

This work presents a robust, real-time 3-d-object recognition, measurement and tracking system which uses a continuous data stream. Therefore, a stereo-camera system with a measurement range between 10 and 100 m is used. A depth map is determined from the stereo image, whose data is then fed to a cluster algorithm. The distance is subsequently given over to a Kalman filter [16], which calculates the velocity and enables further object tracking. Because of the requirements for real-time processing and perspective realization as a micro-system, the algorithm was implemented in an embedded hardware, which consists of a FPGA [6] and a processor which can be embedded in the FPGA [2] as well.

Several application examples for such an image processing system were analyzed. The long-term objective is sensing all surrounding information of a vehicle or a robot. The main objective at present is the obstacle detection for front and rear view. In our case we have application examples in automation of robots for front view and a lane change driver assistance system for rear view. For this paper the lane change assistant is used as example and for determining the boundary conditions.

In the next section we discuss the specifications of the different components.

2. HARDWARE-SOFTWARE CO-DESIGN

Boundary Conditions

This driver assistant system is used to detect dangerous situations on highways or cities while changing lanes. Therefore all objects in the rear of a car are identified, the position has to be determined and the objects should be sorted by lanes.

	covered distance [m]		
time [ms]	at 50 km/h	at 100 km/h	at 250 km/h
1000	13,89	27,78	69,44
500	6,94	13,89	34,72
100	1,39	2,78	6,94
40	0,56	1,11	2,78
20	0,28	0,56	1,39

Tab. 1. Real Time Conditions: Covered Way at Different Speeds

These tasks have to be achieved in real time. Technical processes are analyzed and worst case scenarios are identified to determine the real time conditions. But a car driven by a person is no technical process. A part from all delay times of the systems in the car is the driver. This one can not be measured exactly, due to several circumstances. The main aim of this project is to give him as much time as possible for his reaction. A person needs about a 1-2 seconds [8] for a reaction in a worst case . Under this premise a worst case scenario can be identified as well.

The worst case is, if a car is entering a highway and needs to stop and a second car is driving on the highway with a very high speed. This means for example this car appears with a speed from 100 to 250 km/h in the rear of the stopped car.

From the covered way of a car at different speeds shown in table 1 we can derive the real time conditions and other boundary conditions for the purposed system.

In order to save a reaction time of a second for the driving person the system has to detect the car at least at a distance of 100 m. Therefore a camera system with a high measuring range is needed. Furthermore high resolution cameras need to be used. In the section 3 this point is dicussed.

For realizing a close tracking of the obstacles the sampling rate should be rather high. For the worst case at least every 3 m the position of the second car needs to be determined in order to cover outliers. Considering table 1 at least every 40 ms an image pair needs to be taken respectively the frame rate which is the sampling rate for the position measurement is at least 25 Hz.

Hardware vs. Software

The used cameras are working with a frame rate of 25 Hz. The data stream produced by the two cameras is 32.7 Mbyte/s, due to the resolution of 1024x1024 Pixel with 10bit gray values each. For data processing with a microprocessor it is necessary to reduce this high data rate, thus a hardware-software co-design [12] is a good choice. Therefore, the algorithm must be separated in a hardware and a software part.

Principally, Software algorithms can be implemented in hardware and vice versa. The decision on which basic platform the algorithm should be implemented, depends on the needed processing speed and the implementation costs for the target system [11] (Fig. 1).

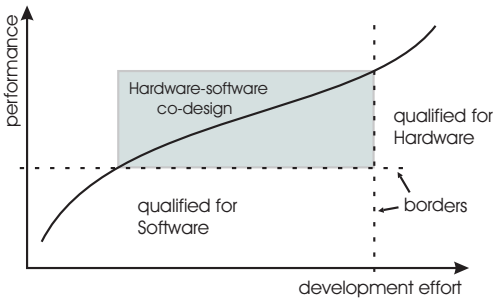

Fig. 1. Costs/Performance-Graph (modified after [11])

The working scheme of a microprocessor is ”computing in time” (Fig. 2.a). That means the algorithms process data sequentially on a single processing element. The next algorithm can be executed only after finishing the preceding algorithm. Thereby, the processing time raises with the number of algorithms because of the sequential working scheme. An algorithm with various branches in the data flow is called ”control flow oriented” and is well suited for microprocessor implementation, due to the need of high flexibility in the data stream.

A ”data flow oriented” algorithm is more suited for a hardware implementation, due to the fact the data flow consists of few branches but many arithmetic operations [19]. During processing no changes of the flowchart is necessary. The benefit of the hardware implementation is the possible segmentation of the algorithm and parallel processing on different processing elements. This leads to a so called ”computing in space” (Fig. 2.b). If one segment needs the results of another segment a ”pipeline structure” (Fig. 2.c) is more suitable. Pipelining does

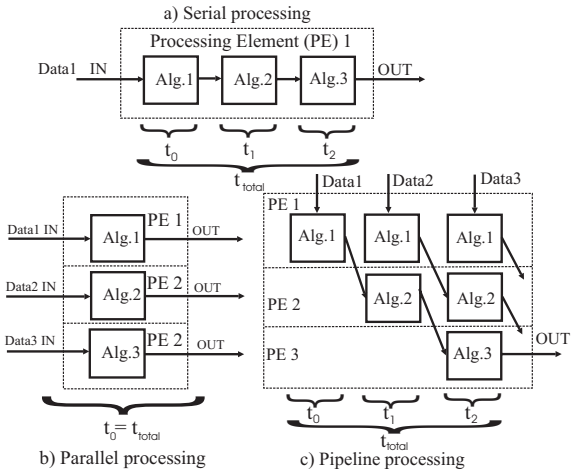

Fig. 2. Processing structures

not increase the execution time but the data throughput. Data flow oriented algorithms have the ability to work with a large amount of data with simple arithmetic operations at high speed. With a high degree of parallelization a big saving of processing time is possible. Whereby the logic costs increase fast with the logic complexity. If ”control flow oriented” algorithms are implemented in hardware, the logic costs rise significantly compared to a possible speed benefit, because of the low flexibility of the hardware.

Hardware Platform

The complete algorithm is realized as an embedded system. In Figure 3 the developed universal purpose board is shown. The power consumption depends on the running application but is 10 W in maximum. An application specific board would be an one chip solution with less power consumption.

