A Motion Estimation System

Author

Kolodko, J, Vlacic, L

Published

2003

Conference Title

2003 IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, VOLS 1 AND 2

A Motion Estimation System

Julian KOLODKO[1], Ljubo VLACIC[2], Senior Member, IEEE

[1,2]Intelligent Control Systems Laboratory (ICSL), School of Microelectronic Engineering, Griffith University, Nathan, AUSTRALIA, e-mail:j.kolodko@sct.gu.edu.au[1], l.vlacic@griffith.edu.au[2]

***Abstract*— The ability to navigate intelligently among dynamic obstacles is critical for autonomous vehicles though the majority of navigation and sensing systems are not explicitly designed for dynamic environments. In our work we consider the design of a sensing system that is able to extract useful motion information using a combination of range and visual data. The generated information is analogous to the information generated by the short-range process of biological visual systems. In this paper we give an outline of our motion estimation algorithm, present preliminary simulation results and describe a hardware implementation of this algorithm.**

***Index Terms*—Motion Estimation, Multi-Sensor Fusion, Autonomous Mobile Robots.**

I. Introduction

The ability to navigate intelligently among dynamic obstacles is critical for autonomous vehicles and robots [1] though the majority of navigation and sensing systems do not explicitly consider dynamic environments [2]. When navigation in a dynamic environment is considered, it is often assumed that the state of the entire environment is known [3]. This assumption overlooks two key practical considerations. Firstly, from the frame of reference of an autonomous vehicle it is possible to estimate the state of only the part of the environment thas is visible to the vehicles sensors. Secondly, that estimate is limited by the accuracy and resolution of the sensors. Theoretically, sensors positioned to have an overview of the environment could provide a global state for the environment. However this violates the principal that an autonomous vehicle should be able to operate without explicit external assistance.

Our work considers the design of a sensor system (that is, a sensor and associated processing hardware) for determining the motion of objects from the frame of reference of an autonomous vehicle. This paper introduces our algorithms and gives preliminary simulation results for the algorithms before moving to a discussion of the architecture of our hardware implementation. The originality of this work lies in the combination of its hardware centric approach, fusion of visual and range data, its dynamic scale space approach and its focus on the use of motion as a primitive quantity.

A. What we are trying to achieve

Our goal is to implement a motion processing system for use as part of an autonomous navigation system. Using a simple geometric argument we found motion estimates must be generated at a rate of 12Hz to ensure temporal aliasing is avoided. Since the system will be used with relatively small autonomous robots, we aim to create a system with minimal physical size and power consumption.

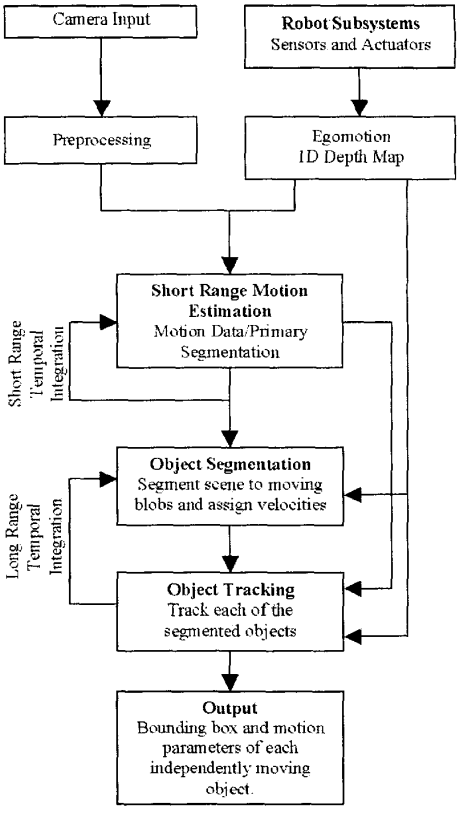

Fig. 1. Motion Processing Architecture Dataflow

B. Motion Processing Scheme

Figure 1 illustrates our overall motion processing scheme which is loosely based on evidence from biological visual systems [4]. Input for the system comes from a camera and a laser range scanner whose data is used to disambiguate visual motion. Processing begins with short-range motion estimation, so called because it uses only a small number of frames in reaching a solution. The result is a preliminary motion segmentation that estimates the position of motion edges and the direction of motion. The work presented in this paper focuses on creating this functional block. Each frames result is used at the subsequent frame to refine the result over time in a process known as temporal integration [5].

