ABINGDON CROSS BENCHMARK FOR PIPE

Sunil Fotedar

May 1990

Introduction

The purpose of the *Abingdon Cross* Benchmark is to evaluate the performance of image-processing architectures [ref. 1]. Its task is to find the medial axis of an 8-bit image of a white cross with a black background. The image of the cross is imbedded in noise, in such a way that the signal-to-noise ratio for the arms of the cross is 0 dB. Any algorithm may be used when the total execution time is the only factor considered. The problem consists of two parts: The first part is the noise removal and separation of the cross from the background. The second part of the problem is to find its *endoskeleton* or *medial axis transform* by connectivity-preserving thinning of the cross. Thinning is an iterative, local process which successively shrinks an object by removing border points. The total execution time T for both these steps is measured in order to compute the normalized performance figure. If the image size is N x M, a normalized value called the *quality factor* (QF), where

$$QF = \frac{\sqrt{NM}}{T}$$

is computed. In addition, the *price performance-factor* (PPF), where

$$PPF = \frac{N M}{T C}$$

is computed, defining C as the total system cost in United States dollars. PPF can be interpreted as the amount of processing power per dollar.

The objective of this report is to evaluate the performance of the Pipelined Image-Processing Engine (PIPE) using the Abingdon Cross benchmark. This work was done as part of a familiarization effort with the PIPE and may not represent the optimum selection of algorithms.

Experiment

An image of the Abingdon cross was generated with additive 0 dB white noise by a program on VAX 8350 computer. A program called *CROSS.PGM* with executable code *CROSS.PXE* was written and compiled on the Compaq 386/20 computer which acts as a host to the PIPE. First, Gaussian filtering is performed on the initial noisy image of the cross. The next process is thresholding of the image, and then dilation is done to fill out holes in the cross. The image output is then eroded a number of times. The number of times an image is looped back to perform erosion depends upon how wide the cross arms are in the image. For the current setup, the image was looped 9 times. No specific details were given for the dimensions of the cross. The dimensions for the vertical and horizontal arms of the image of the cross used in this experiment were 19 x 162 and 162 x 19 respectively. The last step in the program performs the connectivity-preserving line thinning on the image. The program loops back until the arms have been thinned to a one pixel-wide cross. The total time T to execute the benchmark for a 256 x 242 image of the cross was found to be approximately 0.75 second. Thus, for a $50,000 PIPE 1/400, we find:

$$QF = \frac{\sqrt{256 \times 242}}{0.75} = 331.87, \text{ and}$$

$$PPF = \frac{256 \times 242}{0.75 \times 50000} = 1.65.$$

Operation	Time taken (in sec.)
Gaussian Filtering and Thresholding	0.05
Dilating	0.017
Eroding	0.15
Thinning	0.533 approx.
Total Time Taken, T	0.75 approx.

Data Flow Graph

The data flow graph for this program is shown on the next page.

Description of the Loops Used

Three loops or subroutines are used in the program *CROSS.PGM*:

FILTER.LOP

* *Function:* Digitizes the noisy image of the cross, and performs Gaussian filtering, thresholding, and dilating of the image.
* *Stages:* 3 Modular Processing Stages (MPS)
* *Cycles:* 3

ERODE.LOP

* *Function:* Erodes the dilated image (a number of times).
* *Stages:* 3 MPS
* *Cycles:* 1

THIN.LOP

* *Function:* Thins the eroded image of the cross until the cross is one-pixel wide.
* *Stages:* 3 MPS
* *Cycles:* 3

Description of Look-up Tables

5.1 Single Valued Look-up Tables

These look-up tables transform each pixel in the input image, either in unsigned magnitude (pixel value from 0 to 255) or 2's complement (pixel value from -128 to 127) format,to an output image by a transfer function. The output image is also in unsigned magnitude or 2's complement format.

NEGATE.LUT

* *Function:* This LUT writes a 11111111 in the output pixel if the corresponding input pixel is 00000000 and writes 00000000 in the output pixel otherwise.
* *Transfer Function:* output = thr(255,0,1,a).
* *Input/Output Number System:* Unsigned magnitude.

THR0FF.LUT:

* *Function:* This LUT writes a 00000000 in the output pixel if the corresponding input pixel has a value less than zero and writes 11111111 in the output pixel otherwise.
* *Transfer Function:* output = thr(0,255,0,a).
* *Input Number System:* 2's complement.
* *Output Number System:* Unsigned magnitude.

UNITY.LUT

* *Function:* It replicates the value of the input pixel in the output pixel.
* *Transfer Function:* output = a.
* *Input/Output Number System:* 2's complement.

ZERO.LUT

* *Function:* To create an image in which all pixels have a zero value.
* *Transfer Function:* output = 0.
* *Input/Output Number System:* 2's complement.

5.2 Arithmetic Neighborhood Operations

Two types of arithmetic neighborhood operations are performed by the PIPE hardware: 3x3 and 9x1. With these operations, each pixel in an image is replaced by a weighted sum of itself and its 8 neighbors, where the weights are entries of a 3x3 or 9x1 mask.

