Based on the paper in *Proceedings of the IEEE Computer Society Workshop on Embedded Computer Vision, Conference on Computer Vision and Pattern Recognition*, (New York, NY), June 2006.

The Tyzx DeepSea G2 Vision System, A Taskable, Embedded Stereo Camera

John Iselin Woodfill, Gaile Gordon, Dave Jurasek, Terrance Brown, and Ron Buck
Tyzx, Inc.,
3895 Bohannon Dr.
Menlo Park, CA 04025
woodfill@tyzx.com

Abstract

Our goal is to build vision systems suitable for deployment in devices that operate in demanding dynamic, variably lit, real-world environments. For such systems to be successful, not only must they perform their visual analysis well and robustly, but they must also be small, cheap and consume little power. Further, since volume deployments of such vision systems are still nascent, the systems must be taskable -- flexible enough to support many different uses.

We have met our goal with the Tyzx DeepSea G2 Stereo Vision System, an embedded stereo camera consisting of two CMOS imagers, a DeepSea II stereo ASIC, an FPGA, a DSP/Co-processor and a PowerPC running Linux, connected to the Ethernet. It is made practically taskable by the definition of a set of configurable visual primitives supported by specific hardware acceleration. These primitives include stereo correlation, color and depth background modeling, and 2D and 3D quantized representations or projections of the range data. We have defined a common programming interface in which the visual primitives are available both in traditional workstation environments, supported in software, and on the G2 with hardware acceleration. Single G2s are deployed in mobile platforms such as robots and automobiles, while networks of G2s are deployed in tracking systems in public and private sites around the world.

1. Introduction

Large classes of commercial and consumer products have a requirement for visually sensing their environment and reacting to what is happening around them in real time. These products are deployed in demanding dynamic, variably lit, real-world environments. For such systems to be successful, not only must they perform their visual analysis robustly, which often requires 3D data and high frame rates, but they must also be physically small, low cost, and low power. A vision application running on a powerful workstation is not a viable solution for these products. Since volume deployment for embedded vision systems are still nascent, and engineering costs high, we argue that a practical embedded vision system product must also be flexible enough to support many different applications. We have designed a new 3D embedded vision system, the Tyzx DeepSea G2 Stereo Vision System, to meet these demanding requirements; and it has already been deployed in multiple application spaces.

1.1. The Case for 3D Embedded Vision Systems

It is critical to base design requirements on several real applications. We specifically consider distributed person tracking for security applications, and autonomous navigation.

3D sensing is an important modality for a vision platform. Many vision applications have traditionally relied only on 2D data, which is inherently sensitive to changing lighting, and requires computationally intensive algorithms for analysis of application relevant outputs. Now real time 3D sensors can be incorporated into vision systems to simplify data analysis [1,6]. These sensors make some applications, such as tracking of objects for security systems more robust and tractable. For other applications, such as navigation, 3D descriptions of the scene are critical. For example, potential obstacles must be evaluated to determine whether their 3D location, size, and trajectory relative to the vehicle path represent a threat to the vehicle.

High frame rate and low latency are critical factors for many applications which must provide quick decisions based on events in the scene. Tracking moving objects

from frame to frame is simpler at higher frame rates because relative motion is smaller, creating less tracking ambiguity. In autonomous navigation applications, vehicle speed is limited by the speed of sensors used to detect moving obstacles. In safety applications such as airbag deployment, the 3D position of vehicle occupants must be understood to determine whether an airbag can be safely deployed – a decision that must be made within tens of milliseconds.

Bandwidth limitations are also an important consideration in the design of vision systems. At high frame rates it becomes impractical to pass 2D or 3D image data out of the sensing system, particularly when several sensors are used together in the same application, such as distributed tracking systems. The processing of the image data must instead be done close to the sensors, passing only lower bandwidth application specific data on to other subsystems. Consider, for instance, a wide area tracking system based on stereo imagers (one color and one monochrome imager, producing 26 bits (10 bit Y, 8 bit U and V) and 10 bits per pixel respectively). For 640x480x30fps, we have ~330 Mbits of data per second coming out of the imagers. If we feed this data into a stereo processor and down-sample the color image we might end up with as little as 200 Mbits of data per second. If we actually process the range and color data to detect and locate people, we can reduce the information to a few bytes per person tracked per frame – perhaps 80 Kbits per second. For just one tracking camera, the bandwidth of data at any stage in the pipeline could be supported on the network. However, for a hundred tracking cameras, the left and right source data becomes 33 Gbits per second, the color and range 20 Gbits, and the segmented track data might be about 8 Mbits per second. The only practical choice is for all the processing to happen before results are sent out of the vision system.