Depth, egomotion and tracking data are then used in conjunction with our preliminary segmentation to refine the seg-

0-7803-7912-8/03/$17.00 © 2003 IEEE

mentation and to assign a velocity to each blob. Individual blobs are tracked and this tracking information is used to implement long-range temporal integration, which uses many frames of data to assign a velocity to each object. Tracking data is extrapolated to provide an estimate of where a blob boundary will lie in the future and this can be used to refine the preliminary segmentation. Finally, tracking information (object location and velocity) passes onto the navigation subsystem on an autonomous vehicle to enhance its navigation capabilities.

The idea of segmenting the environment in terms of moving regions or "blobs" is core to our work. We consider motion as a primitive visual sensory modality like brightness, colour and stereopsis. That is, we attempt to break the environment down in terms of independently moving object fragments (i.e. moving blobs). No attempt is made to reconstruct the environment in terms of the objects that are moving. This approach is reasonable within the context of a collision avoidance system because it is not relevant if a vehicle collides with a particular moving object. What is important is that it does not collide with any "moving blobs". This approach also seems to be consistent with the early stages of motion processing in biological visual systems [4]. We focus on gradient-based techniques since they best correspond to the concept of motion as a primitive quantity and allow simple fusion of range and visual data.

II. Related Work

The majority of existing hardware motion processing systems follow a sense-digitize-process scheme where visual information is sensed by a camera, captured by a frame grabber then processed digitally. Such schemes can be implemented as a full custom design, using video processors or using a PC.

Motion processing in custom hardware involves design and implementation at the integrated circuit and circuit board levels. For example, Rowekamp [6] shows how a high resolution optical flow field can be calculated at very high speed (up to 1500Hz for a 128*128 vector field) using a digital ASIC implementation. Stoffler [7] uses MPEG correlation processors to compute and segment a sparse but robust optical flow of up to 525 vectors at 25 frames per second. Lee [8] uses a mixed signal VLSI neuroprocessor to compute 64 motion vectors and Boluda [9] uses an FPGA and a log polar vision system to identify objects that are moving independently of the camera at 285Hz.

Video processors are purpose built parallel computers or transputers. Examples of implementations using such technology include Mittal et al [10] who shows how time to collision can be computed at 25Hz allowing a robot to navigate without collision in a static environment. In a more complex system Willersin [11] uses a MiniVista transputer and a pair of PowerPC's to compute and segment optical flow at 12Hz. Szabo [12] presents a physically large system based on the DataCube platform. It has been used for a number of motion estimation tasks including independent motion estimation, gradient based optical flow and correlation based optical flow. The system can operate at up to 15Hz depending on the algorithm.

The simplest approach is to use a PC or DSP system. Such approaches are becoming more practical as computing power increases. Notable implementations include Benoit [13] who developed a correlation based algorithm that can produce motion estimates at 4Hz on a 100MHz processor. Batavia et al [14] show how optical flow can be used to monitor a vehicles blind spot at 15Hz using a Pentium based PC. Camus et al [15] compute flow divergence for both steering control and collision detection (with static obstacles) of a mobile robot. The entire implementation operates at 25Hz on an 80MHz Sparc machine. Krose et al [16] show how a mobile robot can navigate through the center of a corridor using the temporal derivatives of optical flow.

There are also a number of "non-traditional" hardware systems for motion processing that do not follow the normal sense-digitize-process pattern. For example, the Vision Chip [17] integrates sensing and processing elements into a single piece of silicon. Structured light systems [18] project a known pattern of laser light into the environment. Motion is then determined by monitoring the deformation of that pattern. And Houghton [19] shows how the speckle pattern that arises when a laser shines on an optically rough surface can be used to rapidly measure motion.

