# Using Parallel Computing Methods to Improve Log Surface Defect Detection Methods

**R. Edward Thomas**
U.S. Forest Service, Princeton, WV, USA. ethomas@fs.fed.us

**Liya Thomas**
Pearisburg, VA USA

## Abstract

Determining the size and location of surface defects is crucial to evaluating the potential yield and value of hardwood logs. Recently a surface defect detection algorithm was developed using the Java language. This algorithm was developed around an earlier laser scanning system that had poor resolution along the length of the log (15 scan lines per foot). A newer laser scanning system was constructed that had much greater resolution (192 scan lines per foot) along the logs' length. The increased resolution and the slower processing speed of the Java-based algorithm required a new approach. The revised algorithm was designed around the higher resolution data and employs parallel processing technology. The new algorithm processes higher resolution data in less time than required by the original algorithm using the lower resolution scan data. The improved processing power permits a more in-depth analysis of the higher resolution scan data scan data leading to improved detection results.

Keywords: hardwood, log, defect, automated detection, parallel processing

## Introduction

For decades hardwood log processing optimization methodologies and techniques have been one of the top priorities in the wood products industry (Chang 1992). After a tree is harvested, it undergoes a rough manual estimation of potential yields and is bucked into several logs. Before a log is sawn into boards, the operator quickly examines it and picks the best opening or highest quality face. During the examination, surface defects are identified, and type, severity, and topological distribution of the defects are determined. Thus, lumber yield is highly subject to time constraints, operator's skill, and conditions at the time of evaluation. Hardwood log defects vary in size, shape, and type, making them difficult to identify. Defects, such as sawn-knot stubs, rotten knots, surface distortion, holes, cracks, gauges, branch sprouts, swelling, worm holes, and animal damage, are imperfections that decrease wood value and utility.
With the advances in scanning equipment, it is possible to develop automated systems that aid the bucking and sawing process. Researchers have experimented with technologies and methods that locate and classify internal as well as external defects on either softwood or hardwood. Systems including those using X-ray/ CT (Computerized Tomography), X-ray tomosynthesis, MRI (Magnetic Resonance Imaging), microwave scanning, ultrasound, and ground penetrating radar have been researched and developed to detect internal defects (Wagner et al. 1989, Guddanti et al. 1998, Zhu and Beex 1994, Devaru et al., Chang 1992, Nordmark and Oja 2004). Advanced computer vision and image processing methods and algorithms were proposed for internal hardwood log defect inspection (Zhu and others 1996, Li and others 1996, Tian and Murphy 1997a, Tian and Murphy 1997b, Kline and others 1998, Bhandarkar and others 1999). Some of these methods and technologies provide high detection rates as the data have extremely high resolution. However, issues such as high cost, low processing speed, data instability, and environmental safety keep them from being commercially available. Another approach is the use of laser

scanning technologies to collect profile data of softwood and hardwood logs. Thomas et al. (2006, 2007) developed a log surface defect detection system, which uses high-resolution laser scan data. In tests of this system, 53 severe defects were detected out of 64 with 9 non-defective regions falsely identified. When calculated as areas, 90.2 percent of the defects were located.

All of these methods required the processing of vast amounts of data, some more than others. In addition, processing the data needs to occur in real time to be of use in a processing environment. As scanners and data collection systems become more advanced, the detail and volume of the data increases, ultimately requiring greater processing resources to yield the desired higher accuracies. Further, successfully solving these problems often exceeds the capabilities of the algorithms used. Most computers sold today have at least two processors, with many having four or more processors. Designing algorithms to take advantage of increased computing power is a must to be able to solve these problems in real time.

