Proc. SPIE Scanning Microscopy Technologies and
Applications, Vol. 897, Los Angeles, CA., January 1988, 169-173.



# Application of the PIPE Image Processing Machine to Scanning Microscopy

*Martin Herman*

Sensory Intelligence Group
National Bureau of Standards
Gaithersburg, MD 20899

*ABSTRACT*

PIPE is a pipelined image processing device that was designed for real-time robot vision applications. It accepts images from a video camera 60 times per second, and contains hardware for digitizing, displaying, and performing operations on these images at video rates. Each stage of the pipeline contains arithmetic and logic units, convolvers, image buffers, and look-up tables. The purpose of this paper is to introduce PIPE and some of its applications to the scanning microscopy community. Three kinds of applications are described. The first is stereo analysis, whose purpose is to automatically extract range from two cameras mounted side by side. The second application is motion analysis and tracking. This application involves detecting motion, measuring its velocity, using it to obtain three-dimensional information, and tracking it through time. The final application is inspection of two-dimensional patterns.

## 1. Introduction

PIPE (Pipelined Image Processing Engine) is a multi-stage, multi-pipelined image processing device that was designed for real-time robot vision applications. These applications have included robot guidance, medical analysis, flexible manufacturing systems, remote image analysis, autonomous vehicle guidance, inspection, and range image processing. The purpose of this paper is to introduce PIPE and some of its applications to the scanning microscopy community. Three tasks in particular -- stereo vision, motion analysis, and inspection -- seem to relate closely to some of the image processing needs for scanning microscopy. This paper will describe these three applications in the context of robot vision. The paper will also provide an overview of the hardware and capabilities of PIPE.

PIPE was conceived and designed at the National Bureau of Standards [Kent, Shneier and Lumia 85]. It is currently commercially available through Aspex Incorporated. PIPE will accept images from a video camera at field rates -- 60 times per second. PIPE's basic cycle rate is 1/60 second. Therefore, many powerful algorithms can run on PIPE several times per second. The following is a list of the kinds of algorithms thus far implemented on PIPE at NBS.

1. Spatial filtering
   -- edge-preserving smoothing, Laplacian-Gaussian filtering, multiresolution Gaussian and Laplacian pyramids.
2. Temporal filtering
   -- temporal averaging, spatio-temporal smoothing.
3. Edge detection
   -- Sobel edge operator, zero crossings, non-maximum suppression.
4. Line extraction
5. Motion detection and tracking
   -- centroid tracking, optical flow extraction.
6. Stereo matching
7. Image rotation
8. Point processing
   -- scaling, thresholding, contrast stretching.

The most complex of these algorithms -- stereo matching -- runs once per second. Some of the algorithms, such as point processing or spatial smoothing, run at 60 times per second. The other algorithms run at several times per second.

## 2. Description of PIPE

Figure 1 shows how PIPE is meant to fit into an overall computer vision architecture. The architecture includes PIPE, a feature extractor designed specifically for PIPE called ISMAP (Iconic to Symbolic MAPper), and a host computer which is typically a standard computer that performs symbolic processing.

In Figure 1, PIPE serves as the initial processing stage which accepts an image consisting of an array of gray scale values and produces a similar array of low-level symbolic values which encode features obtained from the gray scale array, such as edges, motion directions, corners, range, etc. This process of feature detection produces an iconic description of image features from an iconic description of image intensities. Therefore, associated with each image location is the feature at that location.

Following the feature detection process, ISMAP serves to perform feature extraction (iconic to symbolic mapping). In this step, the iconic image of features, in which symbol values are indexed by image location, is mapped into a list of feature values, where associated with each feature is the set of image locations containing that feature. This is called a histogram. Therefore, image location is indexed by feature value. This accomplishes two goals: it permits finding the locations of features of interest by direct indexing rather than by exhaustively searching the image, and it extracts locations as data to which subsequent operations can be applied. These subsequent operations are performed on the host computer, and involve finding relations among the location data which correspond to geometric relations of features. The geometric relations discovered then serve as input to subsequent classification processes.