III. Algorithm

In this section we define what we mean by "real time" in the context of our work and show how a dynamic scale space is used to achieve real time processing. We then introduce our motion estimation algorithm that fuses visual and range data to eliminate motion ambiguity. Our algorithm separates motion estimation from motion segmentation and reduces what is traditionally an expensive 2D minimisation to a less costly 1D problem.

A. Real Time and Dynamic Scale-Space

Gradient based approaches such as ours fail if apparent motion greater than about one pixel per frame is present. Greater motion results in temporal aliasing and this in turn makes the image derivatives that we use to estimate motion invalid. Based on this, we define "real-time" as a rate quick enough to avoid temporal aliasing.

The usual method of resolving the problem of temporal aliasing is to use a scale-space approach where a pyramid of subsampled images is created. The motion estimation algorithm is then applied at each level of the pyramid. High velocities can be reliably measured at higher levels of the pyramid while the lower levels (i.e. those levels with less subsampling) are used to measure lower velocities. Unfortunately this scheme is problematic. Generating and storing an image pyramid is time consuming and propagating motion estimates from one level to the next adds significant complexity [5] making it extremely difficult to implement such a system in real time.

Rather than implementing a full scale-space scheme, we use a dynamic scale-space [20] where an appropriate scale is

chosen to avoid temporal aliasing based on range data. The nearest object is likely to have the highest apparent velocity, and it is the object with which we are most likely to collide with in the short term, so we choose a scale that prevents temporal aliasing for that object. A simple geometric [20] argument based on the pinhole camera model can be used to derive the following relationship between the required frame rate for real time processing, object velocity (V = 0.1m/s) and distance (D = 0.4m), and the camera focal length (f = 4.8mm) and pixel pitch (ζ = 12.5mm)

$$FrameRate = \frac{2Vf}{D\zeta} fps \qquad (1)$$

This is a worst case relationship based on a maximum relative velocity of 2V between the camera and an object. In our environment, with our camera a frame rate of 192fps is required to achieve real time if dynamic scale space is not used. With a 5 level dynamic scale-space, this falls to a much more attainable 12fps.

B. Motion Estimation

Our short range motion estimation algorithm uses the optical flow constraint equation (OFCE), eq 1 [21] together with the equations of motion [22] to fuse visual and range information [15]. This results in the following constraint equation.

$$E_d = -\frac{f\psi}{Z_{(x,y)}} \mathbf{U}_{\mathbf{X}_{(\mathbf{x},\mathbf{y})}} I_{x_{(x,y)}} + I_{t_{(x,y)}} = 0 \qquad (2)$$

In this equation, Ix and It are the horizontal and temporal derivatives of the image sequence, f is the camera focal length, ψ is a constant converting m/sec to pixel/frame and Z is depth. $\mathbf{U_X}$ is the lateral apparent velocity which corresponds to the speed (in m/sec) at which a point in the image appears to be moving. Since $\mathbf{U_X}$ does not in general correspond to an objects physical velocity its absolute value is irrelevant. This allows us to assume the scale factors f and ψ are equal to one, further simplifying our formulation.

The algorithm begins by solving for $\mathbf{U_X}$ at each pixel. To reduce the data stream we take the median estimate for each column. This reduces computational complexity to $\mathbf{O}(ix)$ (where i is the number of iterations, x is the width of the image) . Without this reduction computational complexity is $\mathbf{O}(ixy)$ (where y is the height of the image) which is typical of optical flow based methods. The net result is a robust motion estimate for each column which is then segmented using a first order weak string model [5][23] giving us the desired object boundaries.

The weak string results in a minimisation problem that is non-convex. To solve such a problem in using custom digital hardware deterministic algorithms are preferred to stochastic algorithms since they are more easily mapped to digital logic. We use the Graduated Non-Convexity method [23] together with Successive Overrelaxation to find the global minimum.

