

# PARALLEL IMPLEMENTATION OF THE HOUGH TRANSFORM FOR THE EXTRACTION OF RECTANGULAR OBJECTS

LeAndra Hopwood, Winston Miller, and Alan George  
*High-performance Computing and Simulation (HCS) Research Laboratory*  
Electrical Engineering Department, FAMU-FSU College of Engineering  
2525 Pottsdamer Street  
Tallahassee, Florida 32310

**Abstract** - In image processing applications, the storage capacity required for images can exceed feasible storage capabilities. A technique to alleviate this problem by removal of unnecessary background information through image processing is discussed. Specifically, a parallel implementation of a first-order, derivative-based edge detection algorithm and the Hough transform applied to rectangular objects is given. A variation of the classical Hough transform to detect lines is employed to locate rectangular objects of known size in an image. Parallel Virtual Machine is used to exploit the inherent parallelism found in these algorithms over a cluster of 7 workstations. Through the use of these techniques, the rectangular object is detected and stored as a separate image, and storage capacity can be reduced by approximately 30%, not including standard data compression. Parallelizing the algorithms provides a significant speedup advantage over the normal sequential operation of the programs.

## INTRODUCTION

The increasing move towards real-time systems to capture, store, and transfer digital images for archival and verification purposes creates a growing need for file size reduction techniques. Image segmentation and geometric classification may be employed as front-end processing to aid in the minimization of file sizes by discerning between the background and relevant portions of the image. This paper examines the use of these techniques to reduce the storage requirements of digital images of special-purpose packaging captured on an assembly line. The segmentation process involves removing “background” information and storing the remaining information in a separate, smaller file. A parallel program was developed using Parallel Virtual Machine (PVM), a parallel language based on C, that first detects the edges of the image and then applies a specialized Hough transform (HT) to identify the rectangular package. Because of the computationally-intensive algorithms inherent to edge detection and the HT,

parallelizing these techniques yielded performance speedup.

The topics discussed in this paper include an introduction to edge detection, the HT, parallel aspects, an overview of the results, and opportunities for future research.

## EDGE-DETECTION TECHNIQUES

Edges are positions in an image where there is an abrupt change, or sharp discontinuity, in some visual property normally in a gray-level image. Computations for edge detection are performed on a pixel by pixel basis, with many arithmetic operations performed on each pixel. The complete detection of edges in a gray-level image is generally employed in three steps. First, the image is convolved with a derivative mask (operator mask) that produces a measure of intensity gradient. Second, a threshold operation is applied in which points contributing to edges are identified as those exceeding a set level of intensity gradient values. Third, the edge points are combined into coherent edges by applying a linking algorithm [5].

The fundamental concept in most edge-detection techniques is the computation of the local derivative operator which is designed to estimate spatial gradient that in turn identifies the edges. The magnitude of the first-order derivative detects the presence of edges in an image and is obtained by calculating the gradient  $f(x,y)$  at location  $(x,y)$ . Edge detection by gradient operations works fairly well with images that have sharp intensity transitions and low noise. The gradient and magnitude in the digital domain are obtained by convolving the image with two masks in the  $x$  and  $y$  directions. Some common masks used for gradient edge detection are the Sobel, Prewitt, and Roberts Cross [1]. For this study, the Sobel masks, illustrated in Figure 1, were chosen because of their smoothing and differencing effects.

The second-order derivative, or Laplacian, of a 2-D image has a zero-crossing at the midpoint in any intensity change and, unlike the first-derivative detection, indicates the direction of change. Although the second derivative is

unacceptably susceptible to noise, it is more effective than the gradient operator for identifying the location of edges in an image which contains blurred edges with somewhat low noise [3,5].

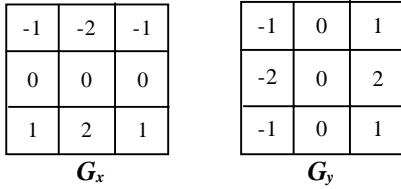


Figure 1. Sobel Convolution Masks

### THE HOUGH TRANSFORM

The Hough transform has been a popular technique for detecting geometric primitives in images since it was first developed in 1962 by P. V. C. Hough [4]. By performing calculations on the pixels, a parameter space consisting of accumulator cells, usually initialized to zero, is created and filled according to the points which satisfy the equations describing the desired shape. In other words, if the image point satisfies the specified equation(s), then the corresponding cell in the parameter space is incremented. The dimensions of the parameter space grow exponentially with the number of unknown parameters in the descriptive equations. Once the parameter space is complete, a thresholding operation determines the desired information.

