[8] A Pipelined Architecture for the Canny Edge Detector

Brendan P D Ruff

GEC Research Limited
Hirst Research Centre, Wembley, HA9 7PP, UK

Abstract

Low level vision algorithms deal with information at the pixel level. Their output is more abstract, meaningful, and compact, as it deals with the structure underlying the scene. General purpose processors are well suited to dealing with abstractions that require flexible processing, more so than they are to the simple and repetitive pixel processing task, a task that does not make use of control flow sophistication. Because of this dichotomy in processing style, a fixed low level front-end processor suggests itself as a dedicated real-time data "abstractor", presenting as its output data relevant, in this case, to edge based stereo processing. The stereo system will deal with the abstracted data at the same scene rate but slower data rate on a general purpose, and possibly parallel, processor. The low-level edge detection algorithm implemented, the Canny edge detector [1] , will be partitioned into a cascade of simpler operations in the dataflow, or **pipelined** manner, to exploit the algorithm's inherent structure.

1 Introduction

The Pipelined Canny system has been designed around the requirements of the PMF stereo algorithm [2] to remove the load of low-level processing from TINA [3] , a stereo 3-D modelling system with capabilities for object manipulation. The goal of any vision system is to operate in step with its environment, reacting to stimuli with sufficiently small processing lag so that a sensible response may be made. Biological systems perform this through massive, though slow, parallelism. Machines are limited in parallelism through size, but operate with many orders of magnitude greater speed. However, general purpose processors have large control time overheads in arranging data for processing and thus cannot achieve the full processing potential of silicon technology. This is not as serious a restraint in the symbolic processing for scene understanding as it is in the early processing of the massive amounts of pixel data thrown at the vision system. A pre-processor is required with control information designed implicitly into its architecture so removing time penalties. This processor, the Canny edge detector for the stereo vision system, extracts edge information from the intensity map of the image, so reducing the data volume from the square of the image luminosity array's dimension to the order of that data volume divided by sigma squared, where sigma is the standard deviation of the Gaussian used in the blurring operation of Canny.

2 The Canny Edge Operator

The Canny edge operator is an optimal edge detector addressing the twin goals of sensitivity and localisation. The processing involves several sophisticated operations. The pipelined Canny has an architecture that divides the processing task into many small stages. These may be grouped together into the following functional units as suggested in the original thesis by J.Canny [1].

1. 2-D Gaussian convolution with the image luminoscity array
2. Gradient and orientation calculation
3. Non-Maximal gradient suppression based upon the the gradient at a pixel and its two nearest neighbours along the gradient direction.
4. Interpolation of the maximum gradient position to sub-pixel accuracy based upon qaudratic or other fitting.
5. Thresholding with hysteresis to grow back weak edges but to allow greater noise immunity by setting a high threshold.

Each unit itself will be further sub-divided until 'nuclear' units are defined for processing as will be demonstrated in later sections.

These functional units are self contained, dealing with input data from the previous unit and producing an output. The final output of the system is a set of edges with gradient strength, orientation, and a sub-pixel offset to the edge in either the X or Y direction.

3 Pipelining philosophy

The pipeline approach to algorithm design is to represent the algorithm in the data flow representation. However all parts of the network are synchronous allowing one period of the pipeline clock to perform the operation within each part of the network. Thus an algorithm that is composed as a cascade of operations is divided into simple