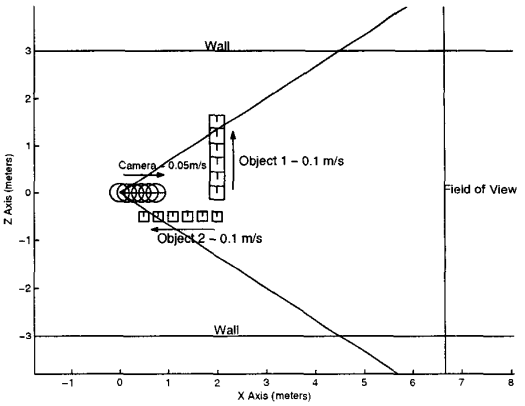

Fig. 2. Experimental Scenario

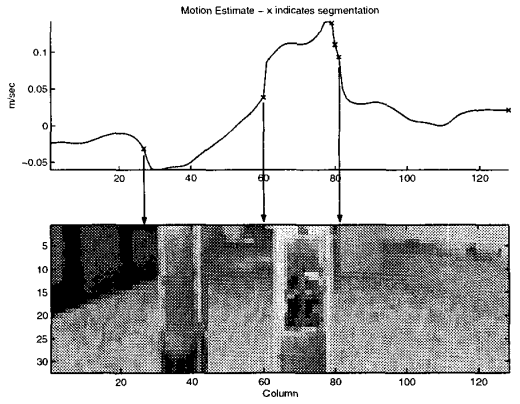

Fig. 3. Motion Processing Result

Implicit in this model is the assumption that the world is piecewise planar with the planes parallel to the cameras sensing surface. The reason is threefold. Firstly, by using image columns as regions of support for the motion estimate we have assumed that each pixel in the column is part of the same object. To minimise the chance of this assumption being violated we use a robust estimate for the motion of each column and we use a relatively narrow image. Secondly, the weak string model allows segmentation of motion effectively allowing piecewise planar world. Finally, because only a single dimensional range scan is available, we assume that range value is constant over the entire image column, hence the planes are parallel to the cameras sensing surface.

C. Algorithm Simulation

In order to verify and tune our algorithm we developed a simulation system based on MATLAB and the freeware raytracer Polyray [20]. This simulation system generates visual and range data corresponding to a user specified environment. The system also generates ground truth motion data which is valuable for testing motion estimation results.

Figure 2 gives illustrates a scenario we used to test the operation of our algorithm. In this example, the camera is

moving forward at 0.05m/s and the two visible objects both have a velocity of 0.1m/s. The "central" object is moving to the right and left hand object is moving parallel to the camera. Only 10 iterations of the algorithm are used and all detected discontinuities are marked with a cross. Figure 3 illustrates the result. Notice that both edges of the central object have been well detected though only one edge of the left object is detected because the first order weak string can only detect step discontinuities - it is unable to detect the "crease" discontinuity [23] that wee see here. The spurious discontinuities for the central object are related to parameter tuning though over-segmentation is not necessarily a problem. The discontinuities are reasonably places (i.e. they are clustered around an object edge) and a complete motion estimation system would be expected to filter out spurious discontinuities at the long range processing stage (fig 1).

IV. Hardware Implementation

We have chosen to develop our system using an FGPA as the primary processing element. This allows us to implement glue logic and processing in a single device so that the final implementation fulfils our goals of a compact sensing system. In this section a brief introduction to the core components of our system is given before some detail of our implementation.

A. Hardware Platform

In order to accelerate prototype development we have opted for a commercial prototype development platform created by Lyr Signal Processing. The SignalMaster/GatesMaster platform provides a Virtex XVC800 FPGA, 16MB of SDRAM and 140 digital IOs [24]. Further processing can be performed using the onboard Analog Devices Sharc DSP. The system also includes a wealth of interfacing options including PC104, USB and Ethernet.

B. Fuga 15D Camera

We use the Fuga 15D CMOS camera from C-Cam Technologies because its RAM-like interface eases development. The Fuga15D has a logarithmic intensity response making it less sensitive to illumination variation however it has relatively high noise levels. Fortunately this noise pattern is fixed and can be minimized using a zero order calibration. Another issue with the Fuga camera is that it does not use a shutter. A pixels value is determined at the moment it is addressed while traditional camera effectively measure the value of all pixels at the same time (i.e. when the shutter closes). Thus if an image frame is not read quickly, it may contain motion distortion appearing as shear in the shape of moving objects. We have found that motion distortion is negligible in our test environment.

