Seeing cheaply: flexible vision for small devices

Dylan Muir and Joaquin Sitte

Smart Devices Laboratory, Centre for Information Technology Innovation,
Faculty of Information Technology, Queensland University of Technology,
2 George Street, Brisbane, QLD 4001 AUSTRALIA

E-mail: dr.muir@qut.edu.au, j.sitte@qut.edu.au

Abstract. In this paper, we outline an architecture for supporting real time autonomous vision in small devices. Our vision processor performs data reduction at the point of image capture, using re-configurable hardware devices to extract and describe image features. By implementing our design on FPGAs, the vision processor can support highly parallel vision processing structures as well as more traditional pipelined or sequential algorithms. Because the processor hardware can be reconfigured, it becomes a flexible prototyping tool. The resulting platform is well suited for research in reactive robot control based on vision, as well as research into biomorphic vision structures.

1 Introduction

Vision has long been the "holy grail" of senses. While it provides the most detailed information about the environment around an autonomous device, vision has also been difficult to implement. The extreme complexity and density of sensory data that is provided by a camera leads directly to difficulties in processing the vast quantities of information.

These difficulties are exacerbated when we try to incorporate vision in an autonomous mini-robot. Vision is power hungry, both computationally as well as physically. Adding acquisition and processing hardware consumes valuable space and energy. Attempting to feed such a high bandwidth data stream directly to the processor of a mini-robot causes its own problems.

In this paper we outline an architecture for supporting real-time autonomous vision in small devices. Our architecture enables vision to be included as a basic sense in autonomous mini-robots, without placing undue computational expense on the embedded system. Feature extraction and description algorithms can be implemented using both highly parallel and traditional sequential structures. The vision processor uses reconfigurable hardware, which provides a flexible common platform for a broad range of applications, as well as allowing quick prototyping and incremental development of vision based designs.

Because vision algorithms are implemented on this processor in hardware and in parallel, the processor is capable of full motion frame rates [1]. Coupled with a flexible communications protocol where the encoding of the features is entirely user-defined, our architecture can be used for applications ranging from vision pre-processing to reactive control performed entirely on board the vision processor.

1.1 Structure of the paper

In Section 2 we give some background on current vision processing techniques for embedded and autonomous systems, including monolithic von Neumann architectures and silicon

retinas. The properties of these systems and their limitations are examined. In Section 3 we discuss the architecture and design goals of our vision processor. The processing model used in our architecture is described, and in Section 4 we outline an implementation of the architecture in re-configurable hardware. In Section 5 we discuss applications for our vision processor. In Section 6 the paper concludes with a summary of the advantages of the vision processor over traditional architectures.

2 Vision in autonomous mini-robots

2.1 Traditional vision technology

In traditional computer vision systems the analog image stream from the video camera is sent over a cable to a frame grabber, and optionally, a digital signal processing (DSP) system hosted on a computer. While this technique yields powerful vision processing systems, it is not suited to autonomous mini-robotics. DSP and frame-grabber hardware is too bulky and too power hungry to mount on a mini-robot.

2.2 On board monolithic vision

Common embedded systems that incorporate on board vision use a custom camera with the image stream directly acquired by the CPU of the device. This method of acquisition uses the CPU interrupt system to read pixel data from the imager. Full motion vision is a high-bandwidth sensory stream, and uses a significant portion of the CPU processing bandwidth just to acquire the image frames. Using a single CPU for acquisition, image processing and control places limitations on the complexity of both the vision and control algorithms, as they have a limited number of processor cycles in which to execute if they are to perform real time control. Often the data stream bandwidth is reduced by lowering the image resolution or the rate of the image capture.

This architecture was used in the Eyebot series robots until the current version [2; 3]. Small to medium scale CPUs that are used for embedded systems and mini-robots struggle to directly acquire a 30 frames per second (fps) full motion image stream.

The latest generation Eyebot uses a buffering system, and can acquire image frames from its low-resolution imager at full motion rates. Previous Eyebot versions could only acquire images at 7 fps, a limitation imposed by using CPU I/O operations to interface directly with the imager.

Another approach, adopted by K-Team in the K6300 Vision Turret, requires an auxiliary processor to perform image acquisition and pre-processing [4]. This technique has the advantage of separating the control algorithms and the high bandwidth requirements of image acquisition, but still imposes compromises in either image resolution or acquisition frame rate.