For self contained vision systems to be actually deployed on ceilings, in cars, or on robots they must be small enough to fit in unused spaces. They cannot have fans or generate too much heat. For battery powered systems, the power usage must be limited. These footprint and power requirements are the most basic system requirements and dictate that workstations or typical power hungry CPU's of any kind cannot be used.

Since stereo processing involves a large computational load that can still consume a high end CPU, it makes sense to use smaller, lower power dedicated hardware resources such as the Tyzx DeepSea II chip to perform stereo correlation.

For obstacle detection or tracking, even after range is produced by special purpose hardware, processing range and color images at frame rates can be a demanding computational challenge -- particularly for the sort of CPU suitable for an embedded system. Our approach to the problem of the vision computational load is to decompose the large computational task into primitives, visual primitives, that can be exposed in a software interface, and that can be efficiently implemented with hardware acceleration. Stereo correlation itself can be viewed as such a primitive. Other visual primitives that we have identified include range and intensity/color based background modeling [2,4], and projection of the 3D data into a Euclidean 3D quantized volume, or a 2D quantized array.

1.2. Related Work

Previous examples of 3D embedded vision systems exist that were not very powerful or did not meet power requirements for real deployment. Konolige [3] presented the SVM I, a DSP based embedded stereo camera that processes small stereo frames at 12 frames per second. Little CPU time was left to perform any interpretation task using the data. Woodfill et al. [6] describe an embedded stereo camera consisting of an embedded Pentium Mother board along with a Tyzx DeepSea II processing board. This two board solution with a Pentium presented a fairly powerful computational environment, particularly since range was computed on special hardware. However, power consumption and footprint were still high.

Stein et al. [5] present the EyeQ, a single chip, monoscopic embedded vision system designed for a set of automotive applications. This system has a couple of general purpose PowerPC engines and specialized engines for automotive specific tasks such as lane detection. This is one of the few low power embedded visions systems built, but does 2D rather than 3D processing, and is dedicated to a particular application.

1.3. Outline

In this paper, we present the G2, a powerful, multi-purpose embedded stereo camera. First we describe the structure of the G2, followed by a description of the various visual primitives for which we have provided hardware acceleration. Given a description of the system and its accelerated visual primitives, we illustrate how two applications can be mapped to the G2 architecture. Finally we conclude with some performance numbers, some lessons learned and thoughts about future work.

2. The Structure of the G2

The G2 is a PowerPC based computer that runs Linux. At the component level (See Figure 2), the G2 consists of an embedded PowerPC chip (666 MHz 440GX) connected to two imagers, a Tyzx DeepSea II chip, an Analog Devices Blackfin DSP, and some memories using an FPGA. This basic configuration is extremely flexible – any component can be configured to talk to any other component by means of FPGA configuration. Implementing a new configuration for the FPGA, in

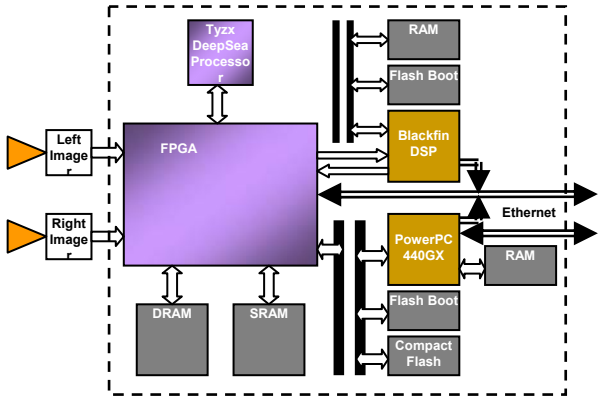

Figure 1. G2 hardware layout

principle a simple task, can take weeks or months of effort. Thus the G2 is designed with a particular dataflow architecture / FPGA configuration in mind that allows the various components to be reconfigured at a course level of granularity. This approach allows the system to be applied to distinct tasks without modifying the firmware in the FPGA.

This dataflow architecture is shown in Figure 3. The basic dataflow model is a generalized form of the model that applies to any computer that is intended to perform vision tasks: on a frame by frame basis the CPU has access to input imagery. Left and right rectified source imagery can be DMAed into main memory on the PowerPC. In principal this data is enough to perform any vision or stereo based task. However, the cost of computing stereo depth at frame rate alone would swamp the embedded PowerPC. To perform real-time vision tasks, the PowerPC has access to several other input image sources on each frame: a range image, a foreground image, a "Projection image" and some output from a DSP that in turn has access to the source images, range image, foreground image and "Projection image".