GAUSS.ANP:

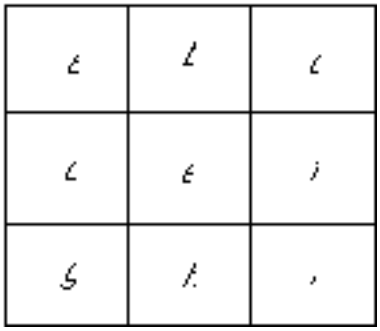
* *Function:* This arithmetic neighborhood operator is used for low-pass filtering of the image.
* *Mask:* It uses the following 3x3 mask:

1	2	1
2	4	2
1	2	1

* *Input/Output Number System:* 2's complement.

5.3 Boolean Neighborhood Operations

All the 8 bits of a pixel can be manipulated independently by boolean neighborhood operations. The nine pixels in the neighborhood are designated by



and the output pixel is designated as *z*. The following boolean operations have been used in this program. Symbols |, &, and ~ represent logical operations OR, AND, and NOT respectively.

B.BNP

** Boolean Operation:*
z = b;

BENOTH.BNP

** Boolean Operation:*
z = b&e&~h;

H.BNP

** Boolean Operation:*
z = h;

HORZFILL.BNP

** Boolean Operation:*
z = ~e&~a&c&b; 9x1

DILATE.BNP

** Boolean Operation:*
z = a|b|c|d|e|f|g|h|i;

ERODE.BNP

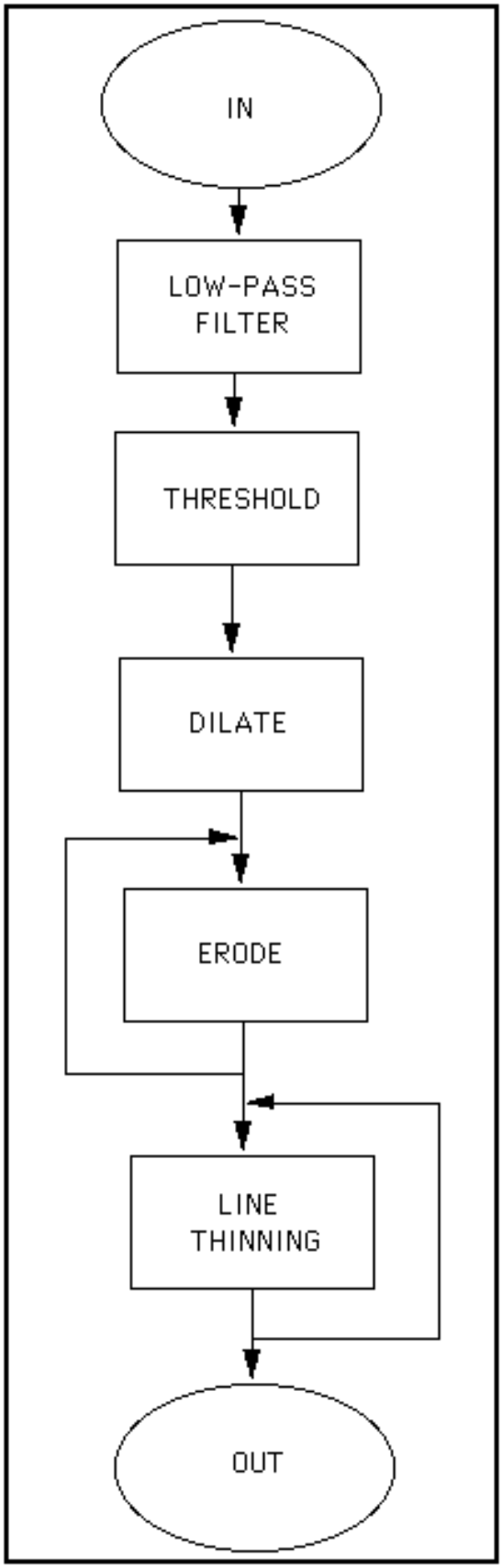
** Boolean Operation:*
z = a&b&c&d&e&f&g&h&i;

THIN1.BNP

** Boolean Operation:*
z = ~(~a&~b&~c&d&e&f&h)&~(~a&~d&~g&b&e&h&f)&
~(~g&~h&~i&d&e&f&b)&~(~c&~f&~i&b&e&h&d);

THIN2.BNP

** Boolean Operation:*
z = ~(~b&~c&~f&d&e&h)&~(~a&~b&~d&h&e&f)&
~(~d&~g&~h&b&e&f)&~(~h&~i&~f&d&e&b);


IN
LOW-PASS FILTER
THRESHOLD
DILATE
ERODE
LINE THINNING
OUT

Acknowledgments

We are indebted to Mr. Randall Luck, Mr. James Herriman, and Ms. Shoshana Biro of Aspex Inc. for their suggestions and constant help.

References

1. Preston, K. Jr., "The Abingdon Cross Benchmark Survey," Computer, pp. 9-18, July 1989.