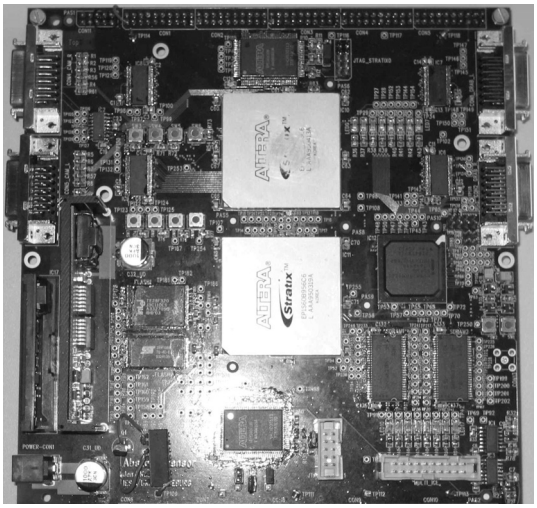

Fig. 3. Embedded board

We used two $ALTERA\ STRATIX\ EP1S60$ FPGAs [3] on the development board with added debug functionality. The FPGAs have 60’000 logic cells (LC) and 500 kByte of memory, each. Furthermore 16 MByte of SDRAM is installed on board. As processors we used $NIOS\ II$ softcore IP-processors. The

processor IP-Block can be implemented directly on the FPGA. The maximum speed is 75 MHz on $STRATIX$ FPGAs.

Due to the high flexibility of the FPGAs and the use of softcore processors it is possible to adapt the hardware to a huge amount of applications. The logic cell (LC) consumption of one processor is $1'500$ LC. Thus an implementation of a number of processors is possible. The number of processors depends on the algorithm itself, as well as the use of logic. The limitation of logic and processors is the number of logic cells.

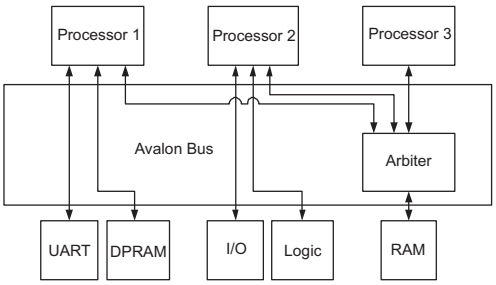

Fig. 4. The avalon bus architecture (modified after [2])

For independent program execution each NIOS II is connected to its own internal RAM. The processors and the logic are connected by the $AvalonBridge$ (Fig. 4). Whereby the connections are very flexible. They depend on the needs of the running application. Due to the independent bus connections within the Avalon Bus a high data bandwidth can be achieved.

HW-SW Co-Design

Image processing algorithms are mostly divided into a slice model (Fig. 5).The main steps are

- image acquisition
- preprocessing
- feature extraction
- classification & interpretation

The image acquisition is done by two cameras. They produce a continuous data stream processed by the following algorithms.

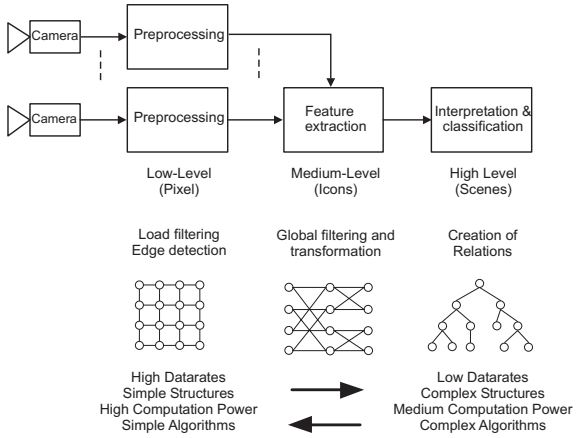

Fig. 5. Slice model of image processing

The data dependency within the individual processing steps changes from simple to complex. The preprocessing consists of local operators with independent data processing and a very high data rate. With their low data dependency these algorithms are data flow oriented. Typical algorithms are edge detections or LUTs. More complex functions like the KKFMF (see section 3) act as a local operator as well. The feature extraction is applied globally for the whole image. The data dependency is more complex and the data rate is high as well. The processing scheme can be data flow oriented or control flow oriented. The generation of histograms is a typical application.

The most complex level is the classification & interpretation. These applications are control flow oriented and can proceed the preprocessed data of more than one image. The data rate is comparably low but a lot of branches are applied. The clustering and Kalman-filtering belongs to this level.

Figure 6 shows the scheme of the algorithm presented in section 3. As the hardware algorithm can be implemented as a massive pipeline structure, it is possible to present the results for the KKFMF of the first camera row with a delay of three camera rows. The resulted depth map is passed to the hardware depth-histogram generation just in time. Thus the actual depth histogram is available $70\mu s$ after passing the last pixel of the image from the cameras.

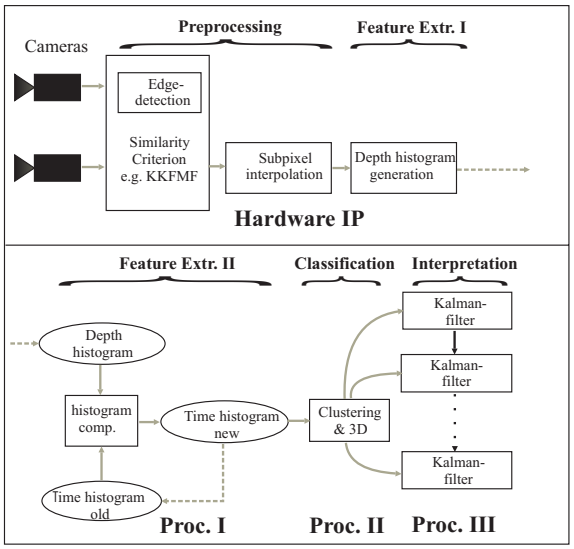

Fig. 6. Hardware-Software Co-Design for the presented algorithm

The other algorithms are processed on 3 pipelined softcore processors with 75 MHz each. The first generates the time-histogram (processing time 15 ms), the second is responsible for the clustering (processing time 35 ms) and the last deals with the Kalman-filtering (processing time 10 ms). The results are passed to a logic for external communication. The algorithm is implemented as the slice model of image processing by using the hardware model. The low-level local KKFMF with the inherent edge detection is a hardware module and passes the data to the subpixel interpolation and then to the global depth histogram generation. The algorithms are pipelined connected via simple local interfaces. The complex medium-level generation of the time histogram is implemented on a processor as well as the clustering and the Kalman-filtering which are High-Level applications.

The processing time of the algorithm realized as a hardware-software co-design is 60 ms. This is a reduction factor of 6 compared to the processing time of the PC. The processing

delay is 1.5 images, but no data will be lost due to the maximal processing time of 35 ms on the processors. Thus, the embedded image capturing and processing system is working in real-time. The pipeline delay is acceptable for most applications.

3. MEASUREMENT METHOD

Stereophotogrammetric measurement system

In photogrammetry, systems of multiple cameras are used for the optical 3-d measurement of objects or scenes. The use of a stereo camera system is optimal for balancing the technical costs with the possibilities of the measurement system [9]. The process which we developed also makes use of stereophotogrammetry. The automatic analysis methods for systems of multiple cameras are extremely computationally expensive, and usually are realized in software at workstations. In the ideal case, using current computational technology, the analysis of an image pair lasts fractions of a second.