To configure the G2 for a particular task, the user selects which vision primitives are to be used (see following section), what parameters to use for the various vision primitives, which image data should be sent to the DSP, and which image data should be DMAed into PowerPC memory. The programming task is completed by writing code to interpret input images on the PowerPC and/or the DSP if necessary.

The G2 consumes about 15 Watts. It has a 100Base T Ethernet interface that can accept Power-over-Ethernet. A Linux kernel and root file-system are stored on the Compact Flash memory card so that the system can boot into Linux on power-up.

As a matter of development convenience, we have designed a common software interface that is available both on Windows and Linux workstations as well as on

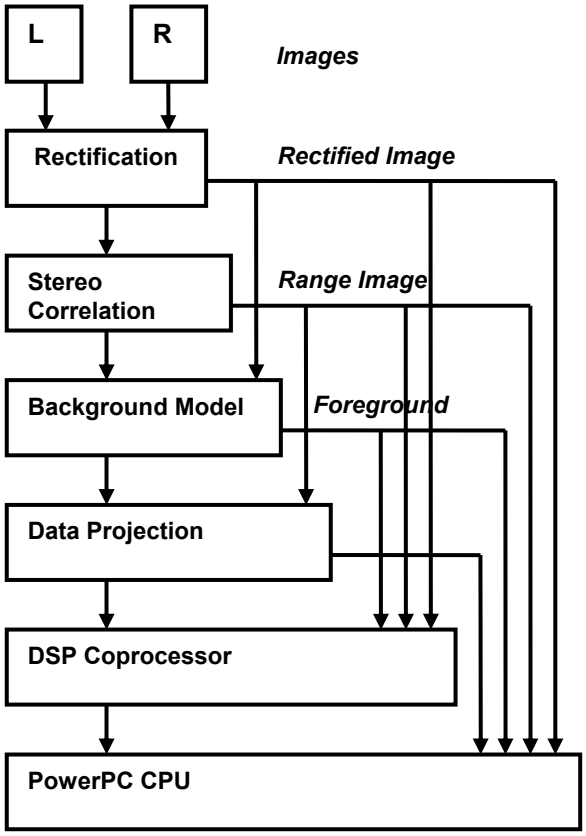

Figure 2. G2 data flow

the G2. On the G2, many of the visual primitives are accelerated in the FPGA and DSP, while many are pure software on a workstation.

As the G2 has a general purpose CPU running Linux, it is possible to edit, compile, and debug code from a telnet window running on the embedded CPU. Since we have a common software interface shared between the workstation and the G2, it is usually easier to develop and debug code on the workstation, cross compile the code for the PowerPC on the workstation, test the compiled code on the G2, using NFS, and finally transfer the debugged executables onto the G2 compact flash.

3. Hardware Accelerated Visual Primitives

The idea of hardware accelerated visual primitives is like that behind the primitives in OpenGL – they represent a decomposition of the computational problem. The primitives in this decomposition are chosen to meet three constraints: that they be useful for many tasks, that the primitives account for a large part of the computational load, and that they be suitable for hardware acceleration. In OpenGL, these primitives are defined with respect to a graphical display, in the G2, the primitives are defined with respect to a left and right incoming image stream, as well as outputs of other image primitives.

As the G2 is an embedded stereo camera, the visual primitives are aimed at producing and interpreting range data. Since stereo correlation is both a computationally intensive task, and a common sub-component of vision algorithms, it is an obvious visual primitive to support. Background modeling based upon range and color requires huge amounts of memory bandwidth and finds a place in many tracking algorithms – hence it is our second visual primitive. A third primitive produces 2D and 3D quantized representations or projections of the range data. This is an expensive, yet commonly useful visual primitive. Lastly, a programmable DSP is included in the G2 as a generalized visual primitive -- an additional resource for doing expensive, regular image operations.

Tyzx DeepSea II Stereo Correlation Chip	
Input Image Size (max)	512 x 2048 (10 bit)
Stereo Range	
Search Window	52 Disparities
Sub-pixel Localization	5 Bits
Z output	16 Bits
Max Frame Rate	200 fps (512x480)
Power	< 1 watt
Pixel Disparities/ Second	2.6 Billion

Figure 3. Tyzx DeepSea II specifications

3.1 Stereo Processing

In the G2, the primary visual primitive is the Stereo Correlation Processor. It takes as input the left and right images, and creates as output a range image. In the software model, the input images are implicit, just as the display is implicit in OpenGL. In the G2, the Stereo Correlation Processor is accelerated using a Tyzx DeepSea II correlation chip. The performance of this chip is characterized in Figure 4 [6].