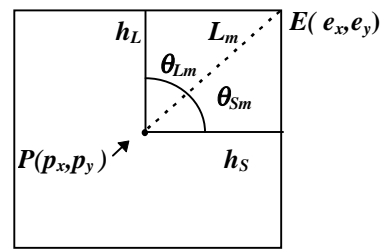
The goal in using the HT in this study is to extract the rectangular package from the digital image captured on an automated assembly line. The first step in using the HT to detect the rectangle is to edge detect the image. The resulting binary image may be noisy, and contain incomplete edges of individual objects and unwanted additional edges located within the package area itself. Therefore, the edge-detected image itself is not sufficient for the rectangle detection.

The next step in locating the rectangle is to apply the HT algorithm to the edge-detected image. Two possible techniques to locate the package are the line method and the sweep method. The line method treats each edge of the rectangle as individual lines and locates them to reformulate the rectangle. However, it becomes more challenging in a noisy picture to identify which lines actually belong to the edges of the rectangle, especially if the edges are not complete. The parallel decomposition of this method over a network of workstations calls for unacceptable interprocessor communication requirements since executing processors must continuously check the relationship of its detected lines with others.

The sweep method deduces a set of equations to produce the center of the rectangle based on edge pixels.

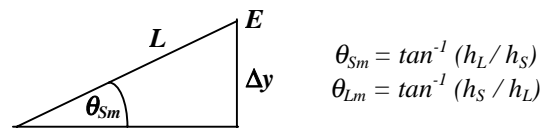
Since the general dimensions of the package are known, the package can be identified by finding the center, then using the dimensions to reconstruct the borders. Although computationally-intensive, the mathematics and logistics of the algorithm are easier to apply than the line method. Also, little interprocessor communication is necessary between processing elements. As packages come down the assembly line, their orientations may vary slightly, which presents a problem for the algorithm. The development section of this paper indicates how this limitation was sidestepped.

Figures 2 and 3 illustrate the formulation of the equations used in the program for locating the center of the package.



- $h_L$ : length from center to horizontal edge
- $h_S$ : length from center to vertical edge
- $\theta_{Sm}$ : maximum angle from  $h_S$  to  $L_m$
- $\theta_{Lm}$ : maximum angle from  $h_L$  to  $L_m$
- $P$ : represents center point of rectangle
- $L_m$ : vector of varying length beginning at center of rectangle and terminating on edge at point  $E$

Figure 2. Rectangle Illustrating Dimension Variables



$$\theta_{Sm} = \tan^{-1}(h_L / h_S)$$

$$\theta_{Lm} = \tan^{-1}(h_S / h_L)$$

$$p_x = e_x - h_s$$

$$p_y = e_y - \Delta y = e_y - h_s \tan \theta_s,$$

$$\text{for } \tan^{-1}(h_S / h_L) \geq \theta_s \geq -\tan^{-1}(h_S / h_L)$$

Figure 3. Triangular Section of Rectangle

As  $L_m$  is swept from  $\theta_{Sm}$  to  $-\theta_{Sm}$ , the cells in the parameter space that correspond to edge points satisfying  $p_x$  and  $p_y$  are incremented. The same method can be applied from  $\theta_{Lm}$  to  $-\theta_{Lm}$ . In summary, all possible values for the center of the rectangle are:

$$p_x = x_x \pm h_s$$

$$p_x = x_x + h_L \tan \theta_L, \quad |\theta_L| \leq \theta_{Lm}$$

$$p_y = x_y \pm h_L$$

$$p_y = x_y + h_s \tan \theta_S, \quad |\theta_S| \leq \theta_{Sm}$$

These equations produce a high value in the accumulator cell(s) corresponding to the center of the rectangle. The center gives the reference point at which the known dimensions  $h_S$  and  $h_L$  may be applied to locate and copy the package information from the original file to the new file.

### PARALLEL ASPECTS

The edge-detection algorithm and the HT present an encouraging prospect for parallel processing. Little interprocessor communication is necessary to implement the sweep method of the HT in parallel. Although some information is lost from parallelizing the edge detection, the robustness of the HT allows this loss to be negligible.