The image calculation time increases for sequences, due to architecture and operating system reasons. In the following, the necessary characteristics for the camera system will be discussed. First, for the measurement, a stereo image pair is simultaneously captured by a calibrated camera system which is aligned in the normal case of stereophotogrammetry. Thereby, the camera axes are parallel to each other. For the normal case of stereophotogrammetry [14] applies.

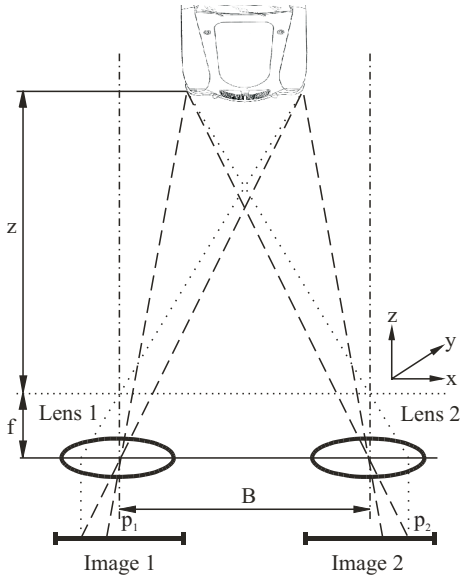

Fig. 7. Normal case of the stereophotogrammetry

$$Z = \frac{f \cdot b}{\Delta u};\ X = Z \cdot \frac{x_{left}}{f};\ Y = Z \cdot \frac{y_{left}}{f} \qquad (1)$$

In equation 1 X,Y,Z are the world coordinates of a certain point. The focal length f and the base width b are determined from the camera system. x_{left} and y_{left} are the image coordinates of the left image which is the reference image in this case. Δu finally is the disparity which needs to be determined during the measurement.

The use of the normal case of stereophotogrammetry ensures an overlap of the camera images over the entire measurement range from 10 to 100 m. At the farthest extremes of the measurement the accuracy should still be in the percent range and rather small objects like motorcycles should be recognized.

The error in a photogrammetric process depends on the pixel size / image resolution, base width (distance between the cameras) and distance of measurement. The error has a quadratic dependence on the distance to the measured object (see equation 1), therefore the base width and the resolution must be maximized to reduce error.

By using error propagation on equation 1 we get equation 2, where σ_z is the error of the distance measurement, $\sigma_{\Delta u}$ is the error of the disparity and z is the distance to the measured object.

$$\sigma_z = \frac{z^2}{f \cdot b} \cdot \sigma_{\Delta u} \qquad (2)$$

The error of the distance measurement depends on the squared distance itself. By using an optimized camera system the error for distance measurement can be reduced. In table 2 the error for the distance is shown against the distance and the focal length.

Distance[m]	Error of Distance Measurement [m] Base Width 0.8 m		
Focal Length	12mm	25mm	35mm
10	0.011	0.005	0.003
50	0.277	0.133	0.095
100	1.108	0.532	0.380
150	2.494	1.197	0.855

Tab. 2. Error [m] of the Distance Measurement

Considering table 2 the error is reduced to an acceptable level by using a base width of 80 cm, a resolution of 1024 pixels per line and a focal length of 25 mm. Base width and resolution could also be optimized by reducing the measuring range as well. Should the base width be lowered while retaining a constant measurement range, the resolution must be increased to hold the error constant. In series-production vehicle a robust camera setup with a base width of 80 cm is nearly impossible due to design reasons. To reduce the base width to 30 cm, which is acceptable, the image resolution must be increased to 2048 pixel per line to keep the error in the same range or the accuracy must be increased by a subpixel interpolation (see equation 5).

Further on, to reduce computational costs, epipolar geometry should be used. This way, the search can be simplified from a two-dimensional to a one-dimensional correspondence problem, if the cameras are very accurately aligned with one another or if the images are adequately rectified. Both images are then correlated by using an area correlation method. In doing so, a reference block from one image is compared with several search blocks from the second image. Using equation 3, the horizontal displacement (disparity Δu) between an object's references in both images can be calculated (Fig 8).

$$\Delta u = |p_1 - p_2| \qquad (3)$$

Δu - disparity
P_1 - object pixel position in the left image
P_2 - object pixel position in the right image

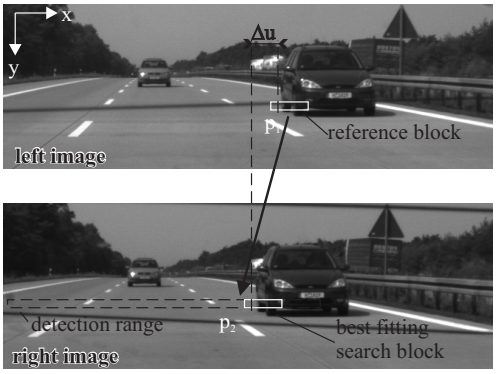

Fig. 8. Principle of area correlation

Once the disparity is known, the 3-d coordinates of the measured object can be attained (equation 1) in reference to the camera coordinate system through use of data from the camera system (base width B) and the calibration (camera constant) through triangulation.

A similarity criterion is used to calculate the disparity. Optical systems in outdoor vehicles are not operating under certain lighting conditions. The normalized zero-mean cross correlation function (KKFMF equation 4 [4]) is well-suited for use in vehicles, since it suppresses additive as well as multiplicative errors. However, the absolute brightness information is lost through the normalization.

$$Q(x,y) = \frac{\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}\left(\overline{F(i,j)}\cdot\overline{P_r(\xi+i,\eta+j)}\right)}{\sqrt{\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}\overline{F(i,j)}^2\cdot\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}\overline{P_r(\xi+i,\eta+j)}^2}} \tag{4}$$

$\overline{F(i,j)}$	-	zero mean pixel values of the search block
$\overline{P_r(\xi+i,\eta+j)}$	-	zero mean pixel values of the reference block
m,n	-	search window dimensions
ξ,η	-	displacement in x, y- direction

Area correlation methods have the characteristic that they can only operate when there is enough information within a block. Therefore, another criterion is introduced in order to prevent blocks with insufficient information from passing further processing. The denominator of the KKFMF consists of the combined variances of reference and search block. A high information content results in places such as the edges of the vehicle. By choosing acceptable threshold values and analysis of the variance, the relevant image area can be selected.