## Methods

### Data Collection

The United States Forest Service constructed a high-resolution three-dimensional laser-scanning system for logs in Princeton, West Virginia. The system has three industrial laser scanning heads designed for the wood processing industry. The scan heads are arranged in an eight foot circle with a 120 degree separation between the scan heads (Figure 1). This allows the three scanners to collect a complete surface scan of the log surface. The log is supported in V-stands every 5 feet at the center of the circle of scanners (Fig. 1). The scanner then over the log and collects a scan line around the circumference of the log every 1/16 inch. Resolution between points within each scan line varies depending on the size of the log, but is typically around 1/8 inch. All points are measured to the nearest 0.001 inch. A dot-cloud image sample of a scanned log is shown in Figure 2. The two vertical white marks in Figure 2 are missing data due to shadowing of the log surface by the V-Stands.

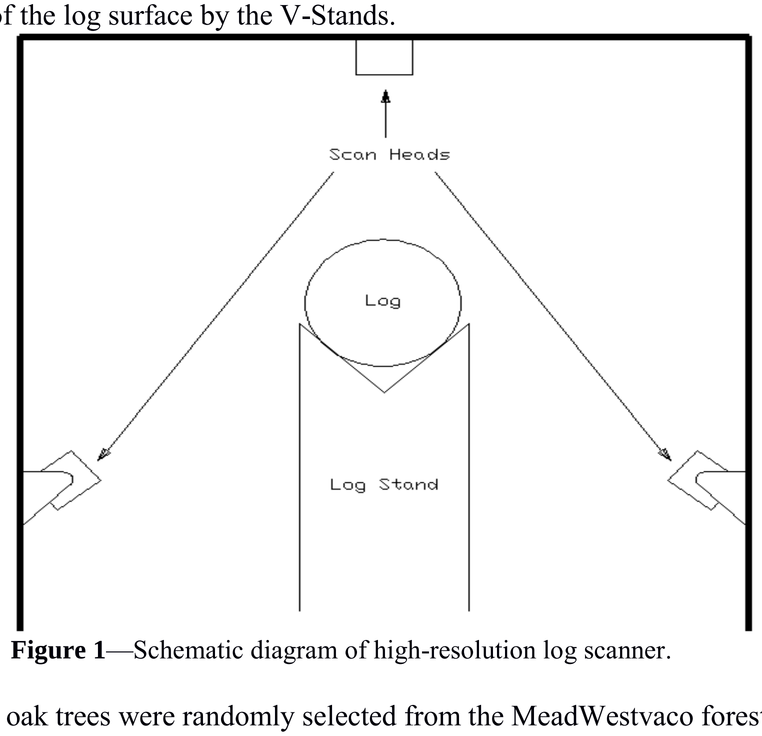


**Figure 1**—Schematic diagram of high-resolution log scanner.

Thirty-three red oak trees were randomly selected from the MeadWestvaco forest located near Rupert, West Virginia. Each tree was was bucked into log lengths and the logs were laser-scanned then manually diagrammed, where the location and type of the defects were recorded. Ten logs were randomly selected for this study from a set of 33 logs. The specifications of the selected logs are listed in Table 1.

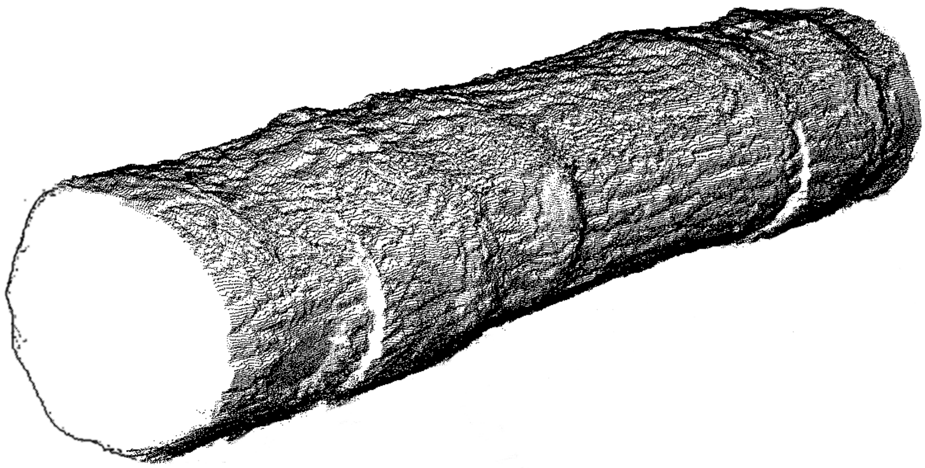

**Figure 2**—Dot cloud view of log 31C.

Table 1.— Study Log Specifications.