These monolithic systems also suffer from limitations in terms of the vision algorithms that can be implemented. Because they rely on von Neumann processors, they are limited to sequential processing algorithms or simple pseudo-parallel constructs.

2.3 Parallel vision processing devices

Silicon retinas and other pixel-level processing devices have been developed to overcome the limitations of von Neumann and DSP vision algorithms, especially in the areas of early vision and optical flow [5; 1]. These devices implement parallel processing of pixels with processing constructs that resemble biological image sensing structures.

These devices have two major limitations. They are usually not programmable [6], or if they do allow configuration it is commonly in the form of weight adjustment. It is generally not possible to modify the algorithm that the device implements. The second limitation, shared by imagers in general, is that they do not reduce the bandwidth of the vision stream. This means that although some pre-processing is performed on the vision stream, the acquisition bandwidth required of the system CPU is not reduced.

In other words, these devices perform feature extraction but not data reduction. For example, a custom silicon device might perform edge detection, but still output an array of pixels. The signal-to-noise ratio is increased, but the bandwidth of the overall stream remains constant. Figure 1 shows an example of bandwidth reduction through describing the output of an edge detection device by identifying the end points of a line.

Some devices (for example, [7; 8; 9]) output a feature location or other information as an analog signal, but these still suffer from the first limitation: their processing algorithms cannot be modified. Fang [10] has developed a device that combines parallel neural constructs with an on-chip processor to perform further processing. This device is an attempt to create a more versatile vision system that can perform feature description and control. However, it is limited to neural-style processing at the pixel level.

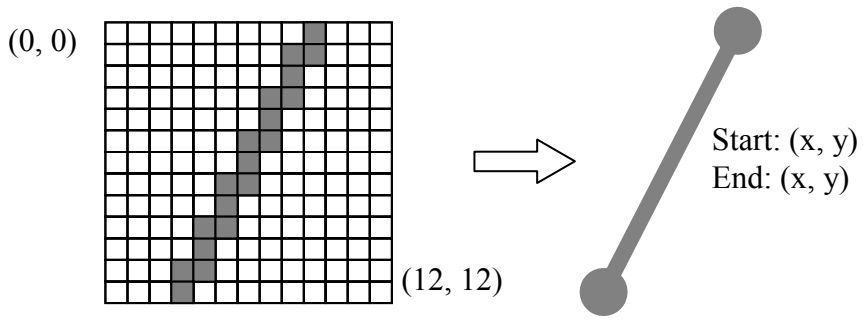


Figure 1. Bandwidth reduction through feature description. The figure on the left is the possible output from an edge detection device. The readout is an 13 by 13 matrix. By describing the line in terms of its end points, the amount of data required to represent the line drops from 169 bytes to 4 bytes.

3 Architecture

3.1 Design goals

The key design goal of our vision processor is to provide data reduction at the point of image capture for a physically small autonomous system with little computing power. This is accomplished by a sending a low bandwidth feature stream to the host device, rather than

requiring the host to perform full image readout and processing for each frame. To generate low bandwidth features, reconfigurable computing resources are used to compress the feature information extracted by highly parallel processing structures. These extra resources can be used to analyse the output of the highly parallel structures and describe the features in a more concise way.

Our design provides 'cheap' (computationally inexpensive) vision services to embedded systems. The design is flexible in the sense that it places few limitations on the vision processing algorithms that the processor can implement. It allows the use of any processing frame rate up to full motion vision. Size and power consumption are secondary goals, and only constrained to the extent that allows the design to be used in a physically small embedded system.

3.2 System interfaces

The vision processor interfaces with the host device through two communications busses. The control signals used to modify the operation of the vision processor are sent via a separate bus to the feature data. The feature data may be sent from the processor either in a continuous stream of feature frames, or on a by-request basis. Both buses are independently selectable for 8-bit parallel or serial communications. Figure 2 shows the vision processor interfaced to a host device.

