PLATFORM FOR EVALUATION OF EMBEDDED COMPUTER VISION ALGORITHMS FOR AUTOMOTIVE APPLICATIONS

Wilfried Kubinger, Stefan Borbely, Hannes Hemetsberger and Richard Isaacs

seibersdorf research, Dependable Embedded Systems Group
Donau-City-Str. 1, A-1220 Vienna, Austria
phone: + (43) 50550 3127, fax: + (43) 50550 4150
{wilfried.kubinger | stefan.borbely | hannes.hemetsberger | richard.isaacs}@arcs.ac.at, web: www.arcs.ac.at

ABSTRACT

Stereo vision algorithms are usually designed to run on standard PCs or PC-based systems. Up to now and due to the limited resources (e.g. internal and external RAM size, CPU power, power consumption, etc.) no general embedded hardware exists, where different algorithms can be implemented and run properly. The presented test platform aims to close this gap and implement a convenient platform for the evaluation of different high level vision algorithms used for embedded systems. Furthermore, it offers an excellent opportunity for testing and evaluating automatic code generation tools (e.g. MathWorks Real-Time Workshop Embedded Coder). In this paper the complete system concept, the automotive application, a stereo vision example algorithm and experimental results are presented.

1. INTRODUCTION

It is nontrivial to identify suitable algorithms for a specific embedded real-time application. To approach this task a platform was developed, which enables the embedding of different computer vision algorithms on a real-time system. The evaluation of the performance was based on an automotive application, namely platooning [3]. The advantage of the platform is the possibility to easily embed different computer vision algorithms and compare the performance of each other. Since model-cars have been used in this testbed, small, lightweight, and power-aware solutions were used, because of the apparent limitations. Furthermore, it is cheap and easy to maintain, compared to real cars. In the presented prototype, it had to be dealt with real-time and embedded systems requirements, otherwise the algorithms – and therefore the platooning application – would not work properly at all.

In the following sections the configuration of the platooning testbed will be presented. The whole concept of the test platform will be described and the achieved results of the realized experiments will be discussed. The paper will conclude with a discussion on future work.

2. SYSTEM CONCEPT

2.1 Platooning Testbed

An overview of the testbed is shown in Fig. 1. It consists of a leading car (car A) and a trailing car (car B). Car A is built by using the Lego Mindstorm series and can be programmed to travel a pre-programmed or random path. The main purpose of car B is to follow car A by keeping the distance between both cars constant. This behavior is called *platooning* [3]. Fig. 1 also shows the definitions of the reference coordinate system, distance r and angle φ. The task of the platooning car B is to keep the pre-defined distance r constant and to minimize the angle φ.

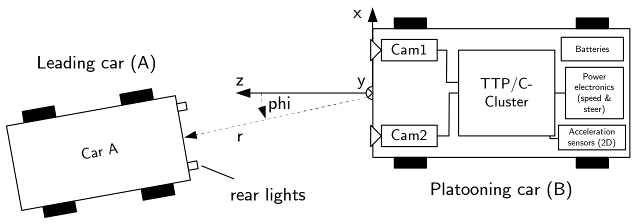


Figure 1: Overview of the platooning testbed

2.2 Test Platform

The test platform is based on a model truck with a Direct Current (DC) drive and steering mechanism. The remote control, which is normally used to control the truck, has been replaced by an on-board embedded system. The system is responsible for monitoring and steering the test platform (respectively the model truck).

The model truck (see Fig. 3) is equipped with a pair of *Basler 601fc* IEEE1394 “FireWire” Complementary Metal Oxide Semiconductor (CMOS) cameras. The cameras are mounted at the front leftmost and the rightmost side of the model truck to obtain the desired stereo effect.

Fig. 2 represents the design of the on-board embedded system. It reflects a distributed embedded system solution, consisting several nodes for battery-powered operations. The core of the system is a TTP/C-cluster [7] composed of the following three different nodes:

- The *ControlNode (CN)* is a TTTech PowerNode [7], running under TTP/OS, and is responsible for acceleration/deceleration and steering servo drives.
- The *SensorNode (SN)* is also a TTTech PowerNode, running under TTP/OS. It interfaces all sensors except the cameras. Two acceleration sensors are responsible for crash warning. A magnetic cavity has been used as a speed sensor. The steering angle sensor has been realized by using a potentiometer.
- The core of the *VisionNode (VN)* is a high performance Digital Signal Processor (DSP) based on a platform by the *Orsys GmbH* [5]. It triggers and receives images from the stereo digital camera system. The stereo head is connected to the VN by a 400 Mbit/s IEEE1394 “FireWire”-bus, using the isochronous frame transmission mode [5]. Furthermore, the computer vision algorithms are executed on this node.