C. ICSL Vehicle Testbeds

Our system is designed to operate within the parameters of our ICSL Robots [25]. The ICSL developed a number of vehicle test-beds for use in evaluating autonomous vehicle concepts without the need for large scale test facilities. These robots feature a distributed multi-microcontroller architecture with the PIC 16C74 microcontroller as the primary building element. Subsystems include infrared and ultrasonic ranging systems for navigation, radio packet modem for communication and a laser based system for intelligent speed adaption. A number of behaviors have been developed for these test-beds including fuzzy logic based leader following, static obstacle avoidance, lane keeping, intersection navigation and overtaking. Maximum velocity for the robots is 0.1m/s.

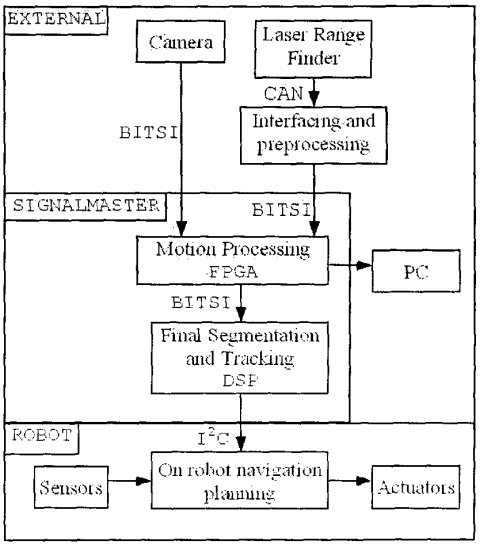

Fig. 4. Mapping architecture to hardware

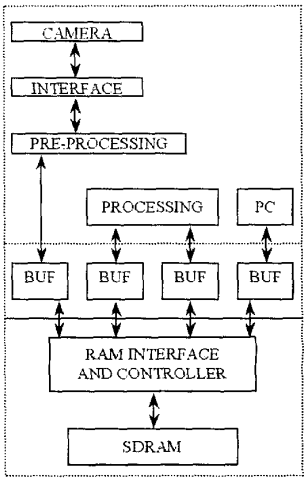

Fig. 5. Implementation Outline

D. Implementation

Biological motion processing has inspired our mapping from motion processing architecture (figure 1) to hardware (figure 4). It appears that biological systems have a short range motion processing system that is implemented in a

highly parallel manner while the long range system is serial in the sense that it depends on where attention is focused [26][4]. Following this scheme we could theoretically implement multiple processing nodes in FPGA for short range processing. Indeed, our algorithm supports SIMD implementation. Long range processing would then be implemented serially in DSP. Unfortunately, resource limits prevent such an implementation however we maintain the overall mapping since it fits well with our prototyping platform. Advantages of this mapping include the ability to implement glue logic in FPGA (reducing the need for external components) and the placement of the most intensive processing task (short range motion estimation) into the device best able to achieve processing goals (the FPGA).

Camera data is fed into the FPGA directly via the BITSI interface while information from the laser range finder is preprocessed externally (e.g. polar to rectangular coordinate conversion, dynamic scale space support calculations, interpolation to camera coordinate system). Short range motion processing is implemented in FPGA and the remainder of this section focuses on the structures required for a successful implementation of this algorithm. The resulting segmentation is passed to both a PC for visualization and onto further processing in DSP. Object location and velocity data is then passed onto the ICSL autonomous testbeds via an I2C interface.

As shown in figure 5, the FPGA component of this implementation has three primary components; the memory subsystem, a collection of processes (i.e. reading from camera, processing data, output to PC etc) and buffers designed to mediate memory access.

E. RAM Interface and Memory Management

Because a number of processes may require simultaneous access to RAM, some form of memory management is required to control contention. This is achieved via a combination of decoupling buffers and the RAM Interface and Controller (RAMIC) Module. Each process has a decoupling buffer implemented using Block SelectRAM within the FPGA. These buffers allow each process to operate at full speed without waiting on memory.