Notice in Figure 1 that the host computer may receive from PIPE iconic images as well as histograms. Furthermore, the host computer can send iconic images to PIPE.

PIPE was designed with the following goals in mind:

1. real-time processing of images at field rate (1/60th second),
2. provision for interactions between related images, such as those arising from dynamic image sequences or from stereoscopic views,

3. provision for the ability to apply different algorithms to different regions of the image in real time,
4. ability to perform multi-resolution image processing,
5. ability for guiding processing by knowledge-based commands and "hypothesis images" supplied from the host computer.

PIPE is a hardware device specialized for parallel image processing rather than a fully general purpose computer. Within the limits of the processes it supports, it is an extremely fast and flexible device. On the other hand, since it is not a general purpose computer, it is not possible to efficiently program arbitrary algorithms on it.

PIPE is intended as a processor for local operations on images; it is not designed to efficiently perform operations that require global knowledge of the image. This latter step is intended for the host computer.

The basic design of PIPE is shown in Figure 2. It consists of a Video Interface Stage (responsible for A/D and D/A conversion between RS170 fields and 256 x 256 x 8 bit images), Input and Output stages consisting of two frame buffers each and some image combining logic, and up to eight Modular Processing Stages (MPS) which perform the primary computations. The Output Stage interfaces to the ISMAP. Each MPS stage has image buffers which receive images, and operators which act on them. Images are transferred from stage to stage at field rate (60 images/sec) by three concurrent pathways. The forward pathway acts as a traditional pipelined image-processing path. The backward pathway carries images in the opposite direction, from the output of a stage to the input of the previous stage. This allows a sequence of images taken over time to be compared. It also allows a hypothesis image to be compared with an input image. The recursive pathway carries an image from the output of a stage back into the input of the same stage. This allows feedback and relaxation processing.

At the input to each stage, the images carried by the three pathways are first mapped separately through arbitrary functions implemented via look-up tables. Then they may individually undergo any arithmetic or boolean operation. Then any arithmetic or boolean operation may be used to combine the three input images into a final input image, prior to its storage in one of two buffers within the stage.

Within each stage, operators may act on images stored in either or both of the two buffers. These operators include two simultaneous and independent arithmetic or boolean neighborhood operations (3x3 convolvers), and the application of an arbitrary function of one or two arguments (single-valued and two-valued look-up tables).

The output from the stage involves sending to the forward, recursive, and backward pathways images resulting from any of these operations or from any of the buffers.

In an alternative mode of operation, one of the two image buffers in each stage may serve as a map for selecting the algorithms to be applied to the contents of the other buffer. In this mode, PIPE functions as a multi-instruction stream multi-data stream (MIMD) machine; the algorithm to be applied to an image is determined on a pixel-by-pixel basis as the image is processed. Currently, 16 such alternative algorithms may be selected in real time.

In another mode of operation, PIPE functions as a multi-resolution pyramid machine. In this mode, images carried by the forward pathway are reduced in size by one half at each stage, while sizes of the images carried by the backward pathway are doubled at each stage. The images carried by the recursive pathway remain unchanged in resolution. Any combination of stages may operate in this mode, under program control.

In addition to the pathways mentioned, PIPE contains six general purpose video busses which allow images to move from the host, or from any buffer in the machine, into any other buffer in the machine.

In any of PIPE's operating modes, the operations of every stage are completely independent, and can be completely reconfigured in the inter-field time (1/60 second) by the system controller stage, which in turn may select stage configurations from a stored sequence.

PIPE processes images in four ways: point, spatial, boolean, and sequence. Point processing involves applying a function to each individual pixel point in the image. The application of a function is done through a look-up table. Examples of such functions are scaling, thresholding, and contrast stretching. Spatial processing involves applying an arithmetic neighborhood operator to the 3x3 neighborhood surrounding each pixel. Examples of such operators are Sobel edge detection and smoothing. Boolean processing involves applying a boolean neighborhood operator to the 3x3 neighborhood surrounding each pixel. Each of the 8 bit planes associated with each pixel is treated separately, and logical operations such as "and" and "or" are applied to each bit plane. Sequential processing involves performing a temporal operation involving a sequence of three images. Examples of such operations include motion detection and temporal smoothing.