| Log | Length (inches) | Large End Diameter (inches) | Small End Diameter (inches) | Total Severe Defects | Total Non-Severe Defects | Total Defects |
|---|---|---|---|---|---|---|
| 16D | 125.8 | 17.1 | 17.0 | 5 | 7 | 12 |
| 27B | 127.1 | 20.7 | 19.1 | 5 | 0 | 5 |
| 25E | 152.9 | 15.3 | 13.1 | 18 | 1 | 19 |
| 31C | 129.3 | 16.8 | 15.7 | 11 | 6 | 17 |
| 7A | 126.1 | 20.5 | 17.2 | 1 | 1 | 0 |
| 6B | 122.4 | 17.7 | 12.4 | 2 | 1 | 3 |
| 33B | 128.9 | 17.8 | 17.3 | 4 | 3 | 7 |
| 11C | 158.2 | 14.4 | 13.3 | 9 | 2 | 11 |
| 21C | 116.0 | 18.3 | 13.9 | 4 | 4 | 8 |
| 3C | 126.8 | 14.5 | 14.1 | 4 | 9 | 13 |
| **Average** | 131.3 | 17.3 | 15.3 | 6.3 | 3.4 | 9.5 |

## Sequential Detection Algorithm

The high resolution three-dimensional (3-D) laser-scanned hardwood log data shows minute surface variations, making it feasible to locate external defects. Four major steps are involved:

1) Fit 2-D circles to log cross-sections. Because the scans may be missing data or includes irrelevant deviant data points; a new, robust estimator was developed to reliably estimate the centers and the radii of the fitted circles.
2) Extract radial distances between the fitted circles and log data to indicate local height changes. Radial distances refer to height values on the surface with respect to reference.
3) Locate defects characterized by significant surface rises or depressions.
4) Filter and refine detection results using an expert system.

Robust statistical estimators provide a mechanism to correctly estimate model parameters in the presence of severe outliers (data with large deviants). A new robust non-linear generalized M-estimator (GME) was developed whose objective function makes use of scale estimates (Thomas and Mili 2007). These are calculated using projection statistics and incorporated in the Huber objective function, such that the influence function of the estimator is bounded. This makes it possible to filter out not only the errors in the measurements, but also the errors in the circle model that is applied to a given cross-section data set.

Radial distances are computed between the virtual log produced from the fitted 2-D circles and log data. The radial distances are orthogonal, and as a whole provide a height map of the log surface with respect to the virtual log. These radial distances typically range from -0.5 to 0.5 inch, scaled to range from 0 to 255 gray-levels, and mapped to create a 2-D image (fig. 3). In this process, the log surface is unrolled onto a 2-D coordinate space to create a “skin” representing the pattern of log bark along with bumps and bulges associated with most defects. Next, the log data are processed and interpolated linearly to fill any gaps between data points. This enables our system to locate defects associated with unique characteristics and height change. The defect detection algorithm analyses both contours and 3-D data to locate severe defects at least 0.5 inch high and 3 inches wide.

