1 Apply Overview

Image-to-image operations are often the most time consuming step in image processing applications. This step involves the mapping of an input image into an output image through a local operation that applies to a window around each pixel of the input image. Algorithms that fall into this class of operation include edge detection, smoothing, convolutions, contrast enhancements, color transformations, and thresholding.

These low-level vision operations are often time consuming because of the size of the images being processed. A 512 x 512 pixel image requires the operation to be applied 262,144 times. Parallel processing speeds up these operations however, and high efficiency can be achieved by partitioning the input and output across processors (domain decomposition).

Typically, when a programmer designs a parallel program, the programmer must build both the program and the programming model. This adds to development time and can result in a program that is difficult to change since the programming model is specific to the program.

For this reason, parallel program generators have been designed to improve parallel programming. Apply is an application-specific parallel program generator for low-level vision image processing that translates a users sequential program into programs for each cell in the system. Apply was developed by Carnegie-Mellon University to implement low-level vision algorithms and has been integrated into the iWarp™ system for image processing applications. Apply allows the programmer to write high-level image processing programs without having to be concerned with the implementation of the programming model.

The Apply programming language is designed for programming image-to-image computations where the pixels of the output images are computed from corresponding rectangular windows of the input images. Each Apply procedure operates on a single pixel position, and the Apply compiler generates a parallel program that executes the procedure in parallel over the entire image on an iWarp array. An Apply procedure does the following:

- accepts a pixel or window of pixels from each input image
- performs a set of computations on the input
- returns a pixel value for each output image

You call Apply procedures from a C main program, passing the input and output images and other parameters as arguments. On the iWarp system you can generate both single-cell (sequential) programs and multi-cell (parallel) programs with Apply.

An Example

The following Sobel edge detection example illustrates the use of Apply. In the example, Sobel edge detection is performed by convolving the input image with two
3 x 3 masks. The horizontal mask measures the gradient of horizontal edges, and the vertical mask measures the gradient of vertical edges. Diagonal edges are operated on by both masks, allowing edge orientation and strength to be measured for all edges. Figure 1-1 illustrates the masks.

$$\begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} \qquad \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix}$$

HORIZONTAL VERTICAL

Figure 1-1: Sobel Convolution Masks

The following example illustrates an Apply implementation of the Sobel edge detection. The lines are numbered in comment form to help explain the program, but line numbers are not part of the Apply language.

```
procedure sobel (inimg   :  in array (-1..1, -1..1) of byte --1
                            border 0,
                 thresh  :  const real,
                 mag     :  out real)

is                                                          --2

      horiz, vert  :  integer;                              --3

begin                                                       --4

      horiz := inimg(-1,-1) + 2 * inimg(-1,0) + inimg(-1,1)  --5
               - inimg(1,-1) - 2 * inimg(1,0) - inimg(1,1);

      vert  := inimg(-1,-1) + 2 * inimg(0,-1) + inimg(1,-1)  --6
               - inimg(-1,1) - 2 * inimg(0,1) - inimg(1,1);

      mag   := sqrt(REAL(horiz) * REAL(horiz)                --7
               + REAL(vert) * REAL(vert));

      if mag < thresh then                                   --8
         mag := 0.0;                                         --9
      end if;                                                --10
end sobel;                                                   --11
```

Line 1 defines the input, output, and constant parameters to the procedure. The input parameter *inimg* is an input image with a 3 x 3 window. The constant parameter *thresh* is a threshold value. Edges that are weaker than this threshold are suppressed in the output magnitude image, *mag*.

Line 1 also defines the input image window. It is a 3 x 3 window centered about the current pixel processing position. **BORDER 0** specifies that the border values are filled with 0 when the window extends beyond the edges of the image. The border keyword is described fully in Chapter 2.

Lines 2, 4, and 11 contain keywords that define separate segments of the Apply procedure.

Line 3 defines the internal variables *horiz* and *vert*, which hold the results of the horizontal and vertical Sobel edge operator.

The computation of the Sobel convolutions is implemented in the expressions on lines 5 through 7. Suppression of edges that are weaker than the threshold is implemented on lines 8 and 9.

Chapter 2 fully defines the Apply language, including border handling and image reduction and magnification.

About This Manual

This manual is intended for the new or experienced Apply user, and contains both tutorial and reference information. The manual is structured as follows:

Chapter 1:	Provides an overview of Apply and this manual.
Chapter 2:	Describes the Apply language.
Chapter 3:	Describes how to develop and execute an Apply program.
Appendix A:	Provides a definition of the Apply language in the Extended Backus-Naur form.
Appendix B:	Describes Apply’s intermediate language.
Appendix C:	Describes WebLib, a library of Apply image-processing routines.