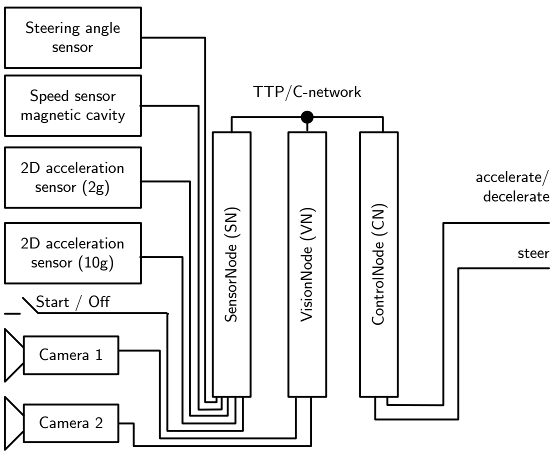


Figure 2: Cluster design

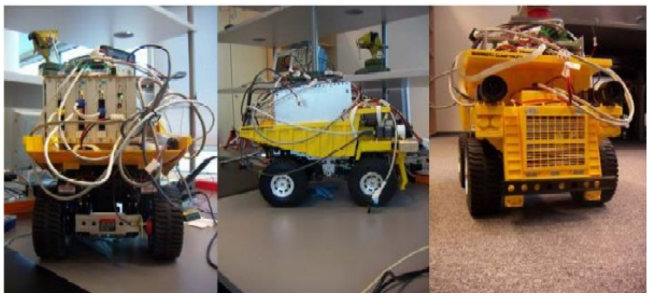


Figure 3: Model truck

2.3 Stereo Vision Sensor

The stereo vision sensor is based on two IEEE1394 "FireWire" cameras, which are connected to the VN. To guarantee a synchronized acquisition of both images from the left and the right camera, the same hardware-trigger signal is used.

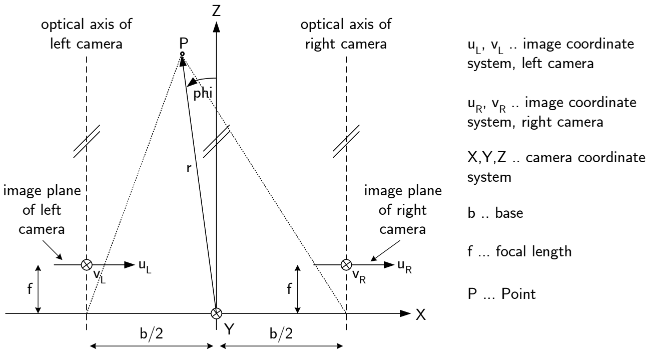
Each camera delivers a 2D-image of the scene. The main task of a 3D-reconstruction algorithm is to map the original 3D position of the *point of interest* from the 2D information. For the successful reconstruction of a point in 3D, it is necessary to know the spatial coordinates of the same point in the left and in the right camera image. Therefore, the first step is the extraction of corresponding points from these two images.

For the calculation of the 3D coordinates of a point, standard geometry is used, as shown in Fig. 4. The transformation between the spatial (u, v) and the image coordinate system (x, y) is done by using the following equation:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x_m \\ y_m \end{pmatrix}$$

where (x_m, y_m) denotes the intersection point between plane and optical axis.

A method called 3D-triangulation [1] has been used to calculate the 3D-position of the point of interest. The main advantage of standard geometry is that equations for calculating the 3D coordinates of any point in the camera coordinate system are become simple. The method starts with the calculation of the disparity $d_s = u_L - u_R$, which is a measure for the distance between the point of interest to the camera system. Based on this disparity and the assumption that $v_L = v_R$, the 3D position of the point of interest in camera coordinates can be calculated by:



Z
optical axis of left camera
optical axis of right camera
P
phi
r
image plane of left camera
image plane of right camera
u_L
v_L
u_R
v_R
f
f
Y
X
b/2
b/2
u_L, v_L .. image coordinate system, left camera
u_R, v_R .. image coordinate system, right camera
X,Y,Z .. camera coordinate system
b .. base
f ... focal length
P ... Point