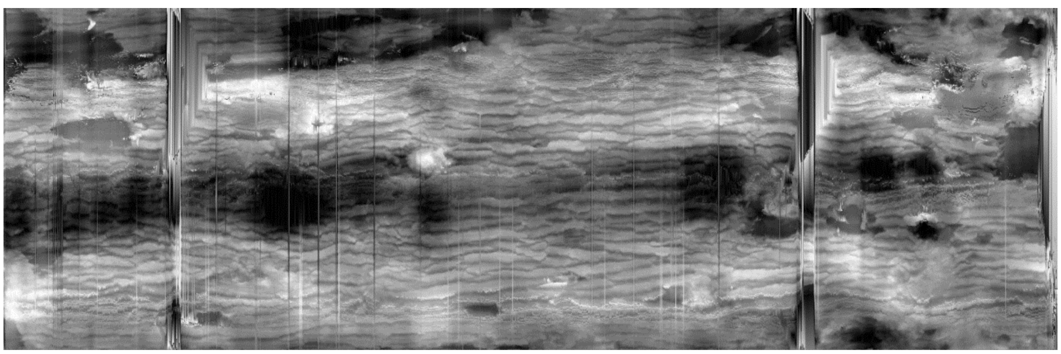

**Figure 3**—Residual image of log 31C created using the Java algorithm

Six contour levels are computed from the orthogonal distances between a virtual log and any point of the cross section, generating a topographical exterior log map. Examining only the highest and lowest-level contours, defects corresponding to surface rise or depression are detected. Expert knowledge, in the form of defect shape and size characteristics, are applied in a stepwise fashion to rule out regions as potential defects, including regions with sizes smaller than a given threshold, nested in other curves, or long and narrow (determined by the actual width to length ratio of real log dimensions). Some regions are removed for further consideration if they contain a large portion of missing data. The algorithm extends boundaries of an identified region only if a relatively high standing corner of the defect is located. Regions may include an elevated yet non-defective log surface. Typically these regions are covered with tree bark, and thus are associated with distinctive bark patterns.

Using radial distances visualized by the gray-level image presented in Figure 3, the algorithm generates a contour plot and determines the rectangle-enclosed regions of the highest level bounding contour curves that identify potential defects. Some regions are selected if they are large enough (at least 3 inches in diameter) or with a significant height (at least 0.5 inch). The region surrounding selected small regions are examined for the presence of relatively straight-line segments. If the coverage of straight line segments is sufficient, the defect region is adjusted to cover the entire defect surface, rather than just a corner. Such regions are usually the top of a sawn knot, either sound (not rotten), or unsound (rotten). Region-removal rules are given as: regions smaller than a given threshold; regions enclosed in curves nested in other curves are removed, as there will only be one defect in the same location. Long and narrow regions are normal bark patterns and are ignored. Also ignored are regions smaller than 50 square inches and adjacent to the selected large regions. The marked squares in Figure 3 identify areas with defects.

## The Parallel Algorithm

The parallel version of the defect detection algorithm performs the image processing functions and contour analyses in parallel. This version was developed using the following software packages:

a) gcc version 4.6.1 (C Compiler)
b) MPICH2 1.4.1 (Message Passing Interface for parallel application development) (Gropp 2002)
c) OpenCV 2.1.0-7 (Open Source Computer Vision library) (Bradski 2000)

The parallel defect detection algorithm can be built and run using Windows 7[1], a standard C compiler and the Windows versions of the OpenCV and MPICH packages. The test system for the parallel and Java algorithms was conducted on an IBM compatible computer with 24GB of RAM and 8 3.6 Ghz Core2 CPUs running the Mint Linux operating system.

MPICH2 is one of many different programming packages available for developing parallel applications. MPICH2 is one of the most popular implementations of MPI and is available for many operating systems and computer architectures (Pacheco 1997). In addition, MPICH2 is the foundation of several corporate versions of MPI. MPI and MPICH2 allow developers to specify how to divide a processing task such that the different parts of the task are solved concurrently on different processors of the same computer or on processors of other networked computers. There are two common ways of distributing processing across multiple processors. The first method segments the problem into equal pieces and gives a piece to each processor available. Once, each piece is processed, the results are compiled together. In the second approach one processor is designated as the master controller. The master then divides work and collects results from the “slave” processors as work is completed. The parallel application here used the master-slave processing paradigm. In either method, the data to be processed and the results that are collected, are transferred using messages. Hence the name “Message Passing Interface”. If one has eight processors, you would have one master and 7 slave processors. Thus, theoretically a parallel-processed job could be solved in $1/7^{th}$ of time required by a single processor algorithm. However, due to the overhead of message passing, the required coordination of processing among all processors, and the portions of the computer program that do not run in parallel, the processing time reduction is always less. How much less depends on maximizing the parts of the original program to run in parallel.