Having enough information in the reference block is an essential condition for getting meaningful values. Furthermore the measured object must still cover the main part of the reference block. By analyzing several scenes and taking the limits of a hardware implementation into account a block size of 16 pixels gives the best results for our system. Using the proposed camera system with a base width of 80 cm, a focal length of 25 mm and a horizontal resolution of 1024 pixel an object like a car is about 20 pixels wide at a distance of 100 m. Thus the edge of the car still covers a significant part of the block. And objects like motorcycles can be detected as well at this distance.

$$\Delta u_r = \Delta u + \frac{\frac{1}{2}(P_{(1)} - P_{(-1)})}{2\cdot P_{(0)} - P_{(1)} - P_{(-1)}} \tag{5}$$

Δu	-	disparity uncorrected
Δu_r	-	disparity corrected
$P_{(1)}$	-	KKFMF value after the maximum
$P_{(0)}$	-	KKFMF value of the maximum
$P_{(-1)}$	-	KKFMF value before the maximum

The accuracy of the disparity is increased to the sub-pixel level by using quadratic interpolation, in order to raise the accuracy of the distance measurement. Hardware interpolation is limited to $\frac{1}{8}$ pixel accuracy.

Hierarchical distance measurement

In many applications of environment sensing, it makes sense to have a constant error over the entire measurement range. Images have a high redundancy respectively if only a specific information is needed. We indicate an interesting object by its rough structure and its behavior over time. Thus no color images are needed. For the measurement in this case we need only the positions of some points of the object to determine its position. For the ascertainment of the object size only the left and the right edge are needed. Therefore only very small resolution is necessary. The high resolution images are only needed due to the high measuring range.

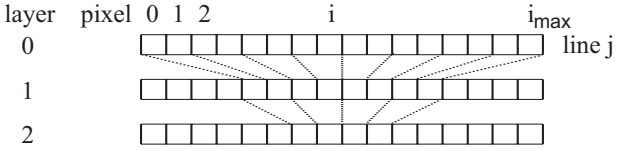

Fig. 9. Generation of the pyramid levels

Taking advantage of this fact, the area correlation algorithm can be optimized in terms of computation for hardware implementation using an image pyramid [22]. This makes use of the fact that the camera's full resolution capabilities are only necessary for the largest distances in the measurement range, while the high resolution is rather inconvenient to close range objects, because of the large disparity values. To reduce computation costs different degrees of resolution are assigned to the levels of the image pyramid. The resolution of each level will vary by a factor of 2 (see Fig. 9). Each level of the pyramid is only responsible for a certain distance range. This way, the entire measurement area can be covered through the combination of all level results.

The number of required calculations is reduced logarithmically by this procedure, particulary for close range measurements. This effect counteracts the inverse relationship between object distance and computational cost.

Due to the application of epipolar geometry, the processing can be realized in a row-oriented manner. In each level, rectangular blocks (16 x 1 pixels) will be chosen from the reference and search image, and compared with one another. Initially, the first reference pattern is shifted pixel by pixel from the start position over the search block. A correlation value is then calculated for each pair of samples (particularly

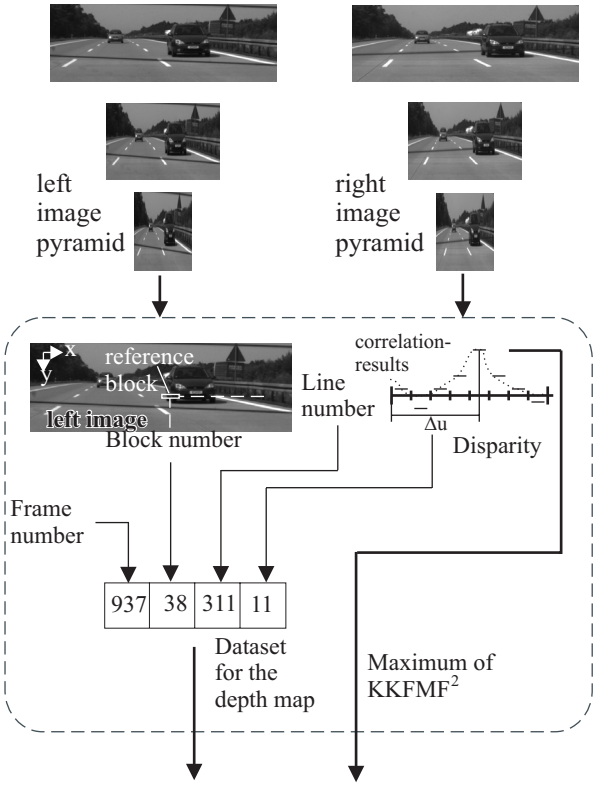

Fig. 10. Generation of the dataset for the depth map

KKFMF, equation 4). For each reference block, there is a maximum search disparity of e.g. 16 pixels. In the resulting search area, the positions of the maxima, which are located above a given threshold value are calculated and relayed to further analysis. Furthermore only blocks with high image information are processed. Therefore the variance (denominator of equation 4) is compared to a threshold. This means that only the maxima, which correspond to object features (e.g. edges) will be included in the further computations. Then, the disparity with the highest correlation value from all levels is chosen for each reference block. Disparity values from levels with reduced resolution must be projected to the original resolution. Then, the 3-d coordinates will be determined in sub-pixel accuracy. The sub-pixel accuracy is calculated using quadratic interpolation of the maximum of the similarity criterion, Q(x,y) [17], and the 3-d coordinates are calculated using equation 1.

Fig. 11. Generated depth map (distance coded with gray values)

These values are collected for an image and result in a depth map figure 11.

In order to test the method, the measurements were taken

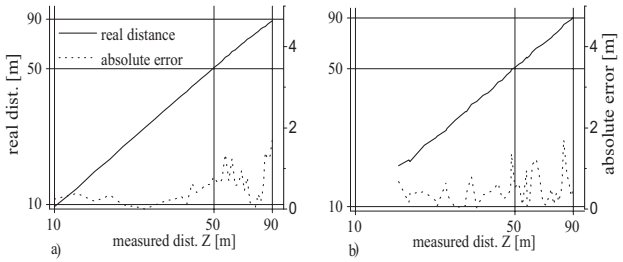

Fig. 12. Measurement results (left: static, right: highway)

statically as well as dynamically. Test were run outside the lab with defined, synthetic objects. These were then repeated with a vehicle as the measured object. The results of the vehicle measurement are shown in Figure 12, left. Further tests were carried out on the highway (Autobahn). As a reference, the laser distance measurement device Lasertape FG21-HA was used. The results of a day trial are shown in Figure 12, right.

Calibration

Numerical correction of the stereo image pair on the base of calibration data is computationally costly. For this reason, an optimized path for the correction of the calibration data is used. By applying the standard-camera-model [1], the systematic errors of the camera systems are compensated. The correction values of ΔZ and ΔX can be determined using the known formulas (equation 6) from the normal case of stereophotogrammetry. The coefficients d, e, f, as g, h, i are acquired during the calibration procedure.

$$\Delta Z = d{\cdot}Z^2{+}e{\cdot}Z{+}f \; and \; \Delta X = g{\cdot}Z{+}h{\cdot}X{+}i \quad d, ..., i \in \Re. \tag{6}$$

Equation 7 results from combining ΔZ and ΔX with the formula for the normal case equation 1.

$$Z = \frac{k}{du^2} + \frac{l}{du} + m \; (k, l, m \in \Re) \tag{7}$$

After the combination of equation 1 and equation 6 into equation 7, only 3 coefficients (k,l,and m) must be attained instead of the 6 formerly necessary. The 3 coefficients contain the camera constants and the base width. The derivation of the equations for X and Y result from considering the x-position of the object in the reference image. This correction calculation is an adaptation of calculation possibilities of microprocessors.