Figure 4: Standard geometry and related coordinate systems

$$X = b \cdot \left(\frac{u_L}{d_s} - \frac{1}{2} \right), \quad Y = b \cdot \frac{v_L}{d_s}, \quad Z = b \cdot \frac{f}{d_s}$$

The main goal for platooning is to keep the distance between the two cars constant and to minimize the relative angle of car A with respect to car B. Therefore, a polar coordinate description in the X-Z-plane has been used (shown in Fig. 4). Distance r and angle φ are calculated as follows:

$$r = \sqrt{X^2 + Z^2}, \qquad \varphi = \frac{180}{\pi} \cdot \arcsin\left(\frac{-X}{r} \right)$$

The distance r describes the distance between the center of the stereo head and the rear light of leading car A. φ is the angle between the Z-axis and the current position of the leading car, since the algorithm detects both rear lights of car A. If r_L and φ_L denote the distance and angle of the left rear light, and r_R and φ_R denote the distance and angle of the right rear light, the distance r and the angle φ can be calculated as given below:

$$r = \frac{r_L + r_R}{2}, \qquad \varphi = \frac{\varphi_L + \varphi_R}{2}$$

3. EXAMPLE: STEREO VISION ALGORITHM

Based on a simple color-based tracking of the rear lights the capabilities of the test platform are described. As already mentioned, the stereo vision system of car B is used to detect the position of the leading car A. This is done by extraction of the red lights from both camera images. A principle schematic of the left camera image is presented in Fig. 5. The goal is to track both rear lights of car A by moving a so-called tracking window, where the red light is approximately in the center of the tracking window. To gain some robustness, both rear lights are tracked simultaneously.

The initial values for the algorithm (x_c, y_c, T_{1L}, T_{1H}, T_{2L} and T_{2H}) are obtained by a calibration procedure. Here, one image from each camera has to be grabbed and both images are searched for red lights. After finding the lights in both images, the initial values are extracted from the images. This routine is also used for a re-initialization of the algorithm after tailing off the red lights (e.g. after an occlusion). The

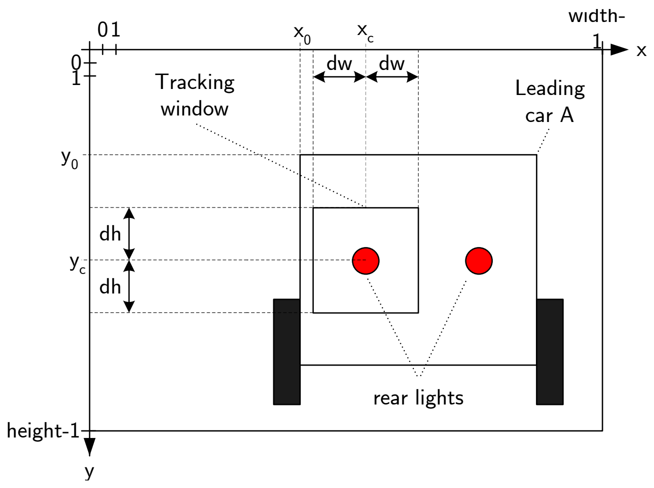


Figure 5: Schematic image of car A in left camera image

tracking algorithm itself will be described in detail as follows:

1. First, Regions Of Interest (ROIs) are calculated from both images. The coordinates for these regions are given by the equations:

$$\begin{gathered}(x_{cL_L}-dw, y_{cL_L}-dh, x_{cL_L}+dw, y_{cL_L}+dh)\\(x_{cR_L}-dw, y_{cR_L}-dh, x_{cR_L}+dw, y_{cR_L}+dh)\\(x_{cL_R}-dw, y_{cL_R}-dh, x_{cL_R}+dw, y_{cL_R}+dh)\\(x_{cR_R}-dw, y_{cR_R}-dh, x_{cR_R}+dw, y_{cR_R}+dh)\end{gathered}$$

where (x_{cL_L}, y_{cL_L}), (x_{cL_R}, y_{cL_R}), (x_{cR_L}, y_{cR_L}), (x_{cR_R}, y_{cR_R}) describe the centers of the four tracking windows for the four rear lights, two in the left and two in the right image. The following operations are described only for one rear light in one camera image.

2. Since the cameras use a Bayer RGB primary color filter for generation of color information, special care has to be taken with the calculation of the color information for each pixel. Each individual pixel is covered by a micro-lens which only allows enough light of only one color to strike the pixel. The color information of each pixel P

$$P(x,y) = \begin{pmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{pmatrix}$$

is calculated using an interpolation technique based on a 2x2 operator [2]. $R(x,y)$ denotes the red component of the pixel $P(x,y)$, $G(x,y)$ the green, and $B(x,y)$ the blue component.