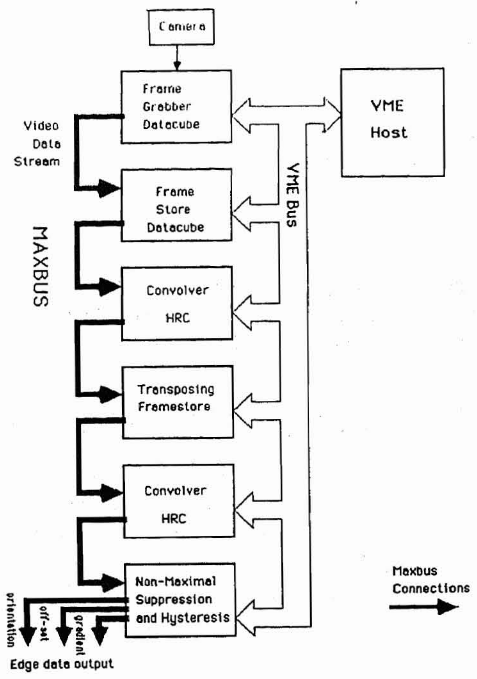

Figure 1: Front-End Edge Detector System

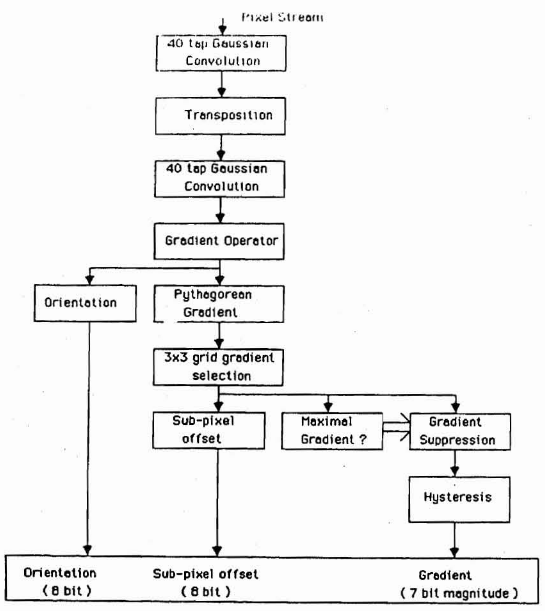

Figure 2: Pipeline of Modules within the Canny Edge Detector

operations, units, that may be performed within a single beat of the pipeline clock. The units are then cascaded so that each unit processes its input during the period between clocks in order that it presents its output to the next unit in the pipeline at the next clock period. It is the ordering of the units that allows input data to be processed in parts until emerging from the pipeline. It is then clear that each stage of the pipeline is busy during every beat of the clock, this being the strength of the pipeline processor. The penalty for such a distributed cascaded processor lies in the time latency between data entering the processor and later emerging. However one piece of processed data emmerges at every clock beat. For image analysis systems the latency induced by a pipelined processor is unlikely to affect performance as a latency of even several frames corresponds to several million stages in the pipeline but only a few milliseconds latency.

4 Pipelined Edge Detector

The edge detector is partitioned over several circuit boards in a standardised environment for its development and for interface to other processors for further analysis. Data is acquired with a frame grabber and then output in non-interlaced format over the Maxbus [4] video bus to a chain of three pipelined modules and an additional frame-store. Data is transmitted between modules on the Maxbus as is the final output of the pipeline. Figure.1 shows this system. Note that each module is additionally interfaced to the VME bus for control purposes. This environment coincides with all of Datacube's image processing modules and also with the MARVIN transputer system being developed as the processing engine upon which TINA is to execute.

The edge detector is composed of a chain of several modules. Figure.2. Each module is internally pipelined. The modules perform the following functions:

1. The first convolver performs a 40-point Gaussian convolution of the pixel stream.
2. The second transposes the output array of the convolver.
3. The third board performs a second 40-point convolution with an 8-bit result.
4. The fourth board performs the non-maximal suppression and hysteresis algorithm.

These modules define the functional parts of the Canny edge detector whose output is a data set containing zeroes for points not containing inflections, but for inflection points three pieces of data are given designed to interface with the TINA system:

1. the edge (gradient) orientation (8-bit orientation code)
2. the edge strength (8-bit sign magnitude)
3. the sub-pixel offset (8-bit sign magnitude) within the pixel to the edge position measured to an accuracy of 0.02 pixels for high contrast edges, degrading to 0.5 pixels for unit contrast edges.

This offset is measured either in the image 'vertical' or 'horizontal' direction according to which is nearest to the true gradient direction. Note that if the gradient is zero the other pieces of data should be disregarded.

It then remains to describe the architecture of each board.