Aside from providing a low level interface to SDRAM (including address unpacking), the RAMIC module has two core functions. First, it provides RAM bus arbitration. This is implemented as a round robin polling scheme which provides an absolute upper bound on memory access time hence we are able to prevent buffer under/overruns. Secondly, the RAMIC implements a number of compound memory operations. For example, we exploit the difference in data width between the camera (8 bit) and RAM (32 bit) to optimize memory access. Because the motion processing algorithm requires three frames of data when computing derivatives, the RAMIC stacks data corresponding to a particular pixel at a single RAM address. In this way, a single read operation will return the required 3 frames of data. This is effective because, during processing, there is a significantly higher reading bandwidth (reading 2D image + range data) than writing bandwidth (writing 1D segmentation result).

F. Buffers

The Virtex FPGA has dedicated RAM (known as Block SelectRAM) available on board. Twenty eight blocks of SelectRAM are available each with 4096 bits. Input and output data widths are user selectable. This RAM is dual ported so our buffers are implemented to allow simultaneous access by a process and by the RAMIC. The buffer controllers provide handshaking controls, generate addresses as necessary, manage buffer pointers and may perform some simple processing. Buffer under/overruns are unlikely, however if they do occur data is lost. We do not wait for buffers to clear since this can cause timing problems with the shutterless FUGA camera.

G. Camera and PC Interface

The camera interface block generates the control signals necessary to read a 512*32 pixel image from the camera. We use a 1MHz pixel clock so each frame is read in approximately 16.3ms, more than fast enough to prevent motion distortion in our test environment. A separate frame clock maintains a 12Hz processing rate. Before being sent to RAM, raw camera data has a zero order calibration applied and is subsampled appropriately for the current dynamic scale space. While new data is being read from the camera, older data is sent to the PC. Here timing is strictly controlled to prevent new data clobbering older data before it is sent to the PC. Before streaming of a frame begins, the PC is sent a frame trigger signal (implemented using the PC parallel port) that synchronizes the PC and FPGA and maintains a 12Hz frame rate on the PC. We implement a "virtual camera" in the FPGA to interface our system to the capture card supplied with the FUGA camera.

H. Timing and Resource Use

Our current implementation is complete except for the processing block and range scanner interface. This implementation has utilized approximately 10% of available logic resources, 35 digital IO's, 8 blocks of SelectRAM. Our implementation allows IO operations for the camera, PC and the laser range finder to be interleaved, however data processing must occur independently. Since I/O operations require 22.5ms per frame, we have 77.5ms available for processing giving a clock cycle budget of 19 clocks per pixel per iteration under the assumption of 512*32 pixel images and 10 iterations per frame. This is more than sufficient for our algorithm.

V. SUMMARY

In this presentation we have given an overview of our motion processing system and in particular, our short range motion processing scheme. Simulation examples show that this scheme functions well in our test environment though further algorithm tuning is necessary. We then showed how this algorithm is being implemented in semi-custom hard-

ware. Our combination of a novel motion processing algorithm and semi-custom hardware ensures motion data is retrieved in real time making the systems a useful component of an autonomous navigation system.

VI. Acknowledgements

We would like to thank Frauenhofer Autonomous Intelligent Systems Group (AiS) for their generous donation of the Signal Master platform and Fuga Camera used in this work.

References

[1] Laugier C., Fraichard T., "Motion Planning for Car Like Vehicles", Chapter 11.5, Intelligent Vehicle Technologies: Theory and Applications, Vlacic L. Parent M., Harashima F. (eds), Butterworth Heinemann, 2001.

[2] Latombe, J. C., "Robot motion Planning", Kluwer Academic Press, 1991

[3] Fiorini P., "Robot Motion Planning Among Moving Obstacles", Ph.D. Thesis, University of California, 1995.