3.3 Background Modeling

Background Modeling takes as input range and intensity data and generates a pixel-by-pixel foreground/background map of the image based on the nature of the pixel's change from previous frames. This task may require roughly 400 bits per pixel to be read from memory, and roughly 200 bits per pixel be written to memory. For 400 x 300 images at 30 frames per second, background modeling requires over 2 Gbits of memory bandwidth per second. In addition, matching and updating each pixel involves several comparisons and updates. Offloading this task to dedicated hardware reduces the workload of the CPU and is a key factor enabling the use of a smaller, embedded CPU. This background modeling visual primitive implements an algorithm similar to that presented by Gordon et al. [2] that is designed to model the scene statistics using both color intensity and dense range. The output is a binary foreground mask image.

3.4 Projection Spaces

Projection of the 3D data into a Euclidean 3D quantized volume, or a 2D quantized array produces a data representation that is useful for many applications and is more compact that a full 3D cloud of points. The Projection Space primitive provides a mechanism to define and create 2D and 3D projections where clipping and quantization parameters can be defined independently for each of the three axes. A rigid transform (in terms of a rotation and translation) can also be specified that is applied to the X,Y,Z data points before the projection.

The Projection Space visual primitive takes as input the range image and, if desired, the foreground mask created by the Background Model primitive. If the foreground mask is used, only the foreground pixels will be processed into the Projection Space. The coordinates of a raw range pixel are in terms of the row (v) and column (u) of the range image, whereas Z is represented in metric units chosen by the user. Each range pixel (u, v, Z) is first converted to Euclidean coordinates, (X, Y, Z), using the camera calibration parameters such that the units of X and Y are the same as the units of Z. A 3x3 rigid transform is then applied to produce (X', Y', Z'). The definition of the projection axis is used to map (X', Y', Z') into the projection space. Each pixel will either fall outside the volume, or into one of the defined projection cells. For each cell defined, the final output is a total count of pixels

falling into the cell, and if using a 2D projection array, the minimum and maximum value on the projection axis of all pixels mapped to the cell.

There is also a mode which enables scaling of the counts in the projection space cells such that they are approximately invariant to distance. When using raw pixel counts, a closer object will generate higher cell counts than a farther object even if they are the same physical size. This is because at closer distances each pixel subtends a smaller surface area. The scaled mode will compensate for this effect, making it easier to process the resulting projection array to identify objects relevant to the application.

3.5 DSP/Co-processor

A fourth, more general visual primitive is provided by a somewhat general purpose DSP or Co-processor. DSPs are ideal for very regular image operations with specialized memory maintenance operations, zero overhead loops, etc. In the G2 dataflow, the DSP is viewed as another hardware accelerator that has more user configurability than the other primitives, i.e., it can be programmed. But like the other visual primitives, its control flow is fixed. A certain set of input images is selected, the DSP processes the input images each frame, and sends an output frame to the PowerPC.

4. Installed Applications

The low power and small footprint of the G2 embedded vision system enables the use of powerful real time 3D vision processing in applications and products that cannot support the use of traditional general purpose workstations. Two excellent examples are person tracking systems, which require networks of smart visual sensors to be mounted on walls and ceilings, and visual navigation for autonomous robotic platforms, which are battery powered and are often physically small. The vision primitives for the G2 platform support the processing requirements of both these applications. G2 platforms are already in use in installed person tracking systems and autonomous navigation systems. We describe these installed systems in more detail in the following sections.

4.1. Person Tracking

The value of real time stereo sensors for person tracking applications has been described previously [6]. The direct measurement of the 3D location of each person creates more robust results than systems based on 2D images alone. Fast frame rates simplify the matching of each person's location from frame to frame. The fact that each stereo camera is already calibrated to produce absolute 3D measurements also greatly simplifies the process of registering the cameras to each other and the world during installation.