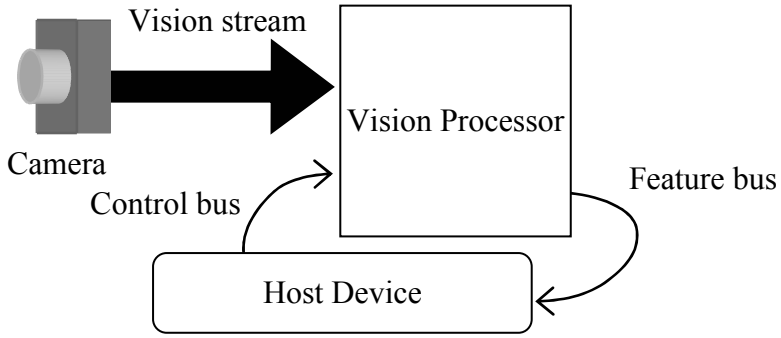


Figure 2. The vision processor interfaced to an embedded device. The vision processor handles all acquisition requirements. The low bandwidth feature data is sent to the host device via the feature bus. The host sends commands to the vision processor via the separate control bus.

The encoding of the feature data is set by the user when designing the feature extraction algorithms. Any type of encoding may therefore be used; from a human-readable ASCII format, through a compact binary representation, to a full frame pixel readout, if required.

3.3 Processing model

The human visual system has several vision streams connecting the retina to the visual cortex [11]. These streams originate in the retina, and have different properties related to the type of processing that is performed on them in the visual cortex. For example, a low resolution low latency stream might be used for detecting motion in the visual field, while a high resolution stream might be used for detailed analysis of a scene.

Inspired by this approach, our vision processor splits the incoming image stream into four vision streams. These streams operate with independent resolutions and regions of interest

within the incoming image. The properties of each stream can be controlled by the host device via the control interface.

3.4 Benefits of this architecture

A more conventional approach to developing vision system prototypes uses workstations to process vision signals from a remote camera. Our vision processor is designed to implement parallel processing structures, which present an added simulation overhead when prototyped on a von Neumann system. Evaluating pseudo-parallel structures on a real time full motion vision stream is almost impossible on a workstation.

Our architecture is compact enough to allow algorithm prototyping directly on the target embedded platform. This also allows the embedded system interface to be developed at the same time as the vision algorithms. In addition, the same processing fabric is used for both prototyping and implementation of the algorithms. The architecture itself can be scaled up or down to meet the needs of a specific implementation.

4 Design

4.1 FPGAs and reconfigurable hardware

To perform acquisition, control and processing, our design uses field programmable gate arrays (FPGAs). FPGAs consist of a large matrix of relatively low level digital logic elements, such as latches and look-up tables. These elements can be configured and connected in a programmable way to provide fully customisable hardware. FPGAs can be programmed via logic circuit schematics or through a hardware description language (HDL). FPGAs are capable of implementing traditional von Neumann processing architectures, or alternatively of implementing highly parallel processing structures. Because the low level computing elements work asynchronously, parallelised algorithms and structures can be evaluated much faster than on a typical fetch-and-execute architecture. They are re-programmable, and can be configured without being removed from the circuit board.

In our design, FPGAs are used to control the vision input device (including exposure and frame acquisition) without requiring separate frame-grabber hardware. FPGAs are also used for filtering and processing the acquired image, as well as handling communication with the host device.

Our design uses the Spartan-IIE family of devices from Xilinx [12].

4.2 Task partitioning

The functionality of the vision processor is spread over two FPGAs. Acquisition and control are performed independently of feature extraction and vision processing. The two FPGAs communicate through a high bandwidth double buffering system, implemented by dual-port RAMs. The use of dual-port RAMs allows the acquisition system to generate vision streams at full motion rates, while allowing the feature extraction system to process each vision stream at any frame rate.

Figure 3 shows an overview of the systems implemented in the two FPGAs.

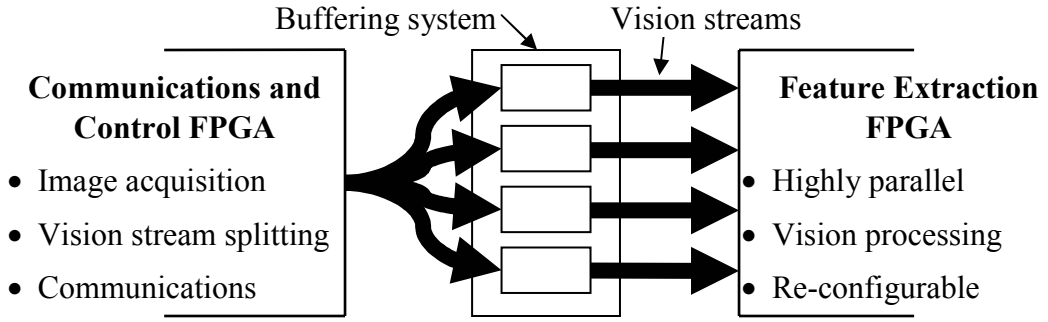


Figure 3. The functionality of the vision processor is spread between two FPGAs. The communications and control FPGA performs acquisition and vision stream splitting, and handles communications with the host device. The feature extraction FPGA is entirely configured by the user. The buffering system allows each vision stream to operate independently at full motion frame rates.