An eight stage PIPE performs 1.2 billion arithmetic, or 7.36 billion Boolean sum-of-products, operations per second. Further details describing PIPE may be found in [Kent, Shneier and Lumia 85].

## 3. Stereo Analysis

Stereo analysis is a technique used to obtain range from a two-camera system. The key problem in stereo computation is to find corresponding points in the stereo images. Corresponding points are the projections of a single point in the three-dimensional scene. Figure 3 shows a two-camera stereo system. The respective left and right focal points are $F_l$ and $F_r$, the respective image planes are $I_l$ and $I_r$, and the respective optical axes are $z$ and $z'$. A point $P$ in 3-space is projected onto $P_l$ in the left image and onto $P_r$ in the right image. The difference in the positions of two corresponding points $P_l$ and $P_r$ in their respective images is called "parallax" or "disparity." The disparity of $P$ is a function of both its location in the scene, and of the position, orientation, and physical characteristics of the stereo cameras. When these camera attributes are known, the location of $P$ can be determined [Barnard and Fischler 82].

A stereo analysis algorithm that computes correspondence has been implemented on the PIPE. The algorithm was developed by Allen Waxman of Boston University, and is based on an algorithm developed for the Connection Machine [Drumheller and Poggio 86]. The steps of the algorithm are as follows:

1. Filter each image using Laplacian Gaussian filtering. The Laplacian Gaussian filter is approximated as a difference of Gaussian smoothed images.
2. Obtain the zero crossings of the difference of Gaussian image. The result of this step is a binary image where zero-crossing points have a value of 1 and the background has a value of zero.
3. Shift one zero crossing image across the other, one column at a time. For each shifted position, check whether a zero crossing from one image falls on top of a zero crossing of the other image. If it does, mark a bit in x-y-disparity space. (This space is represented by the bit planes associated with each pixel; each bit plane represents a different disparity.)

4. Perform a three-dimensional convolution: each point in the center of a three-dimensional cube (in x-y-disparity space) is given a number equal to the number of bits marked inside the cube.
5. At each x,y point in the image, look for the disparity with maximum value. This disparity value is then assigned to that image point.

This algorithm has been tested in real time on several scenes. It runs in approximately one second. Compare this with some stereo algorithms which run in many minutes or even hours. It is a very general stereo algorithm which is designed for arbitrary natural scenes. The results are not as good as we expected, but it is a first step towards real-time stereo analysis on PIPE.

## 4. Motion Analysis and Tracking

Several different motion detection and analysis algorithms have been implemented on PIPE. All of these involve comparing successive images. The simplest such algorithm to implement is to take the difference of successive pairs of images, and to display this difference. This algorithm is useful for detecting motion in the visual field, but it does not give the velocity of this motion.

An algorithm that detects motion and provides velocity of motion in the image plane (i.e., the manifestation on the image plane of object motion in the world) was developed by Horn and Schunk [Horn and Schunk 81]. However, it does not recover the full image velocity. It only recovers the component of velocity normal to image edges. The algorithm assumes that lighting remains constant, and objects in the visual field remain stationary while the camera moves around. The basic notion behind the algorithm is that if intensity of each object in a scene is assumed to be constant, then normal velocity is the time derivative of intensity divided by the intensity gradient. The implementation of this algorithm on PIPE is described in more detail in [Goldenberg et al. 87].

A more recent motion recovery algorithm [Waxman and Bergholm 87] relies on the motion of binary image features such as edges and points, but does not explicitly track features from frame to frame. Instead, each feature generates a Gaussian activation profile in a spatio-temporal neighborhood around the feature. This profile is then carried along with the motion of the feature. Image velocity estimates may be obtained for these features. In the case of edge features, only normal velocity is obtained. However, in the case of point features (e.g., dots, corners, high curvature points on contours), full image velocity is obtained. This algorithm has been implemented on PIPE. It returns 15 velocity updates per second.