To be practical for real installations, however, factors such as network bandwidth, power requirements, wiring requirements, and physical size were all key design criteria for the G2 platform. Limited network bandwidth in large installations dictates that most image processing must occur next to the sensor, sending only low bandwidth results such as locations and descriptions of tracked objects, over the network. Each sensor must also host enough processing power to perform tracking in its view space. Each system must be unobtrusive – fans are not practical. This places a limit on the power dissipation. The system must also be small enough to mount easily to the ceiling or wall. Wiring should be minimized; the G2 is designed to run with a single cable, power over Ethernet, or two connections, Ethernet and power.

Figure 4. Electroland tracking application on the observation deck of the Rockefeller Center.

3D person tracking is a computationally intense task, but the use of hardware accelerated visual primitives offloads most of the processing from the G2 CPU. The 3D range image and intensity image are used by the background model primitive to filter the incoming data to keep only pixels that are dynamic, that is, that do not match well to the typical background data. The remaining data, expressed as a binary foreground mask, is processed

by the Projection Space visual primitive. In this step, the orientation and position of the camera, computed during registration, are used to transform each 3D foreground pixel into a coordinate system aligned with the floor. The data is then projected into quantized cells on the floor plane. In this 2D representation, objects with a high surface area perpendicular to the floor stand out and people are maximally separated. Segmented and tracked results from many tracking camera are sent to a single system to produce a final tracking map.

Our partner, Electroland, has used a network of four G2s running our person tracking application in their interactive architectural installation created for Target Corporation in the new observation deck of Rockefeller Center, NYC, USA. Individuals who enter the space are assigned a personality by the 3-D tracking system and have individualized light patterns follow them around the space. This installation represents a challenging environment for vision based tracking systems because of the constantly changing lighting. The walls and ceiling are covered with dynamic colored light displays and ambient light is highly impacted by two sets of glass doors opening onto the observation deck, as well as an entire glass wall through which a lower level of the building is visible. The installation is active during daylight and nighttime conditions. A traditional 2D visual tracking system, which would incorporate a background model based on reflected light intensities, would not be an effective solution for this environment. Direct stereo measurements of depth on the walls and floor will, however, be robust and remain constant despite the dynamic lighting conditions.

4.2. Autonomous Navigation

Obstacle detection and other navigation tasks are facilitated by 3D data of the scene in front of the vehicle. Laser scanners have been commonly used to provide this data, but high frame rate dense stereo data platforms such as the G2 are now being considered as an alternative or additional sensor. The G2 provides a full frame of 3D data at each update rather than one scan line, which is important in detecting moving obstacles. The G2 can run at higher frame rates. Although adding scene illumination is always an option, stereo is inherently a passive sensor, which is desirable in many applications. The G2, whether located with the stereo imagers or separately, has a smaller footprint than a laser scanner, which for many platforms is a critical design feature. The G2 platform also has much lower power consumption and volume cost. The G2 also can handle much of the processing of the 3D data, offloading many computationally consuming tasks from the other processors on the vehicle.

As with most vision applications, hardware acceleration of as much of the vision processing task as possible is a big win for frame rates and power consumption. The projection space primitive was designed to provide a representation of the scene that is useful in obstacle detection and other navigation tasks. The rigid transformation can be used to put the data into approximate alignment with the road – compensating for installed camera rotation and translation. The data can then be projected onto the plane. This will produce high values when there are objects with sufficient surface area perpendicular to the road and 2D quantized images are much more efficiently processed than a full 3D cloud of points. The G2 CPU is typically able to run the full object

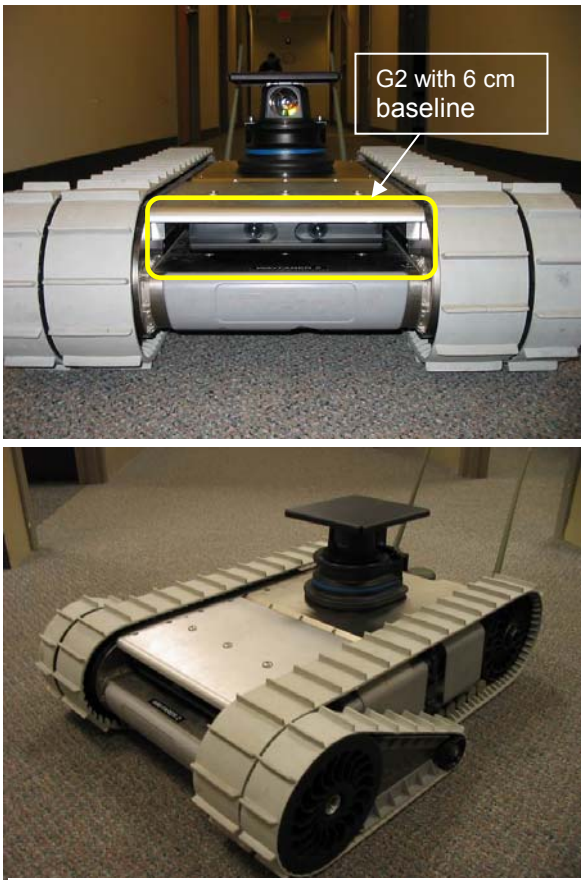

Figure 5. iRobot Wayfarer: Autonomous robot navigation application.

detection and path planning algorithm, providing only high level data to the other vehicle processors.