The algorithm naturally uses a master-worker paradigm. A flow chart of the responsibilities of the master and workers is provided in Figure 4. The advantage of the non-working master, which is used in this case, is that as soon as a worker sends its results, the master can almost immediately receive and evaluate the results. To implement the algorithm, the image is divided evenly to give each processor the same amount of information. The image is divided into rows with the same number of columns to provide an easily scalable algorithm.

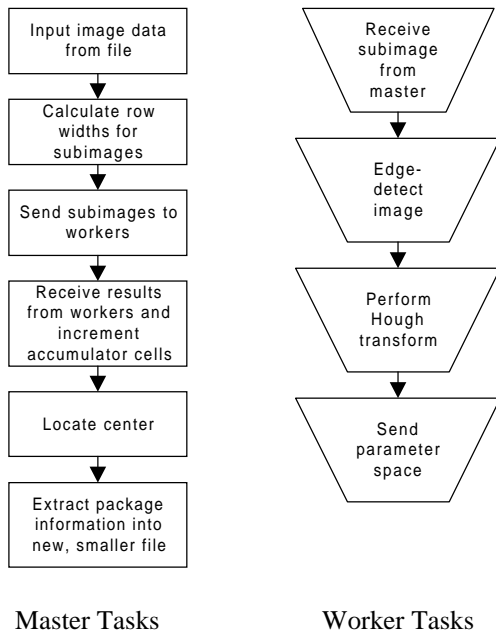


Figure 4. Reduction Algorithm Flowchart

PVM is a programming tool developed by and available from Oak Ridge National Laboratory that permits the use of a collection of heterogeneous processors as a single parallel virtual machine. PVM handles the

message routing, data conversion, and task scheduling for a network of heterogeneous processors and allows the users to assign particular tasks to specific computers. The principles that PVM is based upon include a user-configured host pool, translucent access to hardware, process-based computation, explicit message-passing model, heterogeneity support, and multiprocessor support. A parallel version of an optimized sequential program can be created by incorporating the PVM constructs into the sequential program [2].

### RESULTS

The largest test image was of a package of size 742 x 513 pixels illustrated in Figure 5. The packages were captured in 24-bit color using a scanner and converted to grayscale for the edge detection.

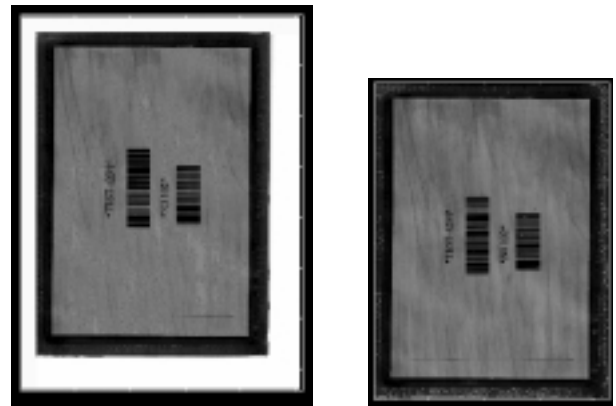


Figure 5. Original Image and New Image

The  $h_S$  and  $h_L$  values of this particular package were 213 and 322, respectively, giving a center at row 353 and column 240. An interesting aspect of this image was that the center of the rectangle actually fell between pixels, meaning that more than one center was possible. To account for instances like these, and for the chance of the package being slightly misaligned under the capture device, the extracted file includes extra white pixels on each edge of the package. Since these pixels are redundant, a good compression scheme will be able to represent them in only a few bytes. Figure 6 illustrates the final parameter space produced by the HT.

The first objective of the program was to reduce the image size. The original image was 3,045,168 bytes, and the resulting file was 2,203,320 bytes, or 70% of the size of the original image. This 30% savings does not include any compression. This clearly illustrates that the objective was achieved. In fact, higher savings are expected, depending on the assembly-line configuration. For example, if the multiple-sized packages are all captured at

the same point by the same device, then the small packages will have much more background than that of the image tested here. The capture platform will be at least as large as the biggest package that can be produced, which adds background to the smaller packages. The output image after the reduction algorithm is shown in Figure 5.