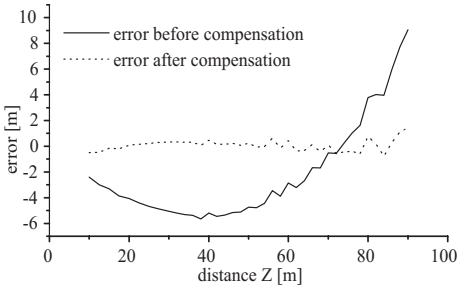

Fig. 13. Error before and after compensation

The number of mathematical operations is substantially decreased through the use of only 3 coefficients. A typical example for the effect of the correction is shown in figure 13.

The results from the distance measurement with the calibrated system are displayed in figure 12.

The derivation of the corrections in x-direction results in equation 8.

$$\Delta X = o \cdot Z + p \cdot X + q \ (o, p, q \in \Re) \tag{8}$$

The systematic error is not fully independent from the x-coordinate in the image, but for the accuracy needed in our application the effect is negligible.

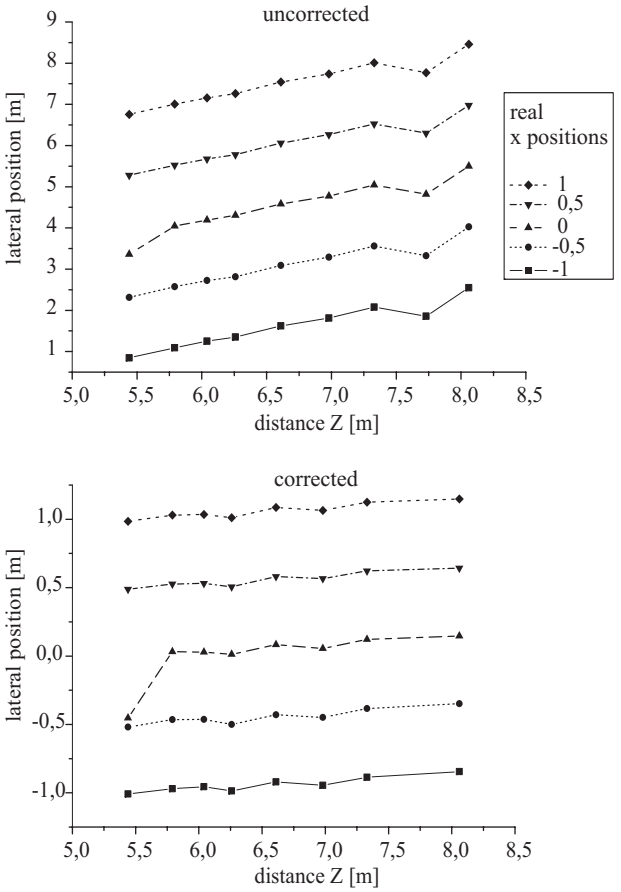

Fig. 14. Measurement results (left: static, right: highway)

To check the effect of the correction several scenarios were tested. Test patterns were moved in x-directions and z-direction. In figure 14 the effect of the correction for test pattern with defined positions which were moved in z-direction are shown.

Hardware implementation for real time purposes

The algorithms were implemented in the programming language C/C++ for evaluation and testing. Afterwards it was implemented in hardware for realtime purposes using the methods descirbed in section 3. Therefore some parts of the processing must be optimized. Due to the usage of square roots proves to be problematic for hardware, the squared KMFMF is implemented, since only the position of the maximum is relevant. Before squaring the nominator, all negative values must be excluded to prevent any false maxima.

Carrying out a regular area correlation on the basis of epipolar geometry to a maximum disparity of 256 pixels increases the amount of processed data and therefore the required work speed by a factor of sixteen. By using hierarchical methods with otherwise the same specifications, the amount of data is only doubled. Due to that fact, the correlator only needs to work with the doubled pixel clock to achieve continuous real-time processing.

The limitation to a single-row correlation window allows to minimize the memory requirements for the implementation. At first, one row must be saved in a dual-port-RAM for the creation of the levels. In doing so, a separation between the circuits with the two clocks can be carried out.

After correlation, the inter-level analysis runs with the doubled speed. At this time, the maximum of the correlation function is searched for. This results in a data set showing image number, block number, row number and the disparity. The data sets are subsequently saved in the dual-port-RAM and simultaneously read for recombination of the different levels. This compares the respectively overlapping blocks by their correlation values. The block with the largest correlation value is used for the generation of the depth map. Again, the dual-port-RAM is used as a separation between the areas with different clocked dircuits. The depth map is then transferred to the processor for further analysis(see section 4).

4. CLUSTERING AND TRACKING

After the generation of the depth map, the object must be identified. For this purpose a general cluster method is applied.

At this point, the only methods that should be considered are those, which allow for automatic analysis of data attained by stereo image analysis. In general, cluster methods can be divided into geometrical and statistical methods.

Both types operate with usage of a-priori knowledge. Geometrical methods are often used for analysis of depth maps. In doing so, a feature space is stretched out over a 3-d space, in which the 3 features correspond to spatial coordinates.

These features can be subsequently clustered into an object by using the vehicle's dimensions [13]. Because of this, the entire database is run through the algorithm multiple times.

A different approach is the observation of statistical [21] occurances with certain characteristics in the database. This approach has the advantage that each point only needs to be tested once. Essentially, statistical methods count the number of points with certain characteristics; for instance column number and distance range. Statistical methods are more effective than geometrical methods, in cases where their limited capabilities suffice.

Object Detection

Clustering is used to detect vehicles with a velocity null or positive relative to the own car. Here a histogram based approach [21] is used. Another cluster method is described in [13]. To detect the vehicles, a histogram is generated from the depth-map, here we call it depth-histogram. Because of the reduced resolution of the levels, there is a different amount of data within each level. Thus we adapt the memory organisation of the depth-histogram (see figure 15). The abscissa represents the lateral offset of a detected 3-d-point and the ordinate represents the disparity values for each level. The disparity ranges from 0 up to 7. To calculate the original lateral offset see equation 9 and for the disparity values see equation 10.

$$x_{Org} = (OFFSET_{Block} + (x_{Level} \cdot INCR)) \cdot 2^{Level} \tag{9}$$

$$\Delta u_{Org} = (OFFSET_{Disp} + \Delta u_{Level}) \cdot 2^{Level} \qquad (10)$$

Because of the dimensions of the search and the reference block OFFSET values have to be used. The OFFSET in equation 9 represents the center of a block. In this application a block is $16x1$ pixels, thus the OFFSET value is 8. The INCR argument represents the block increment, that means the difference between the center of two neighboring reference blocks. The OFFSET in equation 10 represents the minimal disparity. Because of the limited detection range of $100m$, all disparity values smaller then 8 are not saved in the depth-histogram.