3. In the next step the pixels of the tracking window are converted to a more useful color representation. The *YT1T2 color space* has been used, also know as *L1-normalized colors* [4]. Only the chromatic part of the color space model (which are $T1$ and $T2$) has been used, because it is more robust against variations in the light intensity. Under the supposition that the gamma correction circuit of the camera is deactivated and the response of the CMOS sensor cells to the incoming light is linear, the color space conversion can be realized as follows:

If

$$R(x,y)+G(x,y)+B(x,y)=0$$

Then

$$T1(x,y) = T2(x,y) = \tfrac{255}{3} = 85$$

Else

$$T1(x,y) = 255 \cdot \frac{R(x,y)}{R(x,y)+G(x,y)+B(x,y)}$$

$$T2(x,y) = 255 \cdot \frac{G(x,y)}{R(x,y)+G(x,y)+B(x,y)}$$

4. Segmentation of the image has to be performed next. Thus, each pixel is checked if it is part of the rear light of car A or not. That is done by a membership test. This test calculates if either the color of the pixel is inside the color class and associated with the rear light or not. The result of this member test is a binary image I with the size of the tracking window. In this membership test, only the chromatic part

$$P_{cc}(x,y) = \begin{pmatrix} T1(x,y) \\ T2(x,y) \end{pmatrix}$$

is used. Inside the tracking window, the following test procedure is applied to each pixel [6]:

If

$$(T1_L \le T1(x,y) \le T2_H) \text{ and } (T2_L \le T2(x,y) \le T2_H)$$

Then

$$I(x,y) = 1$$

Else

$$I(x,y) = 0$$

A one denotes that this pixel has been classified as part of the rear light. A zero means that this pixel is not part of the rear light. The color class associated with the rear light is established during the initialization procedure. A good balance between under- and over-segmentation must be found by a careful evaluation of experimental test results. This problem has been solved during initialization procedure by calculating the mean values of $T1$ and $T2$ of the rear light. For the size of the color class, 10% of the full scale range has been used. This range has been verified experimentally (see Fig. 6).

5. Next, the center of gravity of the binary image I is calculated. The following two values are calculated using all the segmented pixels:

$$f_x = \sum_{i=1}^{k} x_i, \qquad f_y = \sum_{i=1}^{k} y_i$$

Here k denotes the number of segmented pixels in the binary image I. Furthermore, x_i and y_i are the spatial coordinates of the segmented pixels. The *center of gravity cog* is calculated using the following formula:

$$cog = \begin{pmatrix} x_c \\ y_c \end{pmatrix} = \begin{pmatrix} \frac{f_x}{k} \\ \frac{f_y}{k} \end{pmatrix}$$

An example of such a segmented image is given in Fig. 6.

6. Steps 2 to 5 are repeated for the second rear light. Followed by repeating the same procedure for the image of the right camera.

7. The continuous tracking of a rear light of car A is as follows. The two calculated centers of gravity, cog_L for the left image and cog_R for the right image, are the basis for grabbing the next stereo camera image pair. Then the algorithm starts over with step 1 for the next cycle.

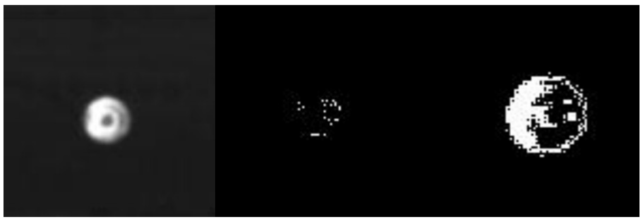


Figure 6: Segmentation of color image (only gray values are shown, segmented pixels are marked with white dots): Left – original tracking window; Middle – segmentation with $T_{1H} - T_{1L} = T_{2H} - T_{2L} = 12$; Right – segmentation with $T_{1H} - T_{1L} = T_{2H} - T_{2L} = 25$.

The result of this tracking algorithm is a pair of coordinates (x, y) for each stereo camera image, describing the detected position of the rear light of car A. Based on this calculation, the 3D position of car A with respect to car B can be calculated.

4. EXPERIMENTAL RESULTS

Many indoor trials were performed in the lab. The algorithm is robust and works under varying light conditions and car B was able to follow car A with a speed of 5km/h.