4.3 Communications and acquisition

Both communication with the host device and the acquisition of the image stream are managed by one of the FPGAs, shown in Figure 4. Within the FPGA, control of this functionality is implemented in a set of concurrent firmware modules.

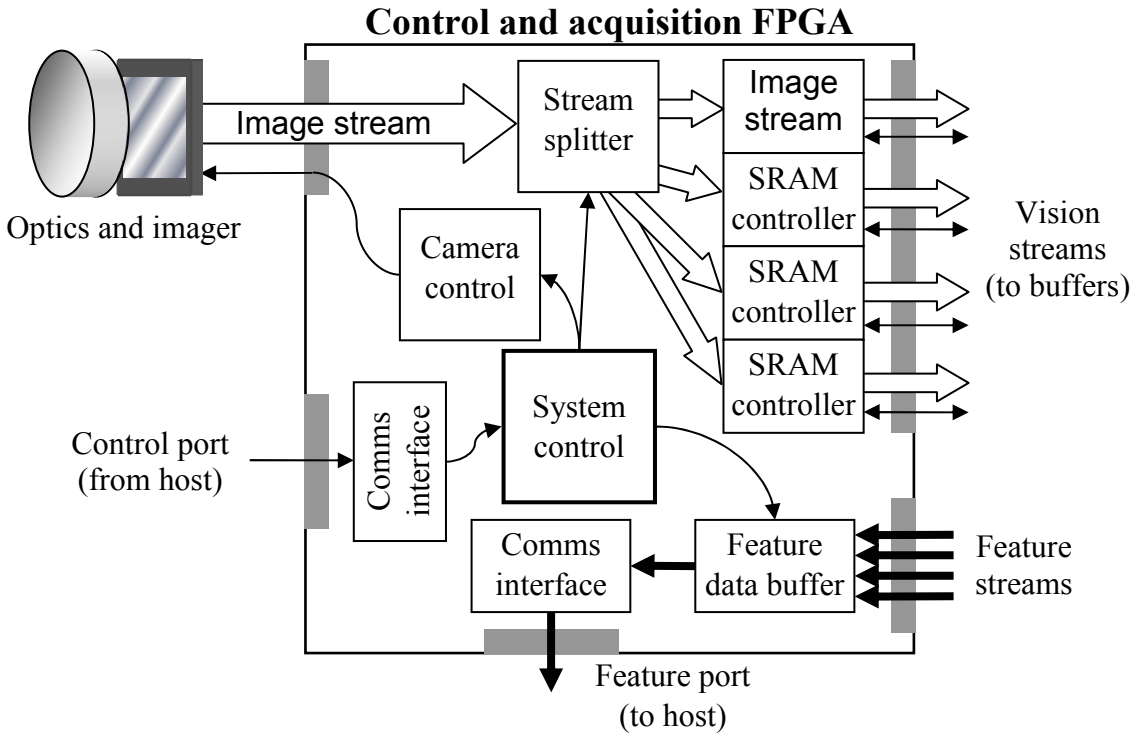


Figure 4. An overview of the control and acquisition systems for the vision processor. These systems are implemented in a single FPGA.

The stream splitter is responsible for dividing the incoming image frames into four separate streams. Each stream can have an independent spatial sub-sampling rate and region of interest within the source image. The vision streams are sent through the buffering system to the feature extraction FPGA by separate SRAM controllers. This ensures that the bandwidth of each stream is sufficiently high, and independent of the bandwidth of the other streams.

Feature data is received from the feature extraction FPGA over four separate feature busses, one for each vision stream. The frame rate of each feature steam is dependent on the processing time required by the feature extraction logic for that stream. The feature streams may run at any frame rate, and may have frame rates that differ from one another.