Of the four major defect detection steps, we have concentrated on improving the processing speed of the first two steps. This is because that most of the tasks involved in these steps involve many complex independent operations that are ideal for parallel processing. The first processing step, fitting 2-D circles to log cross-sections, is divided into four smaller tasks:

1) Read in log data.

---

[1] Windows 7 is a registered trademark of Microsoft, Inc. The use of trade, firm, or corporation names in this publication is for the information and convenience of the reader. Such use does not constitute an official endorsement or approval by the U.S. Department of Agriculture or the Forest Service of any product or service to the exclusion of others that may be suitable.

2) Determine fitting parameters for each log data slice.
3) Box filter fitter parameters between slices.
4) Remove outlier data from each slice.

The second processing step, extracting radial distances between fitted circles and log data to indicate local height changes, is divided into 3 separate tasks:

5) Create image strip for each slice.
6) Fill in missing data “holes” for each slice.
7) Residual image creation from individual image strips.

Of the seven tasks, tasks 2, 4, 5, and 6 are ideally suited for parallel processing. These tasks involve complex processing of individual slices of the laser scan data. More importantly, these operations are independent, (ie. they do not depend on the results of any neighboring slices). To reduce message overhead, the data for 5 slices at a time is transmitted to each processor. When a processor finishes, it transmits the results back to the master processor. The slave processor then receives the next set of 5 slices, if any are remaining. The master processor is responsible for keeping track of which data slices each slave is processing and for collating the results as it receives them. The remaining tasks 1, 3, and 7 are sequential in nature and are not suitable for parallel processing.

The third step, locating defects characterized by significant surface rises or depressions, uses OpenCV to create and analyze the contour data. This library was not written to take advantage of parallel processing, thus this part of the defect detection code is run as a single process. The expert system is the final step that is performed by the Java-based algorithm and is used to reject some areas marked as defective, which are likely not defective. This is accomplished by examining the shape characteristics of the marked defect region to judge if the area is truly a defect. The expert system in the parallel processing version examines only a subset of the conditions that can render an area marked as defective, not truly a defect.

## Results

To determine the parallel algorithm advantages and disadvantages compared to the original Java algorithm, the algorithms were run on each of the sample logs and the individual results compared. The specifications of these test logs are described in Table 1. The logs processed are a mix of lower and upper story logs with an average length of 131.3 inches and an average large end diameter of 17.3 inches and a small end diameter of 15.3 inches. Overall there was an average of 6.3 severe defects per log. Defect types considered severe are overgrown, sawn, and unsound knots and clusters, as well as bumps, and holes. In addition, there was an average of 3.4 non-severe defects per log, which includes distortion and adventitious knot defect types.

Tables 2 and 3 contain detection results from the parallel and Java detection algorithms, respectively. Table 4 compares the performance of the two detection methods. Overall, the parallel method had fewer un-detected severe defects, 2.4 versus 3.3 of the Java algorithm. This can be seen in Table 4 where the parallel method had better detection rates in 6 of the 10 logs processed. However, the Java algorithm resulted in significantly fewer false positive defect detections per log, 1.9 versus 4.7 of the parallel method. This difference is due to the Java algorithm's expert system which refines detection results. Perhaps the most significant results in the comparison is processing time. The average processing time per log of the parallel method was 1.009 seconds compared to 49.973 seconds for the Java-based algorithm. In two instances the Java method took longer than one minute to complete, while in five instances the parallel method required less than one second to process. Thus, the parallel method yields some accuracy, for a greatly decreased processing time.