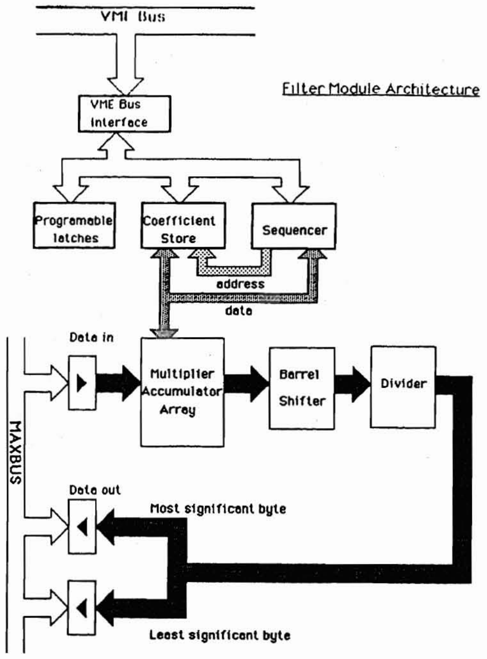

Figure 3: Convolver Architecture

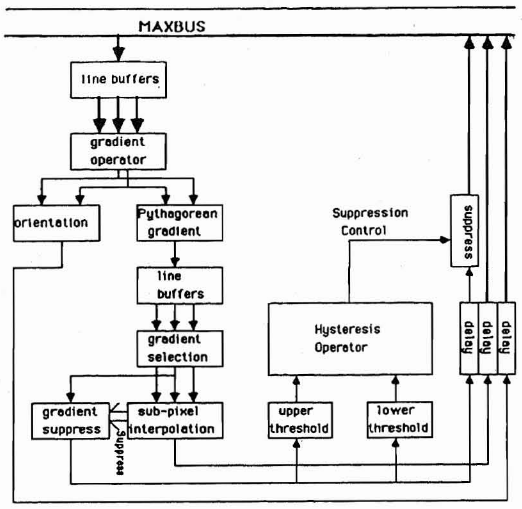

Figure 4: Non-Maximal Suppression and Hysteresis Module

5 Architecture of the boards within the edge detector

5.1 Convolver architecture

Figure.3 shows the architecture of the convolver. The convolver board receives an 8-bit data stream from the Maxbus and performs a 40-point convolution of this data with a 40-point programmable mask of 9-bit twos complement coefficients. The result is converted into sign magnitude format then arithmetically shifted left or right by up to 16 bits. This is programmable. The shifted result is divided by a 16 bit factor using a multiplicative division technique. The result is then either converted again to twos complement format or left as a magnitude only result. The 16-bit result requires two M‵axbus ports for output. The shifting and dividing stage is fully programmable to allow format adjustment and normalisation.

Operations are synchronised to the data stream with timing information supplied by the Maxbus timing port, P3. Control of the board's operation is effected via the VME bus. Various programmable latches control the functions of the board. The coefficient memory is fully read/write accessible to the VME bus for mask definition. The board behaves both as a Maxbus slave and a VME slave as is required for stream processing under the Maxbus philosophy.

Standard MSI TTL integrated circuits and PALs dominate the design. VLSI parts are used in the convolution. The convolution is implemented with five single chip multiplier-accumulator arrays each of which contain eight multiplier-accumulators of 8-bit data and 9-bit coefficient precision producing a 26 bit result.

5.2 Non-maximal suppression (NMS) and hysteresis module

This module houses two cascaded pipelined processors, figure.4. The first performs the non-maximal gradient suppression, outputting gradient strength, sub-pixel offsets, and edge orientation. The second performs the hysteresis edge thresholding algorithm. This produces, on a 3 line array, two single bit maps of the upper thresholded edges and lower thresholded, weak, edges. A third map is produced which consists of the upper thresholded, strong, map with any of the lower thresholded edges adjacent to the strong edges logically ORed onto it. This map then becomes the strong edge map of the next iteration of the algorithm. The weak edge map is unchanged. A number of iterations of this pipelined algorithm reduces edge noise considerably while incurring only a very low hardware overhead.

5.3 Non-maximal suppression, NMS

Eight-bit magnitude data is input from the maxbus. A gradient operator based upon a differencing convolution with the mask (1,0,-1) in horizontal and vertical image directions produces the partial derivatives of the already blurred image. Orientation and pythagorean gradient are then calculated via fast look up tables. Based upon the gradient direction, the central gradient and its two nearest neighbours are selected and passed to a processor that calculates the sub-pixel offset to the maximum, if the central gradient is indeed a maximum. This is performed in a fast look-up table based upon the difference of the central and largest other gradient compared to the smaller, giving only a 14 bit look-up. Otherwise the gradient is suppressed by asigning it a zero value. Gradient values are passed as a 10MHz stream to the hysteresis processor while orientation and sub-pixel offset data is input to delay elements to synchronise their final emergence to the maxbus with the output of the hysteresis.

5.4 Hysteresis

This algorithm initially builds up an upper and lower thresholded edge gradient map by performing a thresholding of the gradient with programmable upper and lower thresholds. The pipelined logical processor in figure.4 generates as its output a single bit specifying an edge or non-edge decision, suppressing noise edges. The gradient is delayed to synchronise it with the emergence of the decision from the processor. The gradient is then suppressed or maintained in accordance with the outcome of the decision.

6 Hardware design philosophy

Pipelined modules are simple encapsulated processors. Each performs its function within the time period of the pipeline clock. This naturally defines a highly modular design philosophy. Specification of interfaces between each module and module function are sufficient to define the module so that detailed design may be delegated to allow highly parallel development. Design testability is simplified as custom data may be inserted at any stage in the pipeline to test the proceeding pipelined unit. Test vectors of the output of the preceding unit must be generated.