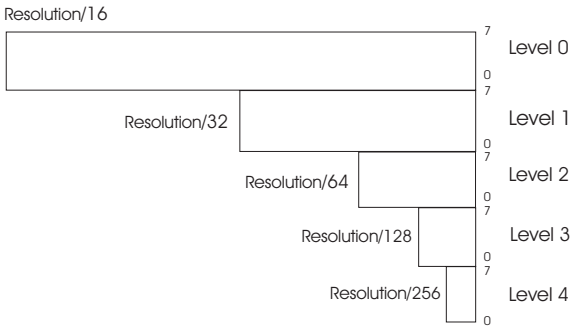

Fig. 15. Memory organization of the depth-histogram

The edges from cars, road signs and other raised objects are detected by using the KKFMF (equation 4). Different edges of an object have nearly the same disparity values at the same lateral offset but in different rows of the image.

The depth-histogram is generated by accumulating the frequency of occurrence for the different disparities of each column in the depth-map (figure 11). Raised objects generate local maxima in the depth-histogram (figure 16). The latest y-coordinate is saved as well to define the base of a cluster within an image. A threshold is applied to detect the raised objects. Entries above a certain threshold are marked as raised. To use the same threshold over the whole depth-histogram the frequency of occurrence is reduced by factor two for each level.

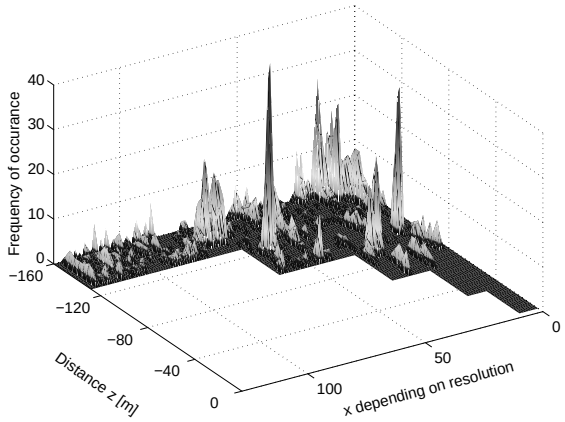

Fig. 16. Depth-histogram of one image

To track objects over time, a time-histogram (figure 17) is used to detect raised moving objects. Furthermore trees and road signs can be filtered out.

To detect the vehicles, the entries of the depth-histogram are used to generate a time-histogram. Entries in the time-histogram are increased by one, if these positions in the depth-histogram are marked as raised. This represents the age of a raised object. Before increasing the value, a search algorithm is started to find the highest time-histogram entry close to the current histogram position. If there is a higher entry, this entry gets the current position with an increased age. For all objects in the time-histogram, with no complementary object in the depth-histogram the age is decreased.

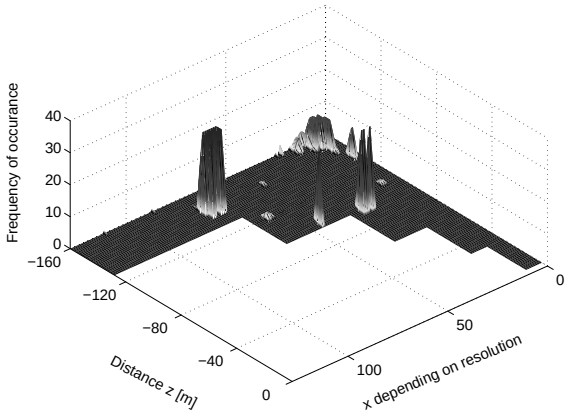

Fig. 17. Time-histogram

The objects with a certain age are used for clustering. Because of the adapted resolution an object appears within the time-histogram with nearly the same dimension, over the whole measurement distance. Therefore, the cluster algorithm can be accomplished within the time-histogram. By using a simple search approach, time and system performance can be saved.

The cluster position is detected by calculating the average of the positions of the local maxima belonging to this cluster. For this middle position only, a 3-d-value is computed. The positions of a cluster from various samples are transferred to a Kalman-filter. In order to decrease outliers in the distance and velocity calculations, a Kalman-filter for each cluster is dedicated to smoothing, due to its low-pass-characteristics. In the same way, the movement of an object is tracked by extrapolation, in case of a temporary disappearance of the object. The results of a velocity smoothing over 100 samples are displayed in Figure 18.

Because of the adapted resolution and the characteristic of a histogram the objects have to feature minimal extensions to be surely detected. The two cameras subdivided into search blocks, hereby the minimal width is $x = 600mm$, for surely detected vehicles. With a minimal height of $y = 534mm$ a vehicle has a height of 8 pixel at a distance of $140m$. With our history threshold this object can be detected. The frequency of occurrence (height in pixel) is low at far distances and rises with the approaching object, at the level boundaries the frequency is adapted with the resolution. With this minimal dimension even bikers can be detected with our system.

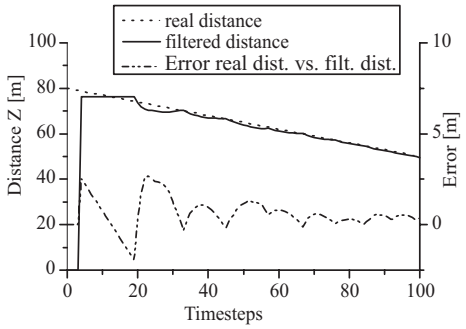

Fig. 18. Filtered distance vs. Error of the distance

LaneDetection

To detect the lane position the Hough Transform [15] is used, other lane detection methods are presented in [24], [18], [7]. Because of the system characteristic with a hardware-software co-design the depth-histogram is used as the starting point. The depth-map is not saved yet.

The Hough Transform is a method to detect collinear points of an image. Collinear points are located approximately on a line. The basis is the normal parametrisation of a line (equation 11).

$$r = x \cdot cos\varphi + y \cdot sin\varphi \tag{11}$$

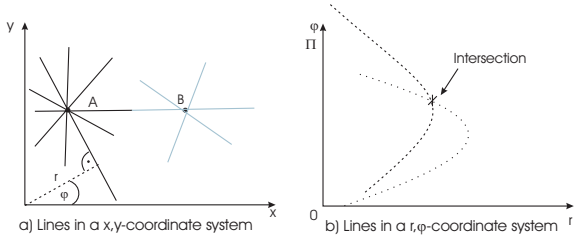

Fig. 19. Hough Transform

If there is single point A (figure 19.a) the gradient of the line through point A is lost. Therefore it is necessary to compute the transformation for a number of lines with a common point A, but with different gradients. By transforming these lines, a figure as presented in figure 19.b is generated. Points A and B have one common line, this is the point of intersection in figure 19.b. All collinear points have a common line, with the same gradient φ and radius r in the r, φ-coordinate system. Using an accumulator (see figure 20) the number of common lines can be detected . If a point in the accumulator exceeds a threshold, this position characterizes a line. The line coordinates are transformed to the x, y-coordinate system. Along this line the flat points are searched and marked as belonging to a flat cluster.

