FPGA based Real-Time Visual Servoing

Stefan Jörg, Jörg Langwald and Mathias Nickl
German Aerospace Center (DLR)
Institute of Robotics and Mechatronics
82234 Weßling, Germany
stefan.joerg@dlr.de

Abstract

Real-time image processing tasks not only require high computing power but also high data bandwidth. Though current processors excel in computing power, memory throughput is still the bottleneck for stream-oriented applications such as low-level image processing tasks. The alternative of special-purpose systems lacks flexibility at a high design effort and long development time. This effort often becomes void by the rapid advance of mainstream computing technology. FPGA technology promises flexibility and the necessary computing performance at affordable design costs. In this paper we describe our approach for a prototype image processing system for robot vision applications, based on FPGA technology. We use a commercially available PCI-Board to implement a typical application based on the Experimental Servicing Satellite (ESS) scenario.

1 Introduction

Real-time robot vision tasks such as visual servoing and object tracking require high computational power and data throughput, which often exceed those available on mainstream computer platforms. The first steps in a typical real-time vision application consist of a sequence of low-level algorithms operating on video streams and require only trivial control structures. This makes them ideal for implementation on a specialised system [7]. Therefore, an application can easily be divided into a pre-processing part located on a dedicated subsystem and a high-level part located on a host system.

Formerly, we used Datacube MaxVideo pipeline processors for implementing our image processing algorithms [2] [4]. The drawback of such specialised systems is the very restricted set of available operators. To overcome this limitation, we investigated alternative platforms for our pre-processing tasks. Special-purpose systems lack flexibility and come at the cost of much design effort and long development time. This effort often becomes void by the rapid advance of mainstream computing technology. Field Programmable Gate Arrays (FPGA) promise flexibility and sufficient computing performance. In the past, several FPGA based systems have been developed and evaluated for image processing, SPLASH-2 [3] and Programmable Active Memories (PAM) [10] being two prominent examples [9]. Those early attempts to create reconfigurable platforms focused on flexibility and scalability, thus yielding costly, general-purpose, multi-FPGA systems with attached memory.

For the domain of real-time robot vision, we propose a simpler system consisting of an image acquisition device and a single FPGA processor, integrated on a single board which interfaces to a host. To emphasise its intended use as a dedicated system for pre-processing tasks we label it the *intelligent frame grabber*. We validate our concept by implementing a typical robot-vision application based on the Experimental Servicing Satellite (ESS) scenario. In section 2 we illustrate the background of the ESS scenario and describe the used image processing methods. Section 3 briefly describes the concept of an *intelligent frame grabber*. In section 4 we lay out the porting and implementation of each step of the algorithm to the FPGA system. The results of our implementation effort are discussed in section 5.

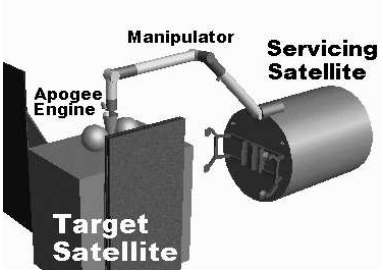

Figure 1. ESS capturing its target

2 The Benchmark Scenario

The Experimental Servicing Satellite (ESS) project investigated the problem of servicing non-cooperative satellites in or nearly in a geostationary orbit [8]. The goal was to capture a free-floating satellite by a platform equipped with an intelligent robotic system (Fig. 1). The most critical phases in this scenario are the final approach and capturing of the target.

Within the ESS project a test bed was implemented at our institute, where those final phases of vision based tracking and capturing were simulated by two robot manipulators: one playing the role of the target satellite and the other the role of the ESS. Capture is accomplished by entering the target's apogee engine thruster with a special tool equipped with sensors and a locking mechanism. Because of the stringent size constraints resulting from the integration of the camera into the capture tool, we used a miniature-sized camera with 12mm diameter and a focal length of 4mm. Unfortunately, lenses of this small size suffer from high distortions. For the required visual tracking of the target a very efficient model-based 3D visual tracking algorithm [11] was used, which relies on the robust extraction of image features such as edges. This requires the use of a calibrated sensor yielding rectified images. J¨org et.al [6] transferred this method to industrial assembly applications. Because of the relevance of the algorithm to a wide variety of applications and its typical complexity in terms of robot vision algorithms, we selected it as a prototype implementation for our *intelligent frame grabber* system.