4.1 Real world test example

For this specified test, leading car A was manually pulled in front of the model truck (platooning car B). The speed controller was a semi analog *Discrete PI*, and the angle controller was a *Proportional Integral and Derivative* (PID) controller. Both controllers were coded by using the MATLAB Embedded Coder. Fig. 7 shows an example of a test run.

Curve No.1 represents the distance r between the cars. The engine voltage (curve No.3) and the *pulse width modulation* (PWM) value of the servo drive (curve No.4) follows the input exactly. The speed (curve No.2) follows the input poorly, since the speed sensor used is very slow and not precise enough for this application.

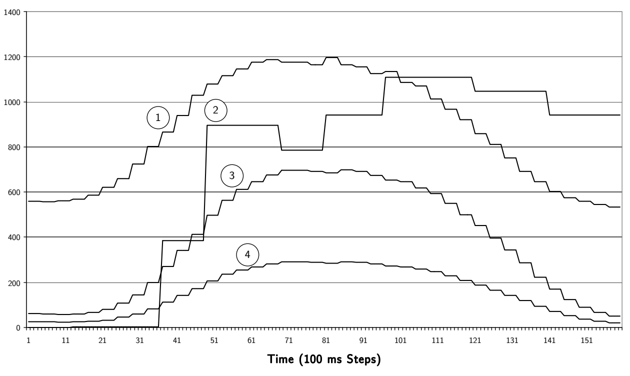


Figure 7: Experimental results

4.2 Performance

Four performance characteristics are relevant for the evaluation of the algorithm. There is the maximal *framerate* specifying the time difference between two subsequent frames to be processed, the *latency* specifying the time delay between the starting of the image acquisition and the action of the controller, and the usage of *internal RAM* and the *Code Size*. An internal RAM-usage of 2.3 kBytes and a code size of 78kBytes were achieved for the implementation of the stereo tracking algorithm. In addition, a framerate of 125ms and a latency of 200ms were obtained. The power consumption was 2.5W for the VisionNode and 3.4W for the stereo camera head. Latency and framerate were constant, because a time-triggered control cluster was used. This is especially advantageous for control applications, because time variations in either of the values would degrade the performance significantly.

Eight frames per second image processing is a very good performance for a non PC-based embedded real-time system of that size.

5. CONCLUSION AND FUTURE DIRECTIONS

In this paper a test platform for the evaluation of different embedded vision algorithms for automotive applications was presented. Furthermore, an example stereo vision algorithm was implemented, verified and tested. Performance measurements have been made and the results have been discussed. After that, the implementation feasibility of so called *processor & resources ravening* algorithms on embedded systems has been shown. These results are very important for future embedded vision applications with limited resources.

In the future, more complex embedded computer vision algorithms will be evaluated and compared. A significant reduction of the cycle time of the embedded vision system is also planned. The goal is an image processing time of 25fps (40ms), which is the maximum speed possible using the manual triggering of the used cameras.

REFERENCES

[1] S. Bahadori, L. Iocchi, “A Stereo Vision System for 3D Reconstruction and Semi-Automatic Surveillance of Museum Areas,” Workshop Intelligenza Artificiale per i Beni Culturali, AI-IA-03, Pisa, Italy, 2003.

[2] D. Brainard, “Bayesian Method for Reconstructing Color Images from Trichromatic Samples”, Proceedings of the IS&T 47th Annual Meeting, Rochester, NY, pp. 375–380, 1994.

[3] L. Fletcher, N. Apostoloff, L. Petersson, A. Zelinsky, “Vision in and out of Vehicles,” *IEEE Intelligent Systems,* pp. 12–17, May-June 2003.

[4] R. Nevatia, “A Color Edge Detector and Its Use in Scene Segmentation,” IEEE Trans. on Systems, Man, and Cybernetics, Vol.7(11), pp. 820–826, 1977.

[5] Orsys Orth System GmbH, *Hardware Reference Guide micro-line C6713Compact High performance DSP / FPGA / IEEE 1394 board,* V1.1, www.orsys.de, 2003.

[6] C. Rasmussen, G. D. Hager, “An Adaptive Model for Tracking Objects by Color,” Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition, 1997.

[7] TTTech Computertechnik AG, *TTP PowerNode – A TTP Development Board for the Time-Triggered Architecture based on the AS8202NF TTP network controller,* V2.01, www.tttech.com, 2002.