To apply the Hough Transform, an applicable input data set has to be used. To detect flat objects we need a lower threshold for the depth-history. To avoid errors this threshold is applied for points that are not marked as raised. Because of the visibility of the lane markers the Hough Transform is limited to distance levels 1, 2 and 3. Lane markers are mostly aligned lengthwise, therefore the generation of clusters of lines is shortened to 0 up to 50 degree for the left side of the image

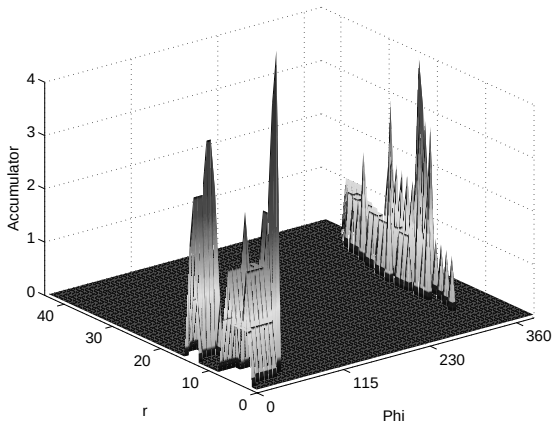

Fig. 20. Accumulator Hough Transform

and to 310 up to 360 degree for the right side of the image, whereby the resolution of the angel is limited to 5.625 degrees (360 degree/64 Steps). Due to this limitation the accumulator entries in figure 20 are concentrated in two areas. There are gaps compared to figure 19.b). The result of the Hough Transform is shown in figure 21.

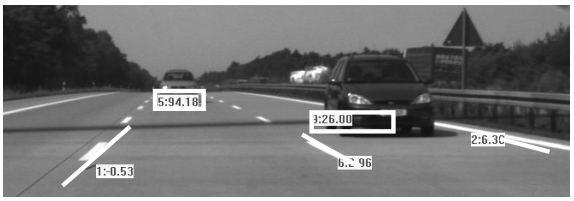

Fig. 21. Lane Detection with the Hough Transform

Testing

With help of the testing environment (section 5), the tests could be run with several image sequences and real time conditions. Sequences with various traffic conditions as well as sequences with motorcycles (see Figure 22) were analyzed.

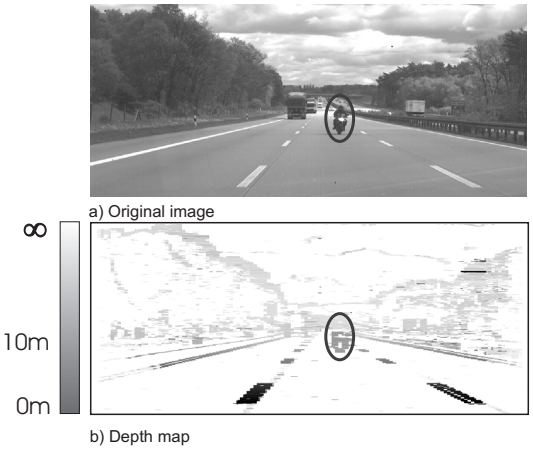

Fig. 22. Sequence with Motorcycle

In the case of night driving (see Figure 23), the process showed a high ability to analyze the floodlights in the depth

map. However, because of other statistical circumstances, an adaptation of the clustering method would be necessary for night driving.

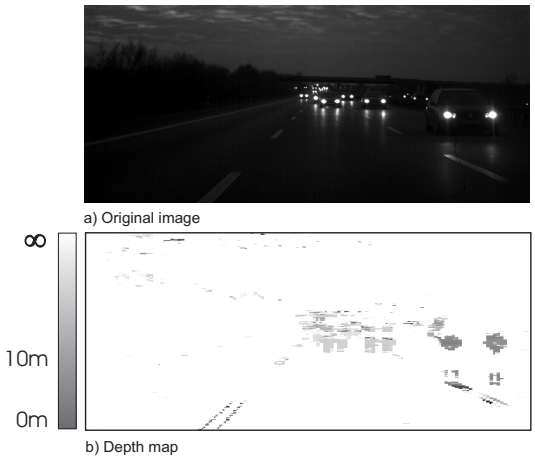

Fig. 23. Depth map of a night trial

Software Implementation

The algorithms for the clustering, tracking and lane detection were primarily implemented in C/C++. In the calculations, specific operations such as division or roots were avoided by using other operations in order to decrease computation time. Furthermore, floating point operations are replaced by integer operations, if possible. In our case, doing so nearly doubled the processing speed.

Thus adequate processing speed is achievable, even with a small embedded processor.

5. DEVELOPMENT ENVIRONMENT

In order to realize a stable FPGA implementation, many tests and simulations are necessary. Because of the complexity of image information, the test vectors for the simulation are very long. For this reason, a comparison implementation for testing was run in software written in C/C++, which is binary compatible to the hardware implementation. The simulation data and the software results were then compared through a specially-designed software. For the testing, real and synthetic image data from static and dynamic measurements are used and translated into test vectors.

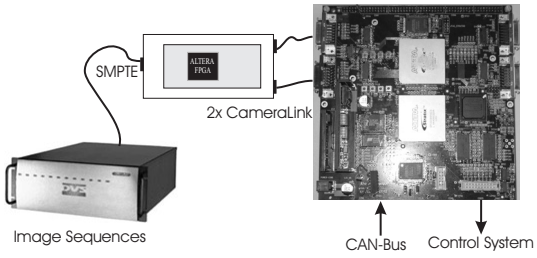

Fig. 24. Development Environment

For development purposes implementing the experimental system in a vehicle is not reasonable. For this reason, a camera simulator was made for stereo image sequences.

As a basis for this, a digital video recorder from the company DVS was used, as it is used in television studios. It is used for the recording and playback of uncompressed image sequences in HDTV- Standard (SMPTE 292M) [20] with a resolution of 1920x1020 pixels.

An image in HDTV- Standard was saved as an interlaced stereo image pair. Using a converter, the HDTV signal was separated into 2 images, and transferred over a CameraLink interfaces to the experimental set-up. This way, the real timing and real image data could be used in laboratory tests.

Because the digital video recorder is too sensitive for use in vehicles, recent computational technology must be used. Two PCs with frame grabbers are used for capturing, with which the image sequences could be taken with synchronized image pairs to the extent of 1000 images.