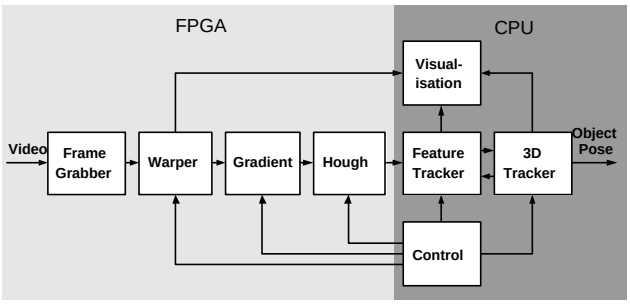

Figure 2. ESS Application

To implement the pre-processing part of the algorithm, the robust extraction of edges, the following operations are necessary: 1) Acquisition of PAL fields with size 384x287 at 50Hz. 2) The image is rectified using a cubic distortion model [12]. 3) The image gradient vector components are computed applying horizontal and vertical Sobel filter masks. 4) The Hough Transform is used to robustly extract all possible edge features of the gradient image. The resulting Hough Accumulator is sent to the CPU based feature tracker (see Fig.2).

3 The Intelligent Frame Grabber

The concept of an intelligent frame grabber is based on a strictly pipelined architecture: the image acquisition starts a pipeline of subsequent image processing operators. This concept allows modular application design at the operator level by implementing a framework for embedding operators in the pipeline. The high-level part of the application resides on the host and controls the pre-processing part, i.e. the parameters of each operator and the image acquisition, via a host interface. The pre-processor outputs images or more abstract information. The host interface extends the pipeline into the host which requires a tight coupling of host and intelligent frame grabber supported by a proper communication method (see Fig.2).

4 The System Implementation

We use a commercially available image processing board with an analogue frame grabber module and a single XILINX Virtex XCV2000E-6 FPGA with one SRAM bank and 6 SDRAM banks. The FPGA operates at 50MHz. As a host we use a Pentium III class Linux system. The image acquisition device is provided as VHDL library by the board manufacturer. For the physical connection of the operators within the FPGA we designed an image link, which consists of the data itself (e.g. grey-level pixel stream) and three synchronisation signals. The synchronisation signals *Pixel Enable*, *End Of Line* and *End of Frame* implicitly define the current image size.

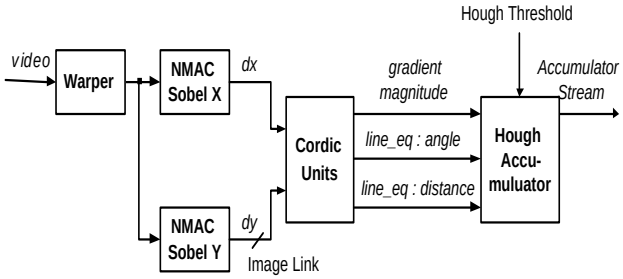

Figure 3. Implementation of the Hough Transform for Lines

The implementation of each of the three operators of our example scenario is described in the following sections. The processing pipeline starts with the warper operator, followed by the x/y Sobel gradient filtering stage, concluded by the Hough Transform for lines, which involves the calculation of the gradient vector from the x/y gradients (Fig. 3). The immediate result of the Hough Transform is the Hough Accumulator, which is transferred to the

host for further processing, thus marking the end of the pre-processing stage.

4.1 Image Rectification

The warper operator for image rectification is based on a sub-pixel accurate coordinate transformation using the following bicubic distortion polynomial:

$$\begin{aligned} x' &= a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 \\ &\quad + a_6x^3 + a_7x^2y + a_8xy^2 + a_9y^3 \\ y' &= b_0 + b_1x + b_2y + b_3x^2 + b_4xy + b_5y^2 \\ &\quad + b_6x^3 + b_7x^2y + b_8xy^2 + b_9y^3 \end{aligned} \quad (1)$$

The warper structure is shown in Fig. 4. The *Address Generator* computes the source address for the current output pixel using (1). The polynomial is implemented with 18 multiply and accumulate fixed point operations optimised for the given distortion coefficients. The integer part of the computed source address addresses the *Input Region Buffer*, which yields the addressed source pixel and its right, lower, and lower right neighbours. Using the fractional address part the grey level of the output pixel is bilinear-interpolated from these four input pixel grey levels.