7 Example of Pipelined circuitry

This example illustrates the pipelined technique for processor design for a simple operation performed at a rate of 10MHz.

Imagine that some second order polynomial for a group of 4 adjacent pixels, in a square, is to be calculated. Figure.5 shows how this may be performed. A line buffer is a set of N registers set head to tail to allow data to be delayed by N clock cycles, where N is the length of a line. The operation performed causes a latency of N cycles for the line delay, 1 cycle for a registered buffering to allow all four pixels to be accessed simultaneously. Squaring is performed by look-up table, an operation that can take as little as about 40ns (nano-seconds) up to 150ns or more for slower memory devices. In general, a 15-bit look up table will not be faster than 70ns, but an 8-bit table could be as fast as 20ns. The output of the square look-up tables is registered to allow synchronisation to the pixel clock so that the input to the next set of arithmetic units is stable. A set of additions is performed in three tiers to allow adequate processing time for each stage. It is possible that the square law look-up table and an addition stage could have been cascaded in one registered section (100ns time slice) and that two or more of the additions could have also been performed in one registered section. The propagation delay of these devices compared to the

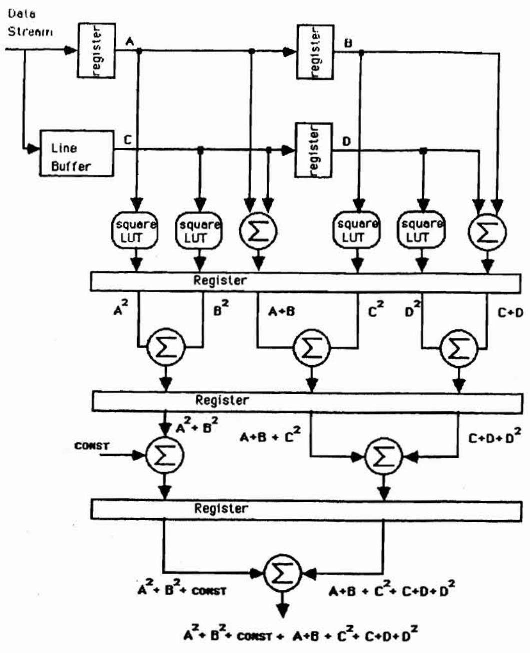

Figure 5: An Example of Pipelining

pipeline clock period will determine the number and size of operations to be performed as a cascade in one registered section of the pipeline.

8 Adapting to Industrial Applications

The pipelined or *data flow* architecture is highly suitable for design in custom VLSI circuits. The modular architecture and unidirectional flow of data in the pipeline allow for a standard-cell design approach (design using libraries of modular logic circuits) with cells linked by very regular and short interconnection. It is envisaged that an extreme compression of circuit area is possible through the custom approach. This will in turn allow very small size for the processor, more suitable to autonomous guided vehicles. Identifiable areas for size reduction through the custom silicon design approach within the Canny architecture are:

1. For smaller Gaussian convolutions a single chip solution is possible for an 8x8 array of multipliers with line buffering on-chip. This reduces the Gaussian convolution to the convolver and interface circuitry.

2. Non-maximal suppression for this architecture requires extensive use of look-up tables, already near their present limit in VLSI, but custom circuits would allow integration of line buffers and registers, multiplexers for the pixel and gradient selection network, either on separate chips or on one of the look-up table chips.

3. Hysteresis may be performed with a single chip incorporating the line buffers, logic network, and

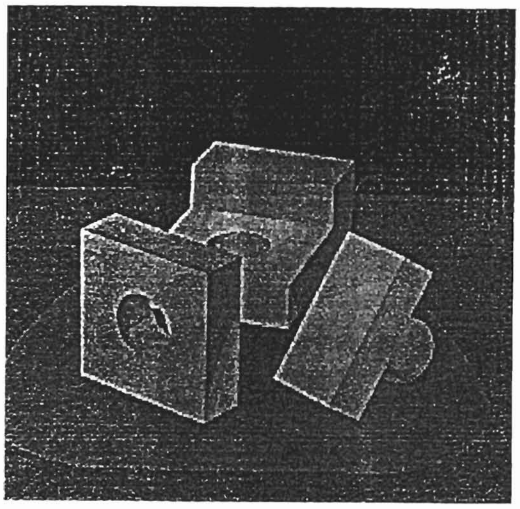

Figure 6: A Widget Scene

threshold selectors.

It is envisaged that the edge operator could be compressed into about seven integrated circuits for the computation, each around the 28 pin dual-in-line size. Additional cicuitry is required for interfacing to the VME bus and MAXBUS. The convolution, non-maximal suppression and hysteresis could then be performed on a single circuit board to provide a compact front-end processor for edge based stereo systems.