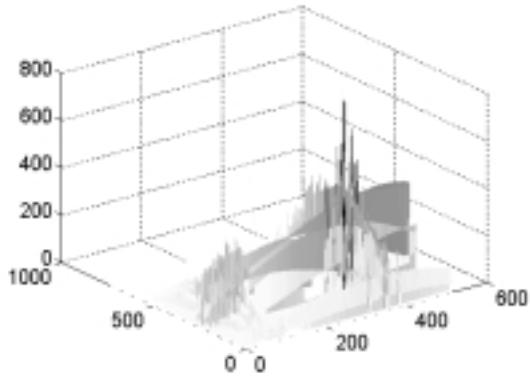


Figure 6. Parameter Space of Transformed Package

The second objective was to improve the speed of the algorithm by implementing it on a network of workstations working in parallel. The master used was a 32MB SPARC-10/40 with 6 workers, each a 32MB SPARC-5/85. The speedup is illustrated in Figure 7.

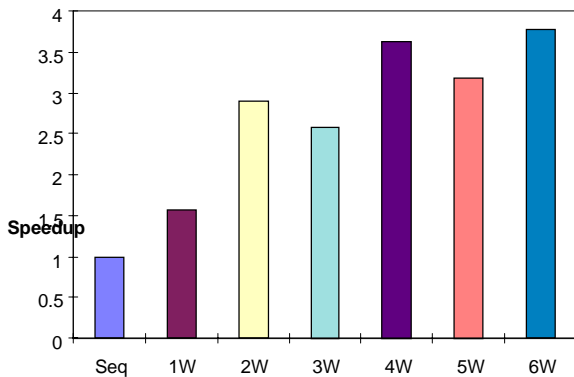


Figure 7. Speedup of Reduction Algorithm (Seq = sequential, 1W = one worker PVM, etc.)

The discontinuities in the speedup occur when an odd number of workers is specified. The method of partitioning dictates how the portions that have more information to be processed are handled. When the image is split evenly, at least two processors share the portion of the image that requires the most processing. However, when the image is divided into an odd number of partitions, one processor may be given the most intensive part of the image, requiring it to process three or four times the amount of information as the other processors. One possible solution is to divide the image such that

every node is given two sections to process, distributed in an interleaved manner. The theory behind this concept is that by interleaving the rows, each processor will ideally receive one subimage that is intense in calculations and another that is not as intense.

Another possibility involves estimating the location of the package. Since the assembly line will generally place the package in the same location on the capture platform each time, the program can assume where most of the processing will occur and divide the image accordingly. The advantage of this algorithm is that it is specialized for certain scenarios and is most likely to provide the fastest execution times of all the partitioning methods thus far. The disadvantage is that *a priori* knowledge is required to implement such an algorithm, and the different packages would each have to have their own capture platform for optimal results. The probability of knowing this information is high though since it is on an automatic assembly line.

## CONCLUSION

The HT reduction algorithm can provide a savings of over 30% in storage space before the application of data compression. This reduced storage requirement is essential for imaging systems that store gigabytes of information daily. With the combination of a good lossless compression format, the images can be compressed to a size comparable, if not better, than that of a lossy compression scheme. With lossy compression, the combination can achieve a file reduction that no compression scheme alone is able to provide.

By decomposing, partitioning, and mapping the Hough transform algorithm onto a cluster of workstations, near-linear speedup can be attained for certain configurations. Further research may reveal methods of interleaving data sets to increase speedup even further.

## REFERENCES

- [1] Fisher, B., et. al., "Edge Detectors," Department of Artificial Intelligence, University of Edinburg, UK, 1994.
- [2] Geist, A., et.al., *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing*, MIT Press, Cambridge, MA, 1994.
- [3] Gonzalez, R. and R. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA.
- [4] Hough, P.V.C., "Methods and Means for Recognizing Complex Patterns," U.S. Patent 3,069,654; 1962.
- [5] Marshall, S., *Parallel Algorithms for Digital Image Processing, Computer Vision, and Neural Networks*, J. Wiley & Sons, Chichester, England, 1993.

## ACKNOWLEDGMENTS

We would like to thank Mr. Javier Medina, formerly of the HCS Lab, for the sweep method idea and his time spent in assisting with this research, Mr. Ryan Fogarty for his time spent discussing ideas, and most of all the DOD for making this research possible.