There are several reasons why we would want to recover image motion. It could be used for image segmentation (e.g., extracting moving edges or moving regions), and to recover rigid body structure and motion of objects in the visual field [Ullman 79, Waxman and Wohn 87]. One example of the way we intend to recover 3-D shape from image motion is to apply motion extraction techniques to images obtained from a camera mounted on a robot (either a mobile robot or a manipulator). The camera motion (both translation and rotation) can be determined from sensors on the robot. Then, under the assumption that objects in the scene remain stationary, image flow can be used to recover 3-D positions of the points giving rise to the image flow. The intuitive idea behind this concept is that points further from the camera will appear to move less than points close to the camera.

An example of the use of image motion is to track an object. Image motion is used to extract the object from the background, while a tracking algorithm is applied to follow the object over time [Goldenberg et al. 87]. The algorithm, which has been implemented on PIPE, assumes that a single object is moving relative to a stationary camera. The difference of two successive images is obtained and thresholded, resulting in a region of motion. Each pixel in this image is either a 0 (no important change) or a 1 (important change). The next step involves computing the centroid of the region of motion. This centroid is then tracked over time. To calculate the x centroid, the binary motion image is multiplied by an X-Ramp -- an image where each pixel has a grey value equal to its x coordinate. The resultant image has, for each pixel in the region of motion, a grey value equal to its x coordinate. The x centroid is the average of all the non-zero pixels in the image. The y centroid is obtained in an analogous manner. These averages are computed using a pyramid-based adding strategy, which places the centroid coordinates in the upper left pixel of each image. This algorithm, when running on PIPE, results in centroid updates 5 times per second.

## 5. Inspection

Inspection of two-dimensional patterns can be done on PIPE in several ways. One way uses the backward pathway of PIPE and the input from the host machine. A blue print, or model, of the ideal appearance of the pattern is stored or generated on the host machine. This pattern is then transmitted to the PIPE which compares it with an incoming image. The comparison results in a difference image which is then analyzed to determine the degree of difference. A main problem that has to be overcome in this kind of application is the problem of registering the model with the image. The position and orientation of the object to be inspected would have to be known in advance (up to a small error) in order for this technique to be effective.

Another technique uses operators known as shrink and expand [Rosenfeld and Kak 82]. These operators are generally applied to thresholded binary images. The shrink operator involves deleting a layer of border points one pixel wide from each region, or blob, in the binary image. The expand operator involves adding a layer of border points to each blob. These operators are easy to implement on PIPE using the Boolean neighborhood operator capability. Generally, several shrink or expand steps are done in succession. We can define the thickness of a blob as twice the number of shrinking steps required to wipe it out. As an example, suppose we want to find all lines in an image that are six pixels wide or thinner. If we apply the shrink operator three times, then all lines six pixels or thinner will disappear. If we then apply the expand operator three times, then lines thicker than six pixels will appear the same as they appeared in the original image, while lines six pixels or thinner will not reappear. If a logical exclusive-or between this final image and the original image is done, then the resulting image will contain only those thin lines which disappeared after performing the shrink and expand operations.

Another inspection technique can be used to find imperfections in a regularly textured region, such as a weave pattern. The technique involves applying a high-pass filter to the image, so that only features whose spatial frequency is higher than that of the regularly textured pattern are extracted. These features might then be imperfections. A high-pass filter can be implemented on PIPE by performing a smoothing operation (e.g., Gaussian smoothing) and then taking the difference between the smoothed image and the original image.

## 6. Conclusion

The PIPE (Pipelined Image Processing Engine) has been described. It is an image processing device that was designed for real-time robot vision applications. Three applications that might have relevance for scanning microscopy have been described.

These are stereo analysis for extracting range from two images, motion analysis for detecting, measuring and tracking motion and for extracting three-dimensional information, and inspection of two dimensional patterns. Future work on PIPE will include developing more sophisticated and more effective algorithms to perform these tasks, and integrating PIPE with a symbolic host processor so that high-level symbolic vision can be combined with the low-level image processing.