9 Simulations

The version of the Canny operator chosen for pipelined implementation uses integer arithmetic, restricted to 8-bits for the gradient and 8-bits for the final convolution output though the full resolution of the 9-bit coefficient and 8-bit pixel data is maintained until a final format adjustment reduces it to 8-bits. This accuracy is the minimum required sufficient to maintain to 0.02 pixel accuracy of high contrast edges using wide Gaussian convolution. Precision less than this degrades this performance. This precision is, however, quite convenient for standard 8-bit data paths.

Some simulations results are presented in figures 6,7,8. These show, respectively, a simulated widget scene generated by WINSOM [5] , a Canny edge map of the scene, and an expanded corner junction of the edge map to illustrate sub-pixel acuity (the grid represents pixel boundaries).

Theoretical edge positional accuracies are given below for the edge detector measured upon a test image of a circle of radius 60 pixels using a Gaussian of sigma 1.0, where the contrast across the circle boundary is varied between 2 to 200.

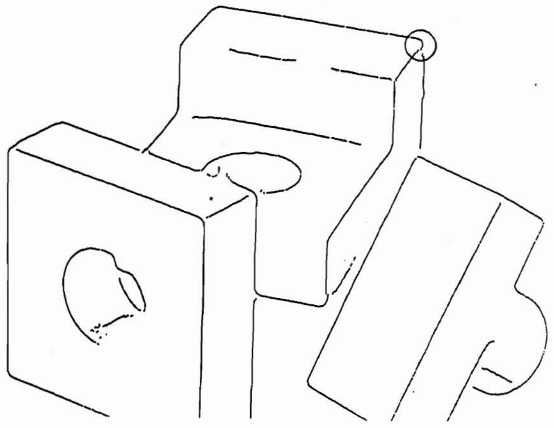

Figure 7: Canny Edge Detected Scene

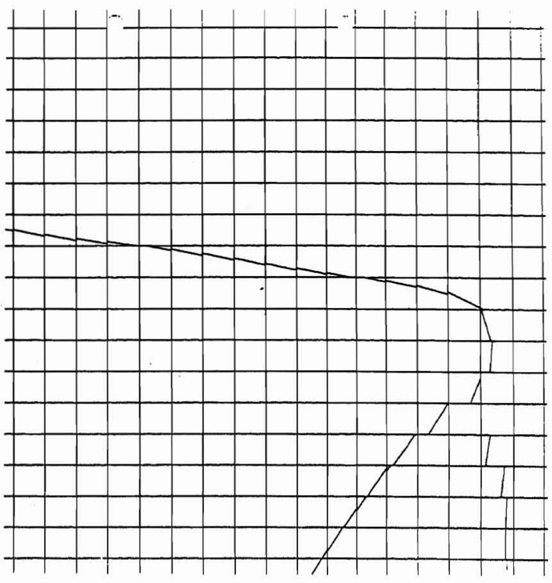

Figure 8: An expansion of the circled corner in figure 7 The graduations represent pixel boundaries

Intensity (0..255)	Mean magnitude error in edge position (pixels)
2	.48
20	.05
200	.02

Figure 9: Edge Positional Error for the Pipelined Edge Detector

10 Discussion

It has been shown how pipelining an algorithm can lead to a fast hardware architecture. It is clear that some algorithms are susceptible to the pipeline, or dataflow, architecture, in particular many of the low level vision algorithms, the algorithms typically used as the 'front end' to an image analysis system. Further, the real-time processing possible with this design philosophy allows the abstraction of higher level data to reduce data rate. This data is scene dependent, asynchronous to the input pixel stream and of a much reduced volume compared to the pixel data volume. It is hoped that the next layer on the analysis ladder of an image processing system will deal with this data in one, or several frame times, to produce higher level, yet still real-time, abstractions. For the PMF stereo algorithm implemented on a transputer network this is possible. It is hoped that a VLSI version of the system will allow a compact standard front-end processor for higher level vision engines.

References

[1] J F Canny. A computational approach to edge detection. *IEEE Trans Pattern Anaysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[2] Stephen B Pollard, John E W Mayhew, and John P Frisby. PMF - a stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14(4):449–470, 1985.

[3] J Porrill, S B Pollard, T P Pridmore, J B Bowen, J E W Mayhew, and J P Frisby. Tina: The sheffield vision system. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1138–1144, 1987.

[4] Datacube Inc. Maxbus specification manual. Technical report, 1985.

[5] P Quarendon. Winsom user's guide, report number uksc 123. Technical report, IBM Scientific Centre, Winchester, Hants, 1984.