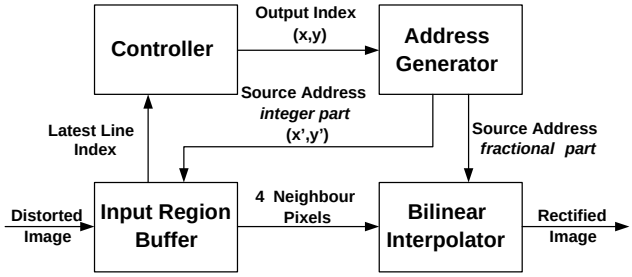

Figure 4. Image Rectification

A straight-forward implementation would require the buffering of the whole input image. An analysis of the camera specific distortion coefficients yields a fixed maximum number of source image lines to be buffered during the generation of one output line. In our case, the required buffer could be reduced to 32 input image lines. Therefore the *Input Region Buffer* could be placed within the FPGA.

The *Input Region Buffer* is filled by the incoming source pixels and the *Controller* is notified of the latest completed source line. If all lines needed for the next output line are available, the *Controller* generates the timing and pixel indices for that line at the full pixel rate of 50 MHz.

4.2 Sobel Gradient Filter

For the gradient calculation we implemented two parallel 3x3 neighbourhood operators to realize vertical and horizontal Sobel gradient filters at pixel rate (Fig. 3). The implementation is optimised for the Sobel coefficients. Two registers per line and two FIFOs are required to buffer the input pixels. Figure 5 shows the implementation of the vertical Sobel filter. After an initial delay of two lines and two pixels the filter yields one output pixel per input pixel.

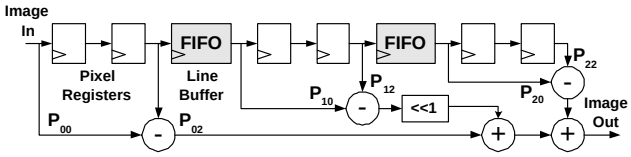

Figure 5. Vertical Sobel Filter

4.3 An Optimised Hough Transform

The Hough Transform for lines is described in detail in [5]. We formulate the feature equation for the Hough Transform as follows:

$$d = x\cos\theta + y\sin\theta \quad (2)$$

where d is the distance of the line to the origin and θ is the angle the x-axis makes with the perpendicular to that line. This θ is the phase of the grey level gradient vector for a point p on that line:

$$\theta = \arctan\left(\frac{\nabla_y}{\nabla_x}\right) \quad (3)$$

The magnitude of the gradient vector is defined as

$$\|\nabla\| = \sqrt{{\nabla_y}^2 + {\nabla_x}^2} \quad (4)$$

We implement the necessary trigonometric functions with the CORDIC algorithm as shown by Andraka [1]. A first CORDIC unit in vectoring mode implements (3) and (4). A second CORDIC unit rotates each image point by θ around the origin. This is equivalent to a rotation of our line (2) around the origin, i.e. the line becomes vertical. Thus the x component of that CORDIC unit equals d amplified by the CORDIC gain. Both units rotate simultaneously by the same angle and therefore share one angle accumulator.

The Hough Accumulator represents the complete, but quantised, parameter space of (2). We implement the accumulator as memory addressed by d and θ. The thus addressed cell is incremented when the magnitude of the gradient vector is larger than a configurable threshold. This reduces the influence of noise. The 16KB accumulator which is small compared to the source image, is transferred to the host. There the maxima in the accumulator, representing the edge features, are sub-pixel accurately extracted.

5 Experimental Results

We successfully implemented the pre-processing task of the ESS project as a validation of our concept of an *intelligent frame grabber* based on flexible logic. Figure 6a depicts the distored test image of a chequered board, Fig. 6b shows the rectified gradient image and Fig. 6c shows the resulting Hough Accumulator. All implemented algorithms

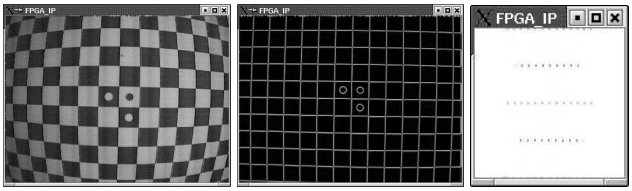

Figure 6. a) Distorted Image b) Rectified Gradient Image c) Hough Accumulator

process at real-time, i.e. PAL field rate. The Hough Accumulator is transferred to the host and processed at that rate, extracting every dominant edge feature in an input image.