Depending on the configuration the host device requires, features corresponding to each stream can be sent to the host on various schedules. The data for each feature can be sent as soon as an updated feature frame is received, in a streaming approach. The host will receive feature information at the highest frame rate the feature extraction logic is able to achieve. The bandwidth of the feature port is such that full image frames can be delivered in real time, if necessary. However, we expect that encoded feature data will be considerably smaller than full image frames.

Feature data can also be sent to the host on a "by-request" basis. Under this scheme, the host device sends a request for a specific feature over the control port. The feature port then returns the latest feature frame available for the requested feature. This is a lower bandwidth solution for the host device, as feature data is only sent when the host is ready to process it.

Control of the imager, the properties of the vision streams and the way in which features are sent to the host can all be managed by the host device via the control port.

4.4 Feature extraction

The feature extraction and description component of the vision processor is implemented in the second FPGA. The interfaces to the vision stream buffering system and the feature stream busses are fixed, however the majority of the hardware resources in this FPGA are available for the user to configure. The feature extraction FPGA is shown in Figure 5.

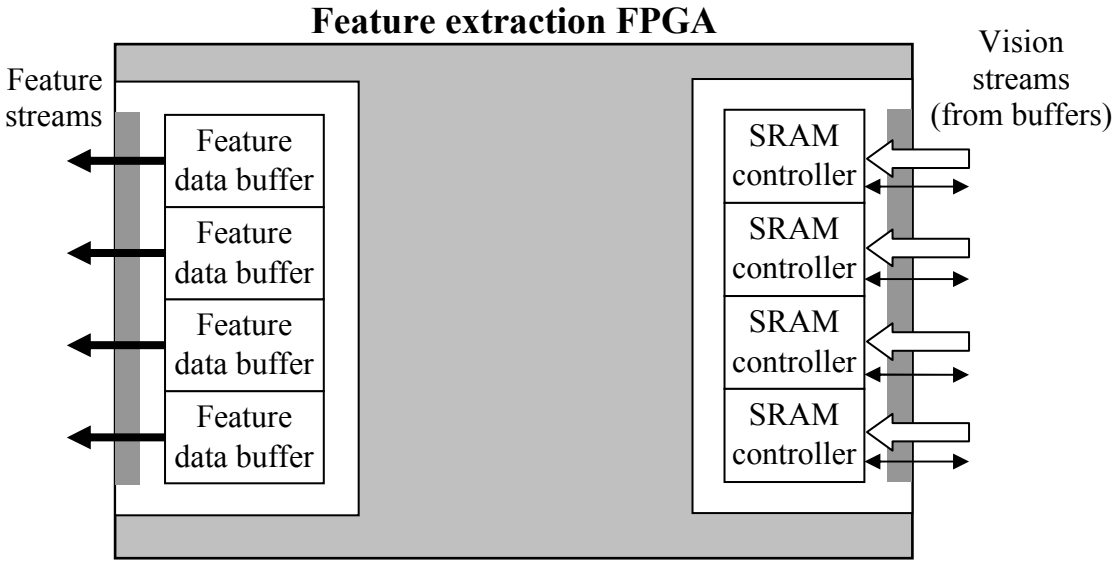


Figure 5. The feature extraction FPGA. The grey area represents the user configurable resources.

The designer of the feature extraction algorithms is presented with a standard SRAM interface to each double-buffered vision stream, as well as signals to indicate when a new stream frame is available. The feature extraction algorithms must interface with the feature data

buffers in a predefined way, to indicate to the control and communications system when feature extraction from a stream frame has been completed.

Aside from these predefined components, the designer is free to commit any amount of processing resources to any vision stream. Development for FPGAs is generally performed in hardware description languages (HDLs) such as VHDL or Verilog. HDLs provide a platform for modular development of asynchronous digital structures. HDLs also allow high level constructs to be developed based on lower level structures. Although most development for this processor would be performed in an HDL-type language, any source format supported by the Xilinx development tools can be used. This includes schematic entry, state machine design and the C language, as well as support for embedded processor cores. For more information, see [13].

4.5 Bus speeds and vision bandwidth

The image sensing device we use in this design is a Kodak KAC-0311 CMOS image sensor [14]. This device is capable of transmitting a full motion 640×480 pixel 10 bit monochrome 30 fps image stream with a 14 MHz pixel clock. A 640×480 pixel 8 bit monochrome full motion image stream, as used in our design, requires a bus bandwidth of 7.5 Mbyte/sec.