[4] K. Nakayama, "Biological Image Motion Processing: A Review", In Vision Res, Vol 25(5), pp625-660, 1985

[5] M. J. Black, "Robust Incremental Optical Flow", Ph.D. Thesis, Yale, 1992

[6] T. Rowekamp, "A Smart Sensor System for Real-Time Optical Flow Estimation", Ph.D. Thesis, Technischen Universität Cottbus, 1997

[7] N. O. Stoffler, Z. Schnep, "An MPEG-Processor-Based Robot Vision System for Real-Time Detection of Moving Objects by a Moving Observer", http://www.lpr.ei.tum.de/ stoffler

[8] J-C. Lee J-C, B. J. Sheu, W-C. Fang, R. Chellappa, "VLSI Neuroprocessor for Video Motion Detection", IEEE. Trans. On Neural Networks, vol 4(2), pp 178-191, March 1993

[9] J. A. Boluda J. A, F. J. Blasco, F. Pardo F., "A Scalable Reconfi gurable FPGA-Based Architecture for Robotic Navigation"

[10] A. Mittal, A. Valilaya, S. Banerjee, M. Balakrishnan, "Real Time Vision System for Collision Detection"

[11] D. Willersinn, W. Enkelmann, "Robust Obstacle Detection and Tracking by Motion Analysis", IEEE Conference on Intelligent Transportation Systems, pp 717-722, 1997.

[12] S. Szabo, D. Coombs, M. Herman, T. Camus, H. Liu, "A Real-Time Computer Vision Platform for Mobile Robot Applications", Journal of Real-Time Imaging, Academic Press, Vol. 2, No. 5, pp. 315-327, October 1996

[13] S. M. Benoit, "Monocular Optical Flow For Real-Time Vision Systems", Ms Thesis, Dept. Elect. Eng., Center for Intelligent Machines, McGill University, Canada, 1999.

[14] P. H. Batavia, "Overtaking Vehicle Detection Using Implicit Optical Flow", IEEE Conference on Intelligent Transport Systems, pp 729-734, 1997

[15] T. Camus T, D. Coombs, M. Herman, T-H. Hong, "Real-Time Single-Workstation Obstacle Avoidance using only Wide-Field Flow Divergence", 13th Int. Conf. Pattern Recognition - Applications and Robotic Systems, August 1996

[16] B. Krose B. A. Dev, X. Benavent, F. Groen, "Vehicle Navigation on optic flow"

[17] A Moini, "Vision Chips or Seeing Silicon", Centre for High Performance Integrated Technologies and Systems, Department of Electrical and Electronics Engineering, The Univ. of Adelaide, Australia. Website http://aaron.eleceng.adelaide.edu.au/Groups/Bugeye/visionchips/,

[18] D. Monteiro,"Visual Servoing for Fast Mobile Robot: Adaptive Estimation of Kinematic Parameters", Proceedings of IECON '93, vol 3, pp 1588-1593, 1993

[19] A. Houghton, G. Rees, P. Ivey, "A method for processing laser speckle images to extract high-resolution motion", Measurement Science and Technology, Vol 8(6), pp 611-617, Jun 1997

[20] J. Kolodko, L. Vlacic, "From Motion Processing to Autonomous Navigation", To appear in Intelligent Robots: Vision, Learning and Interaction, KIAST Press, 2003

[21] B. Horn, N. Schunk, "Determining Optical Flow", In MIT Artifi cial Intelligence Lab, AI Memo 572, April 1980

[22] A. Mitiche, "Computational Analysis of Visual Motion", New York, Plenum Press, ISBN: 0-306-44786-X, 1994

[23] A. Blake, A. Zisserman, "Visual Reconstruction", MIT Press, 1987, ISBN 0-262-02271-0

[24] Lyr Signal Processing, http://www.signal-lsp.com/

[25] M. Hitchings, L. Vlacic, Z. O'Sullivan, "Fuzzy Control of Distance and Motion Tracking Applied to Cooperative Autonomous Robots" Proc of Distributed Autonomous Robotic Systems Conference (DARS 2000), pp.335-345, 2000

[26] F. Crick, "The Astonishing Hypothesis", Touchstone Books, Simon and Schuster Ltd, London, 1995