Another of our partners, iRobot, is developing a fully-autonomous urban reconnaissance platform[7]. The image in Figure 6 shows a G2 system installed on a compact agile robotic vehicle. The G2 is used to run autonomous obstacle detection and path planning algorithms. The small size of the payload bay of the vehicle represents a real limitation on the types of sensors

that it can deploy. The G2 is a perfect solution for providing a powerful 3D vision platform in a small footprint. The computing resources available on the G2 also offload the robot's limited main CPU, enabling the computation for obstacle detection to run directly on the G2. The separation between the lenses of this G2 is 6 cm. The configuration is used to monitor the space from .5 to 20 meters in front of the vehicle.

Acknowledgements:
Tyzx would like to thank Damon Sealy at Electroland and Brian Yamauchi at iRobot for providing pictures used in this paper. The iRobot Wayfarer project was funded by the U.S. Army Tank-automotive and Armaments Command (TACOM) Tank-Automotive Research, Development, and Engineering Center (TARDEC).

5. Conclusion

An embedded stereo vision platform with a well designed decomposition of hardware accelerated visual primitives can be extraordinarily useful. Several applications have been ported to the G2, showing that the resulting system is indeed practically taskable.

The performance of the G2 can be characterized by its frame rate for the whole datapath as described. At 30 frames per second, for 400x300 images, images are fed into the range processor, range results and rectified images are fed into background modeling, the results of background modeling can be fed into the projection primitive; finally, color, range, and projection results are DMAed into the PowerPC.

In developing the G2, several of our most basic fears were proved well-founded. You don't "just recompile C++ into Verilog" The process is a lot more than a push of a button. Floating point must be converted to fixed-point, conditional loops need to be converted to operations that can be performed everywhere. Correct Verilog is of course, not sufficient; everything must be shoehorned into running at the desired clock frequency. Similarly, you don't "just recompile your C++ for a DSP" – to get the desired performance boost, the code must be redesigned and/or massaged to run well on the DSP. You don't "just recompile Linux for a new platform." You probably need a new boot loader for the system. Any special purpose drivers must be done again, etc.

In the future, we anticipate identifying additional visual primitives and accelerating them in hardware, making the G2 applicable for a wider range of applications. In addition, as we begin to understand the power of the visual primitives, and as we get closer to high volume applications, we will pursue more integration, developing new ASICs to attain our ultimate goals – visually competent, smaller, cheaper, and lower power systems that can be more easily deployed in the world.

References

[1] S. Gőktűrk, H. Yalcin, and C. Bamji. A Time-Of-Flight Depth Sensor - System Description, Issues and Solutions. In Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition, Washington D.C., USA, June 2004.

[2] G. Gordon, T. Darrell, M. Harville, J. Woodfill, Background Estimation and Removal Based on Range and Color, In Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition, (Fort Collins, CO), June 1999.

[3] K. Konolige, Small vision sytem: Hardware and implementation, In Proceedings of Eighth International Symposium on Robotics Research, Hayama, Japan. (1997)

[4] C. Stauffer and W. Grimson, Adaptive background mixture models for real-time tracking. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, pages 246--252, 1999.

[5] G. Stein, E. Rushinek, G. Hayun, A. Shashua, A Computer Vision System on a Chip: a case study from the automotive domain, In Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'05) – Workshop on Embedded Computer Vision, (San Diego, CA), June 2005.

[6] J. Woodfill, G. Gordon, R. Buck, Tyzx DeepSea High Speed Stereo Vision System, In Proceedings of the Workshop on Real Time 3-D Sensors and Their Use, IEEE Conf. on Computer Vision and Pattern Recognition, (Washington, D.C.), June 2004.

[7] B. Yamauchi, "Wayfarer: An Autonomous Navigation Payload for the PackBot," Proceedings of AUVSI Unmanned Vehicles North America 2005, Baltimore, MD, June 2005.