Due to needed interaction with the car data system the CAN-data is recorded with the captured images. Both images and CAN-data are held with a synchronized time stamp. Afterwards the CAN-data is coded into not used pixels of the images to provide synrchonized CAN-data with the images.

6. CONCLUSION

This article introduced a system for real-time multi-object position measurement using a hardware-software co-design. The system is based on algorithms from stereophotogrammetry and, through a set-up of cameras in the normal case of stereophotogrammetry. It was tested with static and dynamic measurement objects.

For 3-d measurement, an algorithm based on area correlation was modified and implemented in the hardware. The modification of the algorithm is based on several levels of an image pyramid, which represented non-overlapping distance ranges. The computational costs were reduced by a factor of 8 for the algorithm using 5 levels in comparison to the original area correlation. After Hardware 3-d-measurement the amount of data is heavily reduced and suitable for further processing in an embedded processor.

The software differentiates between static and dynamic objects and lane markings and handles them accordingly. A statistical cluster method [21] is applied, which is a good compromise between the necessary computation time and available capabilities. Both the method for producing the depth map and the similarity criterion are robust against image interference and other disturbances.

In tests, the system proved to be capable of real-time application with accuracy levels in lower percent range, while at that the same time covering the specified measurement range. Furthermore, it was shown that, in comparison to PC implementation, a hardware-software co-design is more suitable for real-time applications.

The hardware is calculating the depth map in 70 μs. This value can be treated as a constant delay in respect to the image capture, since it results from the pipelining of the correspondence analysis. Due to the a massive data reduction during hardware 3-d-calculation the clustering of the necessary amount of data can be archived in an embedded software in capture time of one image. A new object will be recognized and identified as static or moving within the run-time of 5 images. At this juncture, the transient effect of the Kalman filter begins (see Figure 18). This way, new objects are recognized after nearly 0.2 seconds.

A lane-changing assistant system serves as an application example [13] on the first hand, in which the space behind the vehicle is surveyed. Further application areas include real-time 3-d-measurement for driver assistance systems for the frontal view, autonomous robots as well as human body movement. The measurement range is adaptable according to the parameters of the chosen applications.

Acknowledgement:
This research project was supported by BMBF-grant 03i1226a, BMBF-grant 03GL0049/LSA-grant 0028IF0000, AiF-grant KF0056102WD4, AiF-grant KF0056103SZ5 and EU-grant 0046KE0000.

REFERENCES

[1] J. Albertz and W. Kreiling, *Photogrammetric Guide*, 4th ed. Herbert Wichmann Verlag GmbH, 1989.

[2] Altera, *Simultaneous Multi-Mastering with the Avalon Bus*, 1st ed. Altera Corporation, 2002.

[3] ——, *White Paper - Stratix Device Backgrounder*. Altera Corporation, 2003.

[4] P. F. Aschwanden, "Experimenteller Vergleich von Korrelationskriterien in der Bildanalyse," Ph.D. dissertation, ETH Zürich, 1993.

[5] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele, "Stereo vision-based vehicle detection," in *Proceedings of the International Conference on Intelligent Vehicles*. Detroit (MI),USA: IEEE Industrial Electronics Society, Oct. 2000, pp. 39–44.

[6] S. Brown and J. Rose, *FPGA and CPLD architectures: a tutorial*, ser. 2. IEEE Press, Summer 1996, vol. 13, pp. 42–57.

[7] E. Dickmanns and B. Mysliwetz, "Recursive 3-d road and relative ego-state recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 199–213, 1992.

[8] D. Ehmanns and A. Hochstädter, "Driver-model of lane change manoevres," *7th World Congress on Intellgent Transportations Systems*, Turin 2000.

[9] O. Faugeras, *Three-Dimensional Computer Vision*. MIT-Press, 1993.

[10] K. C. Fuerstenberg, K. C. J. Dietmayer, and U. Lages, "Laserscanner innovations for detection of obstacles and road," in *AMAA*, 7th International Conference on Advanced Microsystems for Automotive Applications. AMAA, 2003.

[11] R. K. Gupta, *Co-Synthesis of Hardware and Software for digital embedded systems*, 2nd ed. Kluwer Academic Publishers, 1997.

[12] J. Kaszubiak, M. Tornow, R. Kuhn, B. Michaelis, and H. Bresch, "Hardware-software co-design for an optical, real-time object detection and tracking system," in *Electronic Proceedings of the GSPx'04*. Santa Clara/USA: GSPx'04, Sept. 2004.

[13] C. Knoeppel, A. Schanz, and B. Michaelis, "Robust Vehicle detection at large distance using low resolution cameras," in *Proceedings of the International Conference on Intelligent Vehicles*. IEEE Industrial Electronics Society, 2000, pp. 36–41.

[14] K. Kraus and P. Waldhäusl, *Photogrammetrie*, ser. 3. Bonn: Ferd. Dümmler Verlag, 1990, vol. 1.

[15] V. F. Leavers, *Shape Detection in Computer Vision Using the Hough Transform*. Springer Verlag, 1992.

[16] S. Lee and Y. Kay, "A kalman filter approach for accurate 3d motion estimation from a sequence of stereo images," *10th International Conference on Pattern Recognition*, pp. 104–108, 1990.

[17] R. Mecke, "Grauwertbasierte Bewegungsschätzung in monokularen Bildsequenzen unter besonderer Berücksichtigung bildspezifischer Störungen," Ph.D. dissertation, Universität Magdeburg, 1999.

[18] F. Paetzold, U. Franke, and W. von Seelen, "Lane recognition in urban environment using optimal control theory," *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 221–226, 2000.

[19] K. Saneyoshi, K. Hanawa, K. Kise, and Y. Sogawa, "3-d image recognition system for assist, proceedings of the intelligent vehicles 1993 symposium," *Proceedings of the Intelligent Vehicles 1993 Symposium*, pp. 60–64, 1993.

[20] SMPTE292, *SMPTE 292M for Television - Bit Serial Digital Interface for High definition Television Systems*. Society. of Motion Pictures and Television. Enginieers, 1998.

[21] A. Suppes, *Stochastische Hindernisdetektion aus stereoskopischen Videosequenzen fuer fahrerlose Transportfahrzeuge*. VDI Verlag, 2004.

[22] M. Tornow, B. Michaelis, R. Kuhn, R. Calow, and R. Mecke, "Hierarchical method for stereophotogrammetric multi-objekt-postion measurement," *Pattern Recognition, DAGM Symposium*, pp. 164–171, 2003.

[23] W. Uhler, H.-J. Mathony, and P. Knoll, "Driver assistance systems for safety and comfort," Robert Bosch GmbH, Driver Assistance Systems, Leonberg, EU-Projekt EDEL im 5. Rahmenprogramm, edel-eu.org, 2003.

[24] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision Computing*, vol. 22, pp. 269–280, 2004.