Y/AMT-301

Y-12

OAK RIDGE Y-12 PLANT

LOCKHEED MARTIN

CRADA Final Report for CRADA Number ORNL92-0140

SEMICONDUCTOR YIELD IMPROVEMENT THROUGH AUTOMATIC DEFECT CLASSIFICATION

S. S. Gleason
M. A. Hunt
H. Sari-Sarraf
Lockheed Martin Energy Systems, Inc.
A. Kulkarni
KLA Instruments Corporation

September 1995

PROTECTED CRADA INFORMATION

This product contains protected CRADA Information which was produced on February 2, 1995 under CRADA No. ORNL92-0140 and is not to be further disclosed for a period of five (5) years from the data it was produced except as expressly provided for in the CRADA.

Prepared by the
Oak Ridge Y-12 Plant
Oak Ridge, Tennessee 37831
managed by
Lockheed Martin Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

MANAGED BY
LOCKHEED MARTIN ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

UCN-13672 (26 6-95)

96

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

ORNL

CRADA # ORNL92-0140 FINAL REPORT

SEMICONDUCTOR YIELD IMPROVEMENT THROUGH AUTOMATIC DEFECT CLASSIFICATION

S. S. Gleason
M. A. Hunt
H. Sari-Sarraf

September 1995

Prepared for the
ORNL/KLA Automatic Defect Classification CRADA
under CRADA No. ORNL92-0140
between the U.S. Department of Energy, Martin Marietta Energy Systems and
KLA Instruments Corporation

Prepared by
Instrumentation and Controls Division
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6011
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

Table of Contents

Semiconductor Yield Improvements Through Automatic Defect Classification

Abstract

Automatic detection of defects during the fabrication of semiconductor wafers is largely automated, but the classification of those defects is still performed manually by technicians. Projections by semiconductor manufacturers predict that with larger wafer sizes and smaller line width technology the number of defects to be manually classified will increase exponentially. This cooperative research and development agreement (CRADA) between Oak Ridge National Laboratory and KLA Instruments developed concepts, algorithms and systems to automate the classification of wafer defects to decrease inspection time, improve the reliability of defect classification, and hence increase process throughput and yield. Image analysis, feature extraction, pattern recognition and classification schemes were developed that are now being used as research tools for future products and are being integrated into the KLA line of wafer inspection hardware. An automatic defect classification software research tool was developed and delivered to the CRADA partner to facilitate continuation of this research beyond the end of the partnership.

1. CRADA Objectives and Accomplishments

Objective 1: Sample Set Generation and Characterization - (KLA) Generate sets of characterized image pairs with available database information using the KLA 2100 Series Wafer Defect Inspector. The image pairs will contain a reasonable distribution of circuit cell backgrounds, normal process variations, flaw types and categories for use by the MMES Staff in developing and testing inspection approaches.

Accomplishments: KLA had four milestones to deliver characterized sets of image pairs captured using the KLA 2100 Series Wafer Defect Inspector. Because of some initial difficulties in gaining access to a KLA 2100 inspection system, ORNL staff decided to proceed by first analyzing a set of feature vectors sent by KLA engineers. These feature vectors consisted of 47 defect features extracted from low-resolution grayscale defect images. This data was unlabeled, therefore unsupervised learning techniques were employed by ORNL staff to analyze the data.

All of the four sets of images were subsequently delivered by KLA to ORNL staff over the course of the two-and-a-half year project. Two of the data sets had a variety of classified defects that could be used to test feature discriminatory power using some of the ORNL supervised classification schemes. Details of the data sets are provided in the Technical Accomplishments section of this final CRADA report.

Objective 2: Classification Language Development - (KLA/MMES) Interact with the major manufacturers of semiconductor products and key players in the DOE DO microelectronic R&D program to identify relevant image descriptors and a classification "language" for use with the automated classifier.

Accomplishments: Information was gathered from several manufacturing companies (HP, TI, DEC,...) concerning their particular classification terminologies, and a common defect language was adopted for this project based on the individual classification schemes from each company. It is typical for different manufacturers to have similar defect types but have a completely different naming scheme for the defect as well as a different technique for identifying it as belonging to a certain class of defects. This classification language development allowed ORNL and KLA staff to adopt the same defect language terminology during this project, but a flexible system which allows any user-defined defect class naming convention was implemented in the final ORNL software tool deliverable.

Objective 3: Feature Vector Development - (MMES) Develop novel techniques for image segmentation and feature extraction in the presence of semi-regular background texture, and develop feature vectors that best differentiate between flaw types and classes. Defects can occur on both a local and a global scale. Each case will be handled separately.

Accomplishments: ORNL staff was successful in developing novel feature extraction techniques that would allow unique feature vector descriptions to be created for specific defect classes. Feature extraction techniques based on fractal dimension, multiple-focus imagery, color information, and multiple-defect mask implementations are just a few examples of the ORNL accomplishments in this area.

Because of a KLA in-house effort to perform global analysis of defect distributions, it was mutually decided by the PI at ORNL and the PI at KLA that ORNL staff would focus on the analysis of high-resolution color images of defects to make the most efficient use of resources at both facilities.

Objective 4: Classifier Development - (MMES) Develop high-speed adaptable classifiers based on image feature vectors. Merge classification language, description data-base, and image features into a coherent adaptive classification technique.

Accomplishments: Multiple classifier configurations were researched by ORNL staff in a search for the classifier (or group of classifiers) that performed well on semiconductor defect feature data. Classical classification techniques (both supervised and unsupervised)

such as nearest prototype, k-means, and Fisher pairwise were researched along with more recently developed techniques such as neural networks (backpropagation, radial basis function networks) and Kohonen self-organizing feature maps. From these large groups of classifiers, one classical approach (Fisher pairwise) and one recently developed approach (radial basis function network) stood out from the rest as being applicable to the semiconductor defect data sets that ORNL staff had access to via KLA or were able to create on their own. The radial basis function network is an adaptable classification approach which can be trained based on an individual manufacturer's defect distribution. Both classifiers relied upon the classification language as an input defect naming convention, but both are flexible enough to allow user-defined defect naming conventions.

A context-based classification system was also developed that classifies defects based on a combination of their location on the semiconductor wafer and the defect characteristics. The use of this contextual information in conjunction with defect attributes emulates the approach taken by some defect classification experts. The ORNL-developed classification scheme using region classification based on edge information and color content was one of the major achievements of the partnership.

Objective 5: Product Line Modification Identification - (KLA) Identify the hardware, software, and operating procedure modifications required of KLA Instruments three primary product lines in wafer inspection to accommodate automated defect classification. Perform technical trade-off studies, market assessments, and cost-benefit analysis on the implementations which hold the most promise for benefiting the semiconductor wafer manufacturing industry.

Accomplishments: ~~**Begin Protected CRADA Information**~~[Through KLA's intense marketing analysis effort of their line of inspection systems, they determined the most successful way to modify their current product line to include high-resolution defect classification. Their approach is to eventually allow the end user to incorporate the ADC system into the on-line inspection process, while most competitors are still targeting only off-line ADC solutions. KLA will modify their existing line of inspection workstations to include their own global defect distribution classification scheme mentioned previously. The result of the global defect pattern classification will be statistically sampled to get a good representation of defects. These defects will then be imaged by a high-resolution (0.5 to 1.0 micron/pixel) color camera and subsequently classified using developed ADC technology.]~~**End Protected CRADA Information**~~

Objective 6: Demonstration Prototype Development - (MMES) Assimilate developed approach (optimum feature vectors, classification scheme, and classification language) into a demonstration prototype. Complete system testing, verification, performance characterization, system demonstration, and final report.

Accomplishments: ORNL staff worked with KLA staff to build a comprehensive research ADC tool prototype that will allow both parties to explore the field of feature extraction, data visualization and pattern classification. This software ADC prototype system has sophisticated graphical user interface that allows input of multiple defect and reference images, user-definable feature extraction methods, and user-selectable pattern recognition schemes based on a variety of classifiers. The software tool is built using C++ and Motif libraries. The system was demonstrated and delivered to KLA as the final deliverable milestone for the project. KLA has already used portions of the ORNL software in their ADC beta test prototype system installed at a customer site, and plan to use the ADC tool to help them determine optimal feature measurements and effective classifiers for the variety of semiconductor types that they will address in the future. A user's guide and technical code description were prepared and are included in Appendices H and I.

A summary of all milestones and their respective dates of completion are shown below:

MILESTONES

MMES

Date	Status	Description
08/03/93	Complete	ORNL visit to KLA
09/31/93	Complete	ORNL visit to KLA, capabilities and image analysis update
02/28//94	Complete	Feature extraction progress report
07/31/94	Complete	Classifier design progress report
10/31/94	Complete	Feature extraction demonstration
10/31/94	Complete	Feature extraction final report
10/31/94	Complete	Classifier demonstration
10/31/94	Complete	Classifier final report
02/28/95	Complete	Prototype design complete
05/31/95	Complete	Prototype intermediate test/demonstration
08/24/95	Complete	Deliver final ADC report, final prototype demonstration

KLA

Date	Status	Description

07/31/93	Complete	Deliver image/classification set #1
09/15/93	Complete	KLA/ORNL meeting with defect classification expert
10/31/93	Complete	Classifier language document
01/31/94	Complete	Deliver image/classification set #2
06/30/94	Complete	Deliver image/classification set #3
08/31/94	Complete	Product line modification identification progress update
12/20/94	Complete	Deliver image/classification set #4
06/31/95	Complete	Final product line modification identification

2. Benefit to DOE DP Microelectronics Mission

The DOE relies heavily on the availability of high quality, thoroughly tested integrated circuits for practically all Defense Programs ranging from nuclear weapons production, testing, and detonation, to safeguards and security, and advanced manufacturing feedback and control systems. Most of these circuits are purchased commercially and thus the benefit of providing high speed silicon wafer inspection systems to the manufacturers of semiconductor products is inherent.

DOE also develops and produces custom integrated circuits for its programs. The Microelectronics Development Laboratory (MDL) at Sandia National Laboratory is responsible for semiconductor manufacturing process development and specific integrated circuit design for many DOE defense and intelligence needs. They have primary responsibility for the Development of high temperature tolerant circuits and radiation hardened circuits and processes. Although the state of the art for these technology areas lag those of the commercial sector in wafer area and line widths, they have already exceeded circuit complexity for which human inspection is viable.

While MDL could benefit directly from the use of an enhanced wafer inspection system, they are basically a low volume, prototype development and demonstration facility. The main economic advantage to DOE-DP with respect to custom integrated circuits lies in receiving high yield manufacturing runs from commercial vendors who mass produce MDL perfected circuits (such as the United Technologies Inc. MicroelectronicsCenter, Colorado Springs, Colorado). Another DP program which could benefit directly from this project is the joint SEMATECH/DOE Center for Contamination Free Manufacturing also at Sandia National Laboratory. In each of these cases control of the fabrication line to minimize defects and maintain acceptable yields of high quality product is not possible without on-line wafer inspection and defect classification. This project will allow Wafer Inspection Systems to meet current and future demand for yield optimization by providing rapid analysis of the semiconductor manufacturing process.

2. Benefits to U.S. Economy

The semiconductor manufacturing industry in the U.S. is approximately $50 B/yr.

Defect detection and classification plays an important part in maintaining and improving production yield in semiconductor fabrication lines. A wafer must be processed to 80 or 90% completion before it can be tested electrically. Optical methods and Scanning Electron Microscopes (SEMs) are the only methods available for inspection during most of the processing.

Because "killer" defects are detected on sample wafers after a lot (usually 25) of wafers has been processed, prompt detection and classification of any defects is essential to correcting process problems and maintaining the yield of the fabrication line. With smaller and smaller design rules (feature size, line width, etc.) the allowable particle and defect sizes are correspondingly reduced. In other words, the range of particle and defect sizes is increased leading to a drastic increase in the number to detect and classify. To this is added the larger wafer size, so the area to be inspected is almost 80% greater with the new 200-mm diameter wafers than with the older 150-mm wafers. The trend towards a 300-mm diameter wafer standard will again more than double the area to be inspected, hence double the number of defects to detect and classify.

To put this into a national perspective, the cost of Motorola's newest 200-mm fabrication line in Austin was approximately $500M. The value of product wafers in a typical process line at any one time is in the range of $50M to $100M. (It normally takes about 6 weeks to completely process a lot of wafers.) It should be clear that increasing inspection and analysis time to accommodate the larger wafers and increased number of defect classes cannot be easily tolerated. Both the increased speed of wafer scanning and automation of defect classification is essential for the industry to maintain the volume of production with adequate yield to pay for the rapidly escalating cost of the production facilities and equipment. Improved defect inspection is essential for the long term profitability of U.S. semiconductor industry. A 0.1% yield improvement in wafer production is worth $100M annually to the U.S. semiconductor industry.

3. Technical Accomplishments

Appendix A: Defect Classification Dictionary, January, 1994.

Appendix B: Automatic Semiconductor Defect Classification Technical Update, May, 1994 (~~Protected CRADA Information~~).

Appendix C. Radial Basis Function Routine Descriptions, June, 1994.

Appendix D: Semiconductor Yield Improvement Technical Progress Report, August, 1994 (~~Protected CRADA Information~~).

Appendix E: ORNL's Off-Focus Image Feature Extraction, October, 1994 (~~Protected~~ ~~CRADA Information~~).

Appendix F: Semiconductor Yield Improvement Technical Progress Report, November, 1994 (~~Protected CRADA Information~~).

Appendix G: ADCtool User's Guide, August, 1995 (~~Protected CRADA Information~~).

Appendix H: ADCtool Code Technical Documentation, August, 1995 (~~Protected~~ ~~CRADA Information~~).

4. Invention Disclosures

A Report of Possible Invention form has been submitted entitled "Contextual-based System for Automatic Defect Classification Using Multiple Morphological Masks." ORNL staff developed novel image analysis techniques that allow unique feature vector descriptions to be created for specific defect classes. Feature extraction techniques based on multiple-defect mask images and context-based image segmentation are examples of the ORNL unique contributions in this area. The subsequent combination of defect features with context information to classify defects based on defect features as well as defect location is also a unique approach to the automatic defect classification problem.

5. Commercialization Potential

Correlation of chip failures at the end of the manufacturing process with visual defects occurring during the manufacturing process is critical to improving semiconductor manufacturing yield. To understand how the defects impact yield, semiconductor manufacturers review and classify the defects detected into 10-15 categories per process level, then subsequently correlate those with the electrical failures. Engineering effort is then applied

to identify and eliminate the cause of the defects. This activity results in significant yield improvement which accelerates a new product to the market and improves the profitability of an established line.

Modern automated wafer inspection systems can detect 50,000 to 100,000 defects per day. To detect an excursion of yield impacting defects on which to focus engineering efforts, semiconductor manufacturers must classify a statistically significant portion of all visual defects detected. In doing so, they are faced with the following challenges:

- With state-of-the-art automated review microscopes, an operator can classify 4 defects per minute, so an entire workday is needed to classify a 2% sample of defects found.

- Manufacturing costs cause manufacturers to classify only a very small sample portion of defects detected, thereby discarding valuable information and providing statistically unsound data bases.

- Operator classification is often inconsistent.

- The presence of operators in the wafer fabrication line and their handling of wafers represents a significant source of contamination.

In the last 8 years there has been over 2000X improvement in defect detection speed, and 10X improvement in the defect sizes detected by automated wafer inspection systems. Although substantial improvements have been made in the ability of automated wafer defect detection, no significant progress has been made towards automating defect classification or towards improving defect classification speed and accuracy. As a result classification represents a significant bottleneck in yield improvement. Defect classification is one of the most labor intensive activity remaining in semiconductor manufacturing.

The path to commercialization of the ADC technology has already been established by KLA. Work that ORNL performed with regard to the radial basis function network classifier led to the integration of this classification scheme into KLA'a existing ADC prototype located at a customer site. The ADC tool provided by ORNL will be a valuable research test bed that KLA can use to evaluate the applicability of various combinations of features and classifiers.

6. Future Collaboration

KLA Instruments Corporation was interested in continuing a working relationship with ORNL in two specific areas. The first is non-reference based techniques for defect detection and the second addresses fuzzy inference classification techniques. The ORNL PI provided them with information about Work for Others contracts as well as 100% Funds-In CRADAS. KLA's current position, however, is that they would prefer to hire individuals as permanent members of their research staff. KLA is currently in such a state of rapid growth that they feel they can hire full time engineers to assist them with this type of development activity. They have stated that they are still interested in future collaborations with ORNL if an opportunity arises.

7. Conclusions

The partnership between ORNL and KLA Instruments Corporation was much more successful than it would have been in either participant had attempted such research on their own. Participation in this endeavor certainly benefited both parties. ORNL was able to focus on areas of ADC that KLA did not have the resources and expertise to devote to. Specifically, progress was made on the use of color, and multiple focus images as sources for feature extraction as well as evaluation of multiple classifier types. Researching these areas allowed ORNL staff to gain valuable expertise in the field while providing KLA with a valuable research service. Also, both parties will benefit beyond the expiration of this CRADA through use of the ADC software research tool built by ORNL staff.

KLA provided invaluable help in the area of defect classification expertise from the viewpoint of a human operator. Also, they introduced ORNL staff to the semiconductor manufacturing process so that they might better understand the source of defects. The key to the success of the entire project was the four sets of images that KLA staff painstakingly acquired and delivered to ORNL staff for analysis. Finally, an analysis of their product line led them to a decision about how to integrate the ADC techniques into their current and future product lines.

Appendix A

Defect Classification Dictionary

January 1994

DEFECT CLASSIFICATION DICTIONARY

KLA/ORNL ADC CRADA Team
January, 1994

1.0 INTRODUCTION

This report is being written to support the semiconductor automatic defect classification (ADC) work as part of the Cooperative Research and Development Agreement (CRADA) between KLA Instruments, Corp. and Oak Ridge National Laboratory (ORNL). Semiconductor manufacturers typically have their own in-house defect classification scheme that meets the need of their particular wafers and manufacturing processes. Several defect classification schemes have been collected from seven of the most prominent semiconductor wafer manufacturers: Texas Instruments (TI), Digital Equipment Corporation (DEC), Intel, National Semiconductor, Hewlett-Packard (HP), Cypress, and Advanced Micro Devices (AMD).

Extraction of discriminatory features from defect images is an essential step in the ADC process. The selection of appropriate defect characteristics will directly determine the level of success of the selected ADC scheme. This document is intended to facilitate the feature extraction process by providing a defect dictionary which attempts to consolidate the different defect types and their physical descriptions. Defect types are listed along with physical descriptions that will provide insight into the process of automatic feature extraction for the purpose of ADC.

Another purpose for this dictionary is to rank, when applicable, the defect types in order of importance for ADC. It would not be practical to try to compile one list of defects in a particular order of importance that represents the view of all semiconductor manufacturers. It is also not practical to try to immediate solve the ADC problem by finding a scheme which successfully classifies all defects one-hundred percent of the time. The goal of this CRADA, after all, is to develop an ADC product that can be delivered to KLA customers as soon as possible to provide the majority of manufacturers with a significant level of ADC capability above their current level. A better approach is to divide and conquer the problem by focusing efforts on the most significant defect categories that commonly plague the manufacturing processes for the majority of manufacturers. Based on KLA's experience with the needs of their large customer base, this ranking presents a plan of attack to develop an ADC to meet the previously described goals.

This is intended to be a working document, and defect types, descriptions and rankings may be adjusted during the course of the CRADA. The main goal of this ADC effort is, of course, to improve yield by identifying root causes of defects. The first step in finding the source of a problem is to identify the defect itself. The defect types and corresponding features listed in this report are potential discriminatory features selected from the current images and classification hierarchies. They may change wih time as more important characteristics and/or defect types update this current list.

2.0 DEFECT DESCRIPTIONS

The defects are listed in tabular format along with information such as defect description, location, color, shape and size. This information is given in Table A1. As other defect classes and descriptions are deemed important enough to include in this classification hierarchy, they can be added to Table A1.

Table A1: Defect descriptions

Defect Name	Description/Cause	Location	Color	Shape	Size
1 Particle Defects	Foreign pieces of material contaminating the wafer.	anywhere	see below	see below	see below
1.1 Surface Particle	A foreign material which has contaminated the surface of the wafer.	anywhere	see below	see below	see below
1.1.1 Large Particle	same as above	anywhere	opaque	irregular, 3D	>> 1 micron
1.1.2 Medium Particle	same as above	anywhere	opaque	elliptical, 3D	> 1 micron
1.1.3 Small Particle	same as above	anywhere	opaque	elliptical, 3D	< 1 micron
1.2 Deposition Particle	A foreign material which has contaminated the wafer and has since been covered by another layer.	anywhere	see below	see below	see below
1.2.1 Present	A deposition particle still present on previous layer.	anywhere	current layer	see below	see below
1.2.1.1 Large Particle	same as above	anywhere	current layer	irregular, 3D	>> 1 micron
1.2.1.2 Medium Particle	same as above	anywhere	current layer	elliptical, 3D	> 1 micron
1.2.1.3 Small Particle	same as above	anywhere	current layer	elliptical, 3D	< 1 micron

Table A1: Defect descriptions

Defect Name	Description/Cause	Location	Color	Shape	Size
1.2.2 Absent	A deposition particle that was present on previous layer, but has since washed off or broken away.	anywhere	previous/ current layer	see below	see below
1.2.2.1 Large Particle	same as above	anywhere	previous/ current layer	irregular, 3D	>> 1 micron
1.2.2.2 Medium Particle	same as above	anywhere	previous/ current layer	elliptical, 3D	> 1 micron
1.2.2.3 Small Particle	same as above	anywhere	previous/ current layer	elliptical, 3D	< 1 micron
2 Process Defects	Defects caused by a malfunction in the manufacturing process.	see below	see below	see below	see below
2.1 Pattern	An extension or deletion of the wafer geometry.	adjacent to pattern area	see below	see below	see below
2.1.1 Extra Pattern	An extension of the wafer geometry caused during the lithography step. Typically caused by a defect or contaminant on the litho mask.	adjacent to pattern area	same as pattern	irregular, but adjacent to correct pattern, 2D	typically > 1 micron
2.1.2 Missing Pattern	An omission of the wafer geometry caused during the lithography step. Typically caused by a defect or contaminant on the litho mask.	within pattern area	background	irregular, but within correct pattern, 2D	typically > 1 micron

Table A1: Defect descriptions

Defect Name	Description/Cause	Location	Color	Shape	Size
2.2 Etch	A problem in the process of etching away metal or resist from predefined areas on the wafer.	anywhere	see below	irregular, 2D	various
2.2.1 Over Etch	Too much material (resist or metal) has been etched away causing broken leads or potential shorts.	anywhere	previous layer	irregular, 2D	various
2.2.2 Under Etch	An area on the wafer that has not been etched enough usually caused by something (particle or solvent splash) blocking the etching process.	anywhere	current layer	irregular, 2D	various
2.3 Flake	A piece of the wafer that has broken loose from one area and landed in another.	anywhere	highly varied, colorful	irregular, 2D	typically >> 1 micron
2.3.1 Resist Flake	A piece of resist which has come loose from the substrate and landed elsewhere on the wafer.	anywhere	highly varied, colorful	irregular, 2D	typically >> 1 micron
2.3.2 Metal Flake	A piece of metal which has come loose from the substrate and landed elsewhere on the wafer.	anywhere	highly varied, colorful	irregular, 2D	typically >> 1 micron
2.4 Scratch	Large removal of material from one or more layers.	anywhere	consistent, depends on layers affected	elongated, surrounded by particles, 3D	> 1micron

Appendix B

Automatic Semiconductor Defect Classification Technical Update

May 1994

~~(Protected CRADA Information)~~

Automatic Semiconductor Defect Classification Technical Update

May 2, 1994

Prepared by
Shaun Gleason, Martin Hunt, and Hamed Sari-Sarraf

For
KLA Instruments Corporation

As part of the
Cooperative Research and Development Agreement #ORNL92-0140
Between the Department of Energy and KLA Instruments Corporation

1.0 Introduction

This is a technical report written by the ORNL staff for KLA Instruments Corporation as part of the Automatic Defect Classification (ADC) Cooperative Research and Development Agreement (CRADA). This report describes all of the work done on the Digital Equipment Corporation (DEC) wafer defect images sent to ORNL by KLA. Unix scripts were written to organize the image data, and Khoros workspaces were developed to perform defect segmentation and feature extraction. This set of images has multiple focus depths, and the feature extraction techniques chosen are intended to capitalize on potential characteristics of off-focus defect information that may differentiate between particle and process type defects. Feature vector files in the On-Line Pattern Analysis and Recognition (OLPARS) software format were generated and made available to ORNL as well as KLA staff for analysis. Another parallel effort in defect classification that attempts to automatically describe the context of the defect in relation to its background is being pursued. This model-based approach and its integration with the current defect analysis work are described.

2.0 Automatic Defect Classification System

In this section, we propose and describe the overall architecture of the Automatic Defect Classification System (ADCS). The proposed system is depicted in block diagram form, and is shown in Figure 1. The main characteristic of this approach to defect classification is that the identity of the layer(s) whose domain is interrupted by the defect is taken into account, and is included among the features extracted from the defect itself. These particular measures are essential in ensuring the correct classification of some types of particle defects such as poly, field, and active area particles. Another characteristic of the ADCS is that it is modular, thus, an algorithm redesign or update in one module can be carried out without altering the overall architecture of the system. In what follows we describe the nature and the flow direction of the data from each of the modules in the ADCS.

As stated above, a crucial task in the ADCS is to determine the location of a defect, *i.e.*, the layer(s) on which it falls. To accomplish this, we propose the implementation of a model-based, 2-D object recognition subsystem as illustrated at the top of Figure 1. To gain an understanding of this approach, let us consider the reference, as well as the defect image shown in Figure 2. What is apparent from these images, and typical of most high-resolution images of dies, is that the background (all that is not defect) is composed of a finite number of geometric (curvilinear and rectilinear) 2-D objects. The images plus shape and color related features of these objects (see Figure 3 (a) for an example) are the basis for the proposed model-based recognition.

The aforementioned symbols, and features thereof, are kept in a database which is queried by an operator to generate a subset of candidate symbols for a given lot of wafers. The object detection module performs a coarse segmentation by finding the closest match of each of the candidate symbols in the reference image. Instead of considering the entire field of view, the output from the registration & subtraction module, *i.e.*, the defect location within the field of view, can be utilized to limit the search domain and to speed up the detection process. An example of a typical output from the object detection module is shown in Figure 3 (c). This probability matrix was generated by running a wavelet-based object detection algorithm that attempted to find the object in Figure 3 (a) in the image in Figure 3 (b). In the next module, features, *e.g.*, lines, vertices, and color, for each of the symbols are compared to those that are extracted from the reference image. For a given symbol, features are computed only within those subregions of the reference image whose counterparts in the segmented image correspond to high probabilities of match. As the final step in the object recognition subsystem a classifier labels the background symbols in the reference image based on the extracted features, and to each, assigns a confidence value. The labeled background, which identifies the different layers within the field of view, is encoded as features, and is utilized in the context-sensitive defect classification subsystem of the ADCS, shown at the bottom of Figure 1. This subsystem performs defect classification based on the obtained set of features as well as those that are extracted from the flaws themselves.

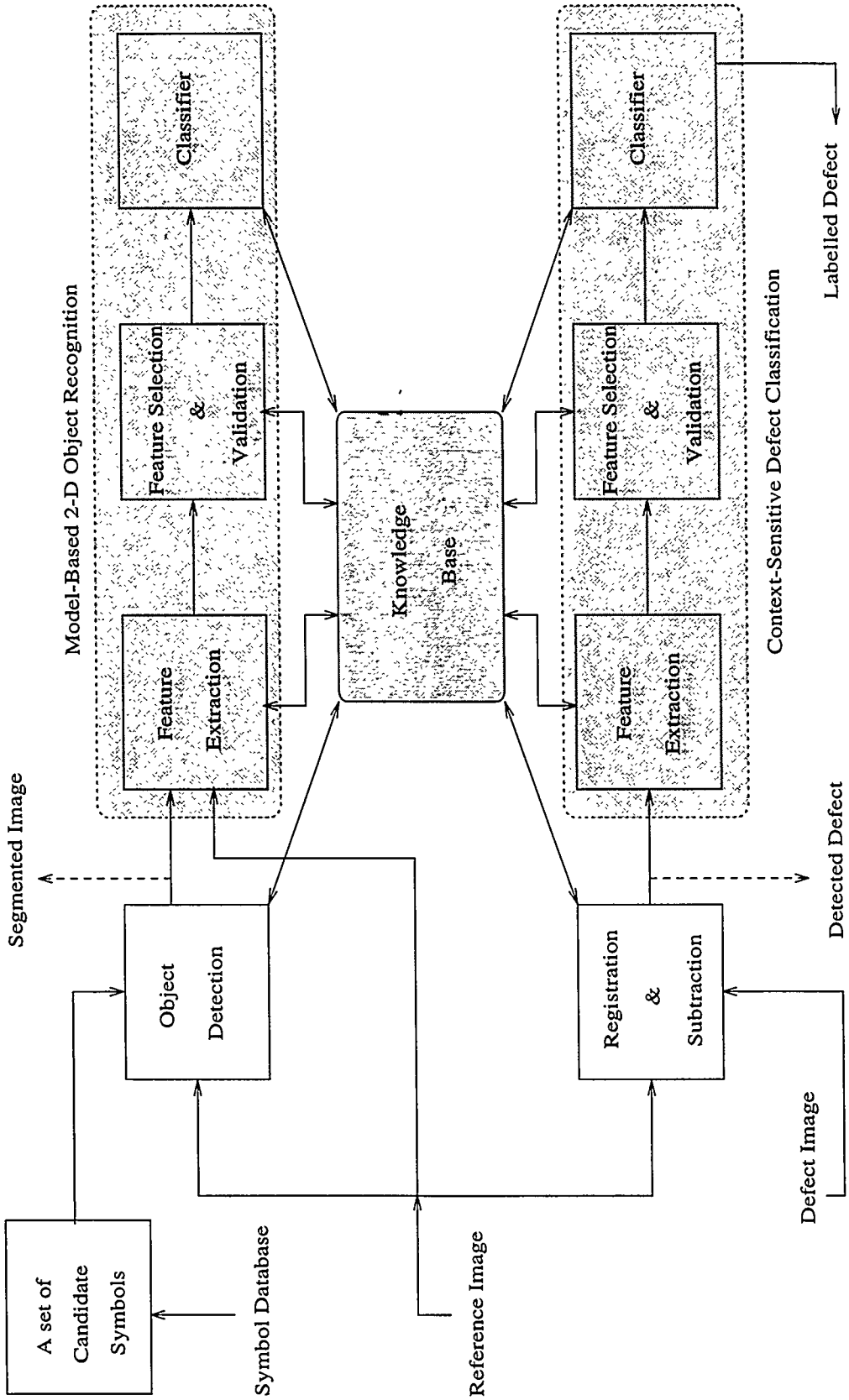
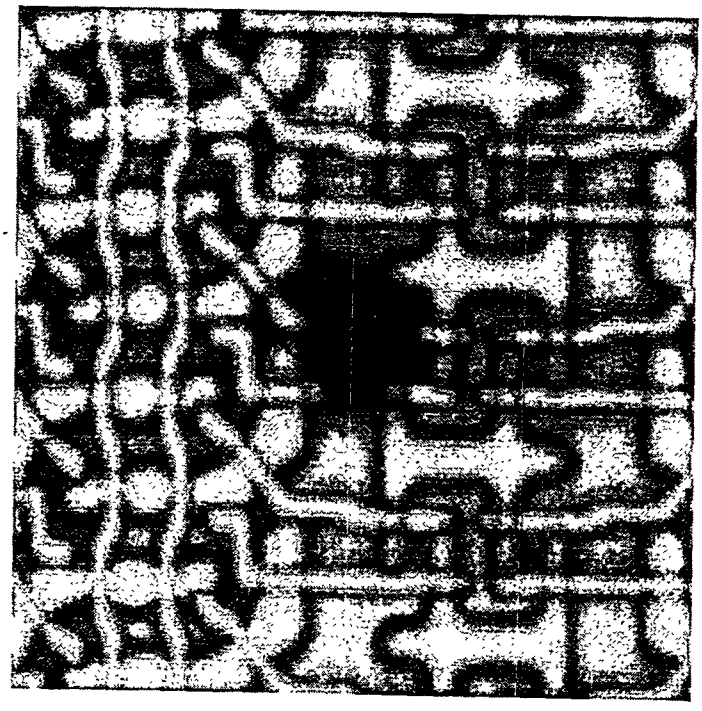


Fig. 1. Automatic Defect Classification System Block Diagram

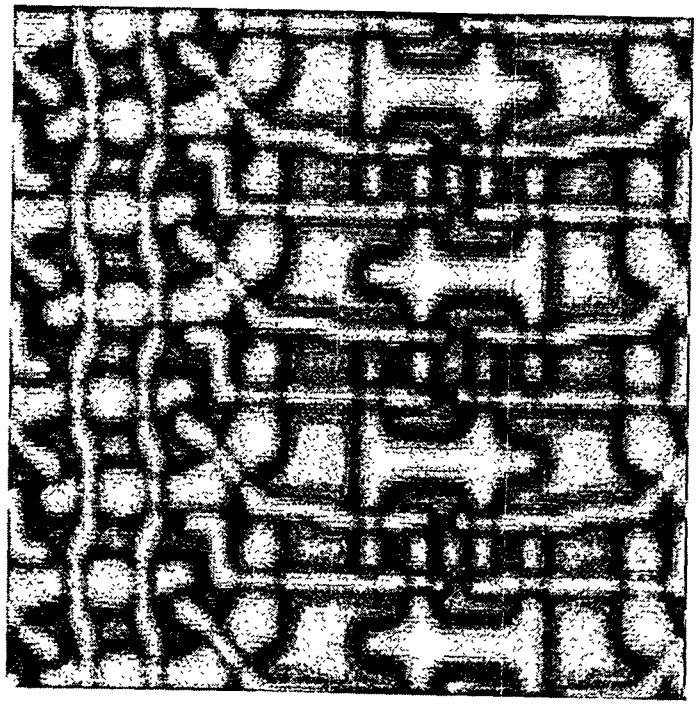
All of the prior knowledge about the problem at hand is encoded into the ADCS in the form of a knowledge database, thus the knowledge base subsystem shown in the center of Figure 1. The function of this subsystem is threefold:

1. It indicates the region(s) of an image where the information of interest is contained, *e.g.*, through the knowledge base, the registration & subtraction module gives the relative location of the defect to the object detection module.
2. It holds a list of defect/background characteristics, as well as the interrelationships among them, *e.g.*, color characteristics of a poly layer, or the fact that a bridging particle tends to create a connection between two layers.
3. It facilitates the feed-forward and feedback interactions between modules, *e.g.*, the classifier module may need, and consequently request through the knowledge base, additional features to be extracted from the considered defect.

We conclude the description of the architecture of the ADCS by making a final observation. The location of those defects that can occur anywhere on the die, such as a scratch, will still be encoded by the ADCS. This information, however, can be used to look at trends of defect location over a given period of time to uncover process related causes.

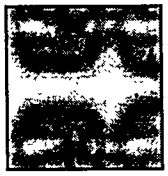


(a)

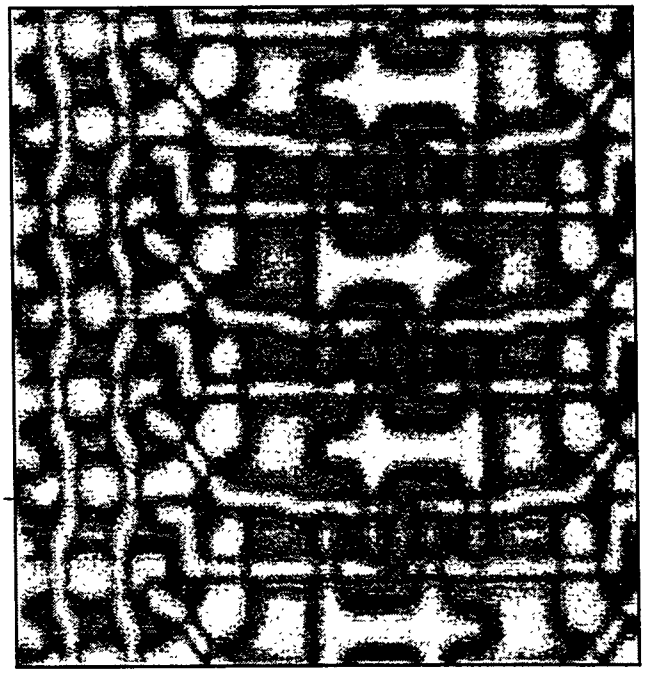


(b)