The processing pipeline runs parallel to the image acquisition, i.e. processing starts as early as the warper is able to deliver the first rectified line. Compared to a standard frame grabber setup, the pre-processing stage is completed at about the same time the frame grabber delivers the source image to the host. Since the system clock operates at 50 MHz, the pipeline yields 50 MPixels/sec sustained throughput. To compare this figure with a general-purpose architecture we measured the throughput of a warper operator on a Pentium IV 3.0GHz/FSB800 system. An optimisied implementation achieves a maximum pixel rate of only 44 MPixels/sec. Moreover, this rate drops with every operator and processing stage added to the application. The *intelligent frame grabber* achieves shorter delay, constant throughput and an idle CPU available for further processing and visualisation tasks.

The FPGA usage of the application is moderate (see Tbl. 1). We require only 12.3% of the logical resources (Configurable Logic Blocks) and 57.5% of the internal RAM of our medium sized FPGA.

6 Conclusions

Typical pre-processing applications fit well in current FPGAs and can be implemented with reasonable effort. This is true for operators utilizing only internal memory, which is flexible and efficiently usable. Once operators depend on external resources the board design constrains its implementation. Thus the structure of an application affects the design requirements of an *intelligent frame grabber* board. Even for our specific domain a one-fits-all approach will not work. Future work will include the development of a highly-integrated board with design focus on easy adaption to the requirements of different applications.

Operator	Logic Blocks	%	Memory/kB	%
Warper	1645	8.6	12	15.0
Gradient	202	1.1	2	2.5
Hough	511	2.6	32	40.0
Total	2358	12.3	46	57.5

Table 1. Used FPGA Resources

References

[1] R. Andraka. A survey of cordic algorithms for fpga based computers. In *Proceedings of the Sixth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 1998.

[2] K. Arbter, J. Langwald, G. Hirzinger, G.-Q. Wei, and P. Wunsch. *Robust Vision for Vision Based Control of Motion*, chapter Proven Techniques for Robust Visual Servo Control. IEEE Press, 2000.

[3] P. Athanas and A. Abbott. Real-time image processing on a custom computing platform. In *IEEE Computer*, Feb. 1995.

[4] Datacube, Inc., Danvers, MA. *Pipeline Image Processing with ImageFlow*, Nov. 1998. Document No. MS001-2.1.

[5] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Comm. Assoc. Comput. Mach.*, 15(1):11–15, 1972.

[6] S. J¨org, J. Langwald, J. Stelter, C. Natale, and G. Hirzinger. Flexible robot-assembly using a multi-sensory approach. In *Proc. IEEE International Conf. on Robotics and Automation*, San Francisco, CA, 2000.

[7] W. Mangione-Smith and B. Hutchings. Configurable computing: The road ahead. In R. Hartenstein and V. Prasanna, editors, *Reconfigurable Architectures: High Performance by Configware*, pages 81–96, Chicago, 1997. IT Press.

[8] E. Settelmeyer, R. Hartmann, K. Landzettel, E. Lehrl, and W. Oesterlin. The experimental servicing satellite ess. In *21th. ISTS Conference*, Omiya, Japan, 1998.

[9] R. Tessier and W. Burleson. Reconfigurable computing for digital signal processing: A survey. *Journal of VLSI Signal Processing*, 28(1):7–27, June 2001.

[10] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, and P. Boucard. Programmable active memories: Reconfigurable systems come of age. *IEEE Transactions on VLSI Systems*, 4(1):56–69, 1996.

[11] P. Wunsch and G. Hirzinger. Real-time visual tracking of 3-d objects with dynamic handling of occlusion. In *Proc. IEEE International Conf. on Robotics and Automation*, Albuquerque, NM, 1997.

[12] P. Wunsch, G. Koegel, K. Arbter, and G. Hirzinger. Kalibrierung eines nichtlinearen, binokularen hand-auge systems. Technical Report No. 515-96-27, German Aerospace Center - DLR, 1996. http://www.robotic.dlr.de/vision/projects/Calibration/.