Table 2—Parallel algorithm detection results.

| Log | Total Severe Detected | Total Non-Severe Detected | False Positive Detected | Total Severe Undetected |
|---|---|---|---|---|
| 16D | 5 | 0 | 4 | 0 |
| 27B | 2 | 0 | 3 | 3 |
| 25E | 11 | 0 | 3 | 7 |
| 31C | 6 | 1 | 2 | 5 |
| 7A | 0 | 1 | 10 | 1 |
| 6B | 1 | 1 | 6 | 1 |
| 33B | 2 | 0 | 5 | 2 |
| 11C | 6 | 0 | 5 | 3 |
| 21C | 3 | 0 | 2 | 1 |
| 3C | 2 | 1 | 7 | 1 |
| **Average** | | | 4.7 | 2.4 |

Table 3—Java algorithm detection results

| Log | Total Severe Detected | Total Non-Severe Detected | False Positive Detected | Total Severe Undetected |
|---|---|---|---|---|
| 16D | 3 | 0 | 1 | 2 |
| 27B | 3 | 0 | 1 | 2 |
| 25E | 5 | 0 | 4 | 6 |
| 31C | 2 | 1 | 0 | 9 |
| 7A | 0 | 0 | 2 | 1 |
| 6B | 2 | 1 | 1 | 0 |
| 33B | 2 | 0 | 5 | 2 |
| 11C | 3 | 0 | 0 | 6 |
| 21C | 2 | 1 | 3 | 3 |
| 3C | 1 | 1 | 2 | 2 |
| **Average** | | | 1.9 | 3.3 |

Table 4—Algorithm performance comparison.

| Log | Parallel Algorithm Processing Time (seconds) | Java Algorithm Processing Time (seconds) | Parallel Severe Detection Rate (percent) | Java Severe Detection Rate (percent) |
|---|---|---|---|---|
| 16D | 1.041 | 56.500 | 100 | 60 |
| 27B | 0.892 | 67.910 | 40 | 60 |
| 25E | 1.146 | 58.959 | 61 | 28 |
| 31C | 1.943 | 64.075 | 55 | 18 |
| 7A | 0.853 | 42.534 | 0 | 0 |
| 6B | 1.028 | 34.559 | 50 | 100 |
| 33B | 0.689 | 59.891 | 50 | 50 |
| 11C | 1.003 | 42.069 | 67 | 33 |

| | | | | |
|---|---|---|---|---|
| 21C | 0.701 | 38.248 | 75 | 50 |
| 3C | 0.794 | 34.985 | 50 | 25 |
| **Average** | 1.009 | 49.973 | 55 | 42 |

Figure 4 is the residual image of log 11C (see Figure 2 for dot cloud view) with the defects as detected using the parallel algorithm. Note that the log support stands created shadows or disruptions on the scan which are visible as two ")" shaped areas on the right edge of the image. These areas were detected as defects and are regarded as false-positive errors.

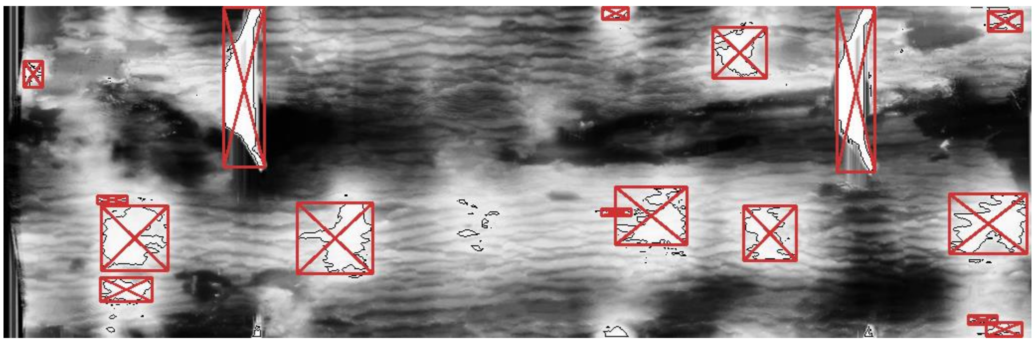

Figure 4—Residual image of log 31C with contour levels and defects identified using the parallel algorithm.

The Java algorithm's expert system is very complex, and is the main reason for the algorithm's low false-positive detection error rate: 1.9 defects per log versus 4.7 defects per log of the parallel algorithm. Approximately 33 percent of the java algorithm's processing time occurs while running the expert system, which on average is about 26 seconds per log. Thus, the expert system also would need to be developed as a parallel algorithm to keep processing times low. The parallel algorithm has a filter that can exclude contours with too many or too few data points. Reducing the parameter to exclude contours with larger contour sizes, resulted in fewer false-positive and detected defects. Increasing the parameter to include the larger contour sizes, resulted in more false-positive as well as more detected defects. The current settings of the system capture most defects, at the cost of greater false-positive.

## Summary and Conclusion

The parallel algorithm provided greater detection accuracy on some logs compared to the Java algorithm. This was evident on six of the ten logs processed in the comparison study. However, this was at the cost of greater false positive defect detection where the parallel algorithm indicated that there were almost three additional defects per log, on average, which were not present. In addition, each algorithm detected a total of four non-severe defects on the log sample, but not always on the same log. The greatest advantage of the parallel algorithm is detection speed, on average approximately one second is needed by the parallel algorithm, compared to 49 seconds of the Java algorithm.

The greatest need at present is the development of a parallel expert system to filter and enhance the results of the parallel algorithm. This would greatly reduce the number of false positives as this step in the java algorithm eliminates most false positive results. The expert system also determines which areas are most likely defective and thus improves detection accuracy. The reason that the parallel system detection performance is slightly better in some cases is due to the improved contour analysis algorithms it uses, which is not available for use in Java. In addition, development of the expert system for the parallel system would allow the tuning parameter, (discussed earlier), to be increased allowing for the detection of more defects, but without the penalty of more false-positive defects.

Due to differences of the contour data structures used by the parallelized and sequential algorithms that identify severe defects, it is better to rewrite the expert system that determines whether an indentified area is really an external log defect. Upon this we may implement a parallelized version of the expert system. A parallel algorithm equally partitioning the identified regions among processors could be developed, each running the same system during the entire process for a significant portion of the execution time. This would greatly reduce processing time for this step. However, the key is to rewrite the rather complicated expert system.

## Acknowledgements

The work upon which this publication is based was funded in whole or in part through a grant awarded by the Wood Education and Resource Center, Northeastern Area State and Private Forestry, Forest Service, U.S. Department of Agriculture.

## References

Bhandarkar, S.; Faust, T.; Tang, M. 1999. Catalog: a system for detection and rendering of internal log defects using computer tomography. Machine Vision and Applications 11(4): 171–190.

Chang, S. 1992. External and internal defect detection to optimize cutting of hardwood logs and lumber. Technical Report 3, USDA and National Agricultural Library, Beltsville, MD. ISSN 1064-3451. 24 p. Available: http://www.nal.usda.gov/ttic/industry/ hrdwood3.htm.

Devaru, D.; Halabe, U. B.; Gopalakrishnan, B.; Agrawal, S.; Grushecky, S. 2005. Algorithm for detecting defects in wooden logs using ground penetrating radar. Proc. SPIE Vol. 5999, 59990B.

William Gropp. 2002. *MPICH2: A new start for MPI implementations*, In Recent Advances in Parallel Virtual Machine and Message Passing Interface, Dieter Kranzlmüller, Peter Kacsuk, Jack Dongarra, and Jens Volkert.

Guddanti, S.; Chang, S. 1998. Replicating sawmill sawing with topsaw using ct images of a full length hardwood log. Forest Products Journal 48(1): 72–75.

Kline, D.; Widoyoko, A.; Wiedenbeck, J.; Araman, P. 1998. Performance of color camera machine vision in automated furniture rough mill systems. Forest Products Journal 48(3): 38–45.

Bradski, Gary. 2000. The OpenCV library. Dr. Dobbs Journal of Software Tools. http://www.drdobbs.com/open-source/the-opencv-library/184404319. Accessed Feb. 12, 2013.

Li, P.; Abbott, A.; Schmoldt, D. 1996. Automated analysis of ct images for the inspection of hardwood log. In Proceedings of the International Conference on Neural Networks, Washington, D.C., USA. 3: 1744–1749.

Nordmark, U.; Oja, J. 2004. Prediction of board values in Pinus sylvestris sawlogs using X-ray scanning and optical three-dimensional scanning of stems. Scandinavian Journal of Forest Research, 19(5):473-480(8).

Pacheco, Peter S. 1997. Parallel programming with MPI. Morgan Kaufmann Publishers, Inc. San Francisco, CA. 418p.

Thomas, L.; Mili, L. July, 2007. A Robust GM-Estimator for the Automated Detection of External Defects on Barked Hardwood Logs and Stems. IEEE Transactions on Signal Processing 55(7): 3568–3576.

Thomas, L.; Mili, L.; Thomas, E.; Shaffer, C. October, 2006. Defect Detection on Hardwood Logs Using Laser Scanning. Wood Fiber Science Journal 38(4): 682–695.

Thomas, L.; Shaffer, C.; Lamine Mili, L.; Thomas, E. April, 2007. Automated Detection of Severe Surface Defects on Barked Hardwood Logs. Forest Product Journal 57(4): 50–56.
Tian, X.; Murphy, G. 1997a. Automated feature extraction and defect recognition from digital images of tree stems using texture analysis. In Proceedings of the first joint Australia & New Zealand biennial conference on Digital Image & Vision Computing –Techniques and Applications.
Tian, X.; Murphy G. 1997b. Detection of trimmed and occluded branches on harvested tree stems using texture analysis. International Journal of Forest Engineering 8(2): 65–78.
Wagner, F.; Taylor, F.; Ladd, D.; McMillin, C.; Roder, F. 1989. Ultrafast ct scanning of an oak log for internal defects. Forest Products Journal 39(11/12): 62–64.
Zhu, D.; Beex, A. 1994. Robust spatial autoregressive modeling for hardwood log inspection. Journal of Visual Communication and Image Representation 5(1): 41–51.
Zhu, D.; Conners, R.; Schmoldt, D.; Araman, P. 1996. A prototype vision system for analyzing ct imagery of hardwood logs. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics 26(4): 522–532.

## Abstract

The 18th International Nondestructive Testing and Evaluation of Wood Symposium was hosted by the USDA Forest Service's Forest Products Laboratory (FPL) in Madison, Wisconsin, on September 24–27, 2013. This Symposium was a forum for those involved in nondestructive testing and evaluation (NDT/NDE) of wood and brought together many NDT/NDE users, suppliers, international researchers, representatives from various government agencies, and other groups to share research results, products, and technology for evaluating a wide range of wood products, including standing trees, logs, lumber, and wood structures. Networking among participants encouraged international collaborative efforts and fostered the implementation of NDT/NDE technologies around the world. The technical content of the 18th Symposium is captured in this proceedings.

Keywords: International Nondestructive Testing and Evaluation of Wood Symposium, nondestructive testing, nondestructive evaluation, wood, wood products

September 2013

Ross, Robert J.; Wang, Xiping, eds. 2013. Proceedings: 18th International Nondestructive Testing and Evaluation of Wood Symposium. General Technical Report FPL-GTR-226. Madison, WI: U.S. Department of Agriculture, Forest Service, Forest Products Laboratory. 808 p.

A limited number of free copies of this publication are available to the public from the Forest Products Laboratory, One Gifford Pinchot Drive, Madison, WI 53726–2398. This publication is also available online at www.fpl.fs.fed.us. Laboratory publications are sent to hundreds of libraries in the United States and elsewhere.

The Forest Products Laboratory is maintained in cooperation with the University of Wisconsin.



## Contents