The buffering system performs synchronous readout at 40 MHz, with an independent 8 bit data bus for each vision stream. This provides a per-stream bandwidth of 40 Mbyte/sec. Each vision stream can occupy up to 2 Mbit per frame, giving an approximate maximum per-stream resolution of 640×410 pixels.

For a 30 fps image stream the frame time is 1/30 of a second, or approximately 33 msec. At 40 MHz, the image stream buffering system requires 6.6 msec to read a 2 Mbit frame, leaving 26 msec to process the frame and extract the features to meet the full motion deadline.

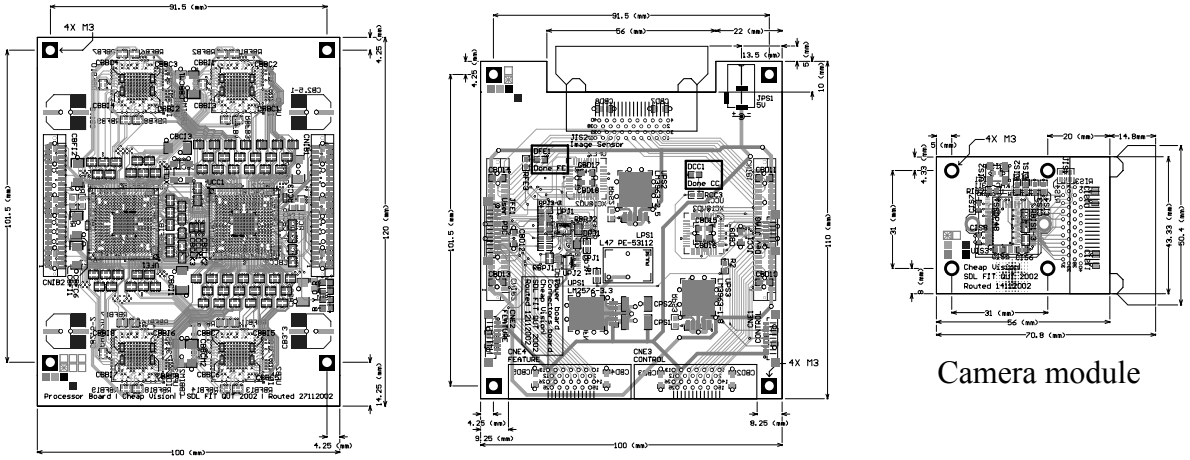
The feature extraction logic implemented in the FPGA can operate asynchronously, with internal configurable logic block (CLB) switching times in excess of 300 MHz [15]. In 26 msec, over 7.8 million layers of parallel processing structure could be evaluated. FPGAs do not yet contain anywhere near this amount of available logic, but this figure gives an indication of the size of the parallel structures that can perform feature extraction on a full motion vision stream.

The system clock distributed to the feature extraction FPGA runs at 80 MHz, therefore any synchronous feature extraction logic will run at this speed. The 26 msec full motion deadline leaves over 20,000 system clock cycles in which to process each frame.

4.6 Progress

The design of the vision processor hardware is complete. The processor consists of two stacked PCBs and a satellite digital camera module, connected by an IDC standard ribbon cable. The processor is approximately 120×100×30 mm in size, and the camera module is approximately 70×45×15 mm, with interchangeable lenses. The connections to the host device are via standard high density “D” connectors when operated in parallel mode, and miniature connectors when in serial mode. The PCB layouts for the vision processor are shown in Figure 6.

The system is configured using the Xilinx in-system programming software designed for their line of FPGAs. The software component of the system which will manage communication and support user designed vision algorithms is currently being designed.



Bottom PCB Top PCB Camera module

Figure 6 PCB layouts for the vision processor and the satellite camera module.

5 Applications

The re-configurability of our configurable vision processor makes it ideal for prototyping real time vision algorithms. Because the system is able to implement highly parallel structures, low-level and early vision algorithms can be tested and modified very rapidly. The platform is ideal for research in reactive robot control based on vision, as well as research into biomorphic vision structures.

The fabric of FPGAs is suited to structures and algorithms based on low-level digital hardware. This differs from traditional sequential or pipelined architectures, and limits the use of convolution-type operations. Analog structures such as those designed for silicon retinas also cannot be implemented in an FPGA. Designing vision processing for our architecture may require a re-thinking of these algorithms.

The vision processor we have described is designed to be interfaced to common computing hardware. However, because the feature output encoding is extremely flexible, the processor could be used to perform reactive control without external processing hardware. There is no limitation on the processing that can be performed within the feature extraction FPGA. Control algorithms could be included in the HDL design, coupled to the vision processing structures. The feature bus could then transmit encoded control information rather than image features. Combined with simple decoding and actuator driving hardware, the vision processor could be used alone for reactive control.

6 Summary

In this paper we have described our design for an embedded vision processor. This processor supports real time autonomous vision in small devices such as mini-robots.

Our design provides several advantages over traditional vision systems designed for autonomous mini-robots. Because image acquisition is performed by our processor, the CPU of the autonomous system is not loaded by a high bandwidth data stream. This leaves more CPU cycles free for control and higher level image understanding algorithms. Additionally, because our processor is not limited by the IO bandwidth and interrupt driven processing of modern CPUs, full motion vision can be incorporated into an autonomous system. FPGAs provide our design with the ability to implement sophisticated vision processing based on highly parallel structures, as well as allowing more traditional sequential algorithms to be used. The re-configurability of our processor means that these structures can be rapidly prototyped and tested on a real time vision platform.

References

[1] **Etern, Gail and Salam, Fathi M.**, Real time realization of early visual perception, *Computers and Electrical Engineering* **25**, pp. 379-407, 1999.

[2] **Bräunl, Thomas**, EyeBot: A Family of Autonomous Mobile Robots, *Proceedings of the International Conference on Neural Information Processing (ICONIP'99)*, Perth, Australia, pp. 645-649, 1999.

[3] **Bräunl, Thomas**, Scaling Down Mobile Robots: A Joint Project in Intelligent Mini-Robot research, *Proceedings of the Autonomous Minirobots for Research and Edutainment AMiRE2001, 5th International Heinz Nixdorf Symposium*, Paderborn, Germany, pp. 3-10, 2001.

[4] **K-Team**, *Khepera K6300 Vision Turret* [Online], Available: http://www.k-team.com/robots/khepera/k6300.html [2002, 23rd September], 2002.

[5] **Etienne-Cummings, Ralph and Van der Spiegel, J.**, A Focal Plane Visual Motion Measurement Sensor, *IEEE Transactions on Circuits and Systems I* **44**, 1, pp. 55-65, January 1997.

[6] **Moini, Alireza**, *Vision Chips or Seeing Silicon*, University of Adelaide, Adelaide, 1997.

[7] **Standley, David L.**, An Object Position and Orientation IC with Embedded Imager, *IEEE Journal of Solid-State Circuits* **26**, 12, pp. 1853-1859, December 1991.

[8] **Harrison, R. R. and Koch, Christof**, A Neuromorphic Visual Motion Sensor for Real-World Robots, *Proceedings of the Workshop on Defining the Future of Biomorphic Robots (IROS '98)*, Victora, B.C., Canada, pp. 1998.

[9] **Etienne-Cummings, Ralph and Van der Spiegel, J.**, A Foveated Silicon Retina for Two-Dimensional Tracking, *IEEE Transactions on Circuits and Systems II* **47**, 6, pp. 504-516, June 2000.

[10] **Fang, Wai-chi**, A Low-Power High-Speed Smart Sensor Design for Space Exploration Missions, *Acta Astronautica* **46**, 2-6, pp. 241-250, 2000.

[11] **Wandell, B.**, *Foundations of Human Vision*, Sinauer, Sunderland, USA, 1995.

[12] **Xilinx, Inc.**, *Xilinx Home : Products and Solutions : Silicon Solutions : Spartan-IIE FPGAs* [Online], Available: http://www.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?title=Spartan-IIE [2002, 25th September], 2002.

[13] **Xilinx, Inc.**, *Xilinx ISE Logic Design Tools* [Online], Available: http://www.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?title=Design+Tools [2002, 26th September], 2002.

[14] **Eastman Kodak Company, Inc.**, *Kodak Digital Science KAC-0311 Image Sensor* [Online], Available: http://www.kodak.com/US/en/digital/ccd/specDownload.shtml [2002, 3rd October], 2002.

[15] **Xilinx, Inc.**, *Spartan-IIE Data Sheet* [Online], Available: http://www.xilinx.com/partinfo/ds077.htm [2002, 3rd October], 2002.