**References**


1. Barnard, S.T. and Fischler, M.A. "Computational Stereo." *Computing Surveys*, Vol. 14, No. 4, December 1982, 553-572.

2. Drumheller, M. and Poggio, T. "On parallel stereo." *IEEE International Conf. on Robotics and Automation*, San Francisco, CA, April 1986, 1439-1448.

3. Goldenberg, R., Lau, W.C., She, A. and Waxman, A.M. "Progress on the prototype PIPE." *IEEE International Conf. on Robotics and Automation*, Raleigh, NC, April 1987,

4. Horn, B.K.P. and Schunk, B.G. "Determining optical flow." *Artificial Intelligence*, 17, 1981, 185-203.

5. Kent, E.W., Shneier, M.O. and Lumia, R. "PIPE (Pipelined Image Processing Engine)." *J. Parallel and Distributed Computing*, 2, 50-78, 1985.

6. Rosenfeld, A. and Kak, A.C. *Digital Picture Processing*. Second Edition, Academic Press, Orlando, Florida, 1982.

7. Ullman, S. *The interpretation of visual motion*. MIT Press, Cambridge, MA, 1979.

8. Waxman, A.M. and Bergholm, F. "Convected activation profiles and image flow extraction." Technical Report LSR-TR-4, Laboratory for Sensory Robotics, Boston University, August 1987.

9. Waxman, A.M. and Wohn, K. "Image flow theory: A framework for 3-D inference from time-varying imagery." In C. Brown, ed. *Advances in Computer Vision*, Erlbaum Publishers, 1987.


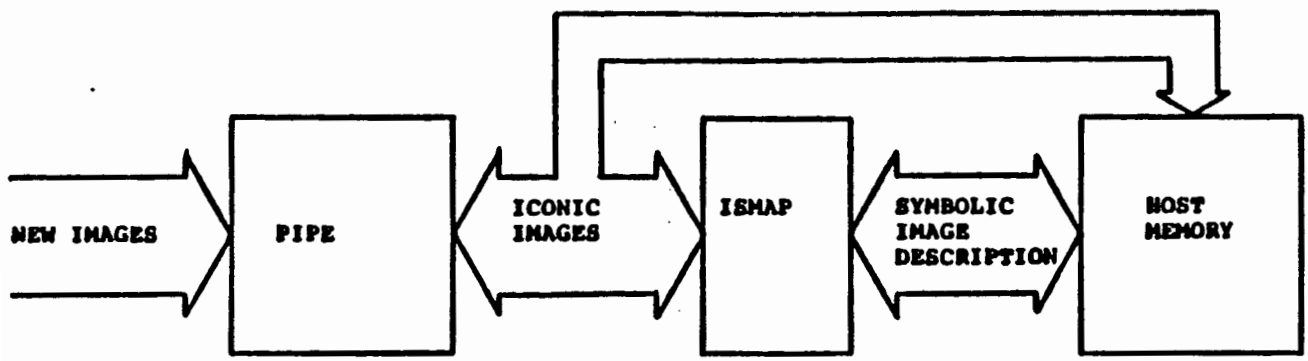


Figure 1: Major image-flow relationships between PIPE, ISMAP, and the host computer.

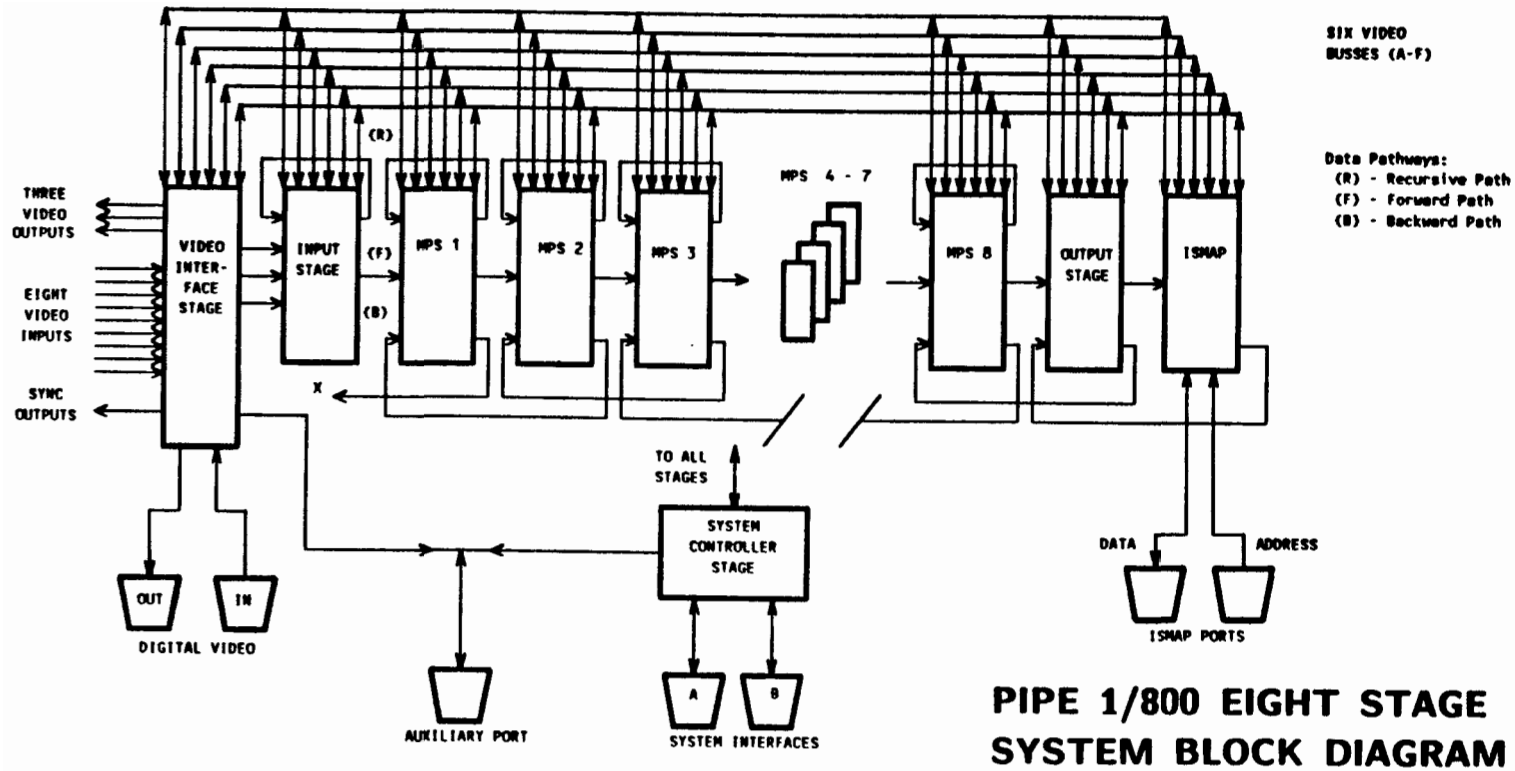


Figure 2: PIPE system block diagram. (Reprinted with permission from Aspex Incorporated.)

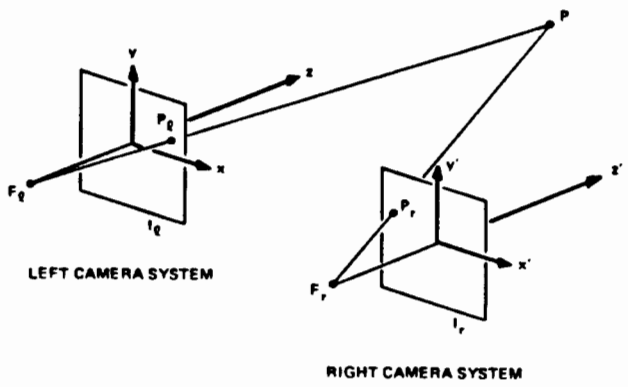


Figure 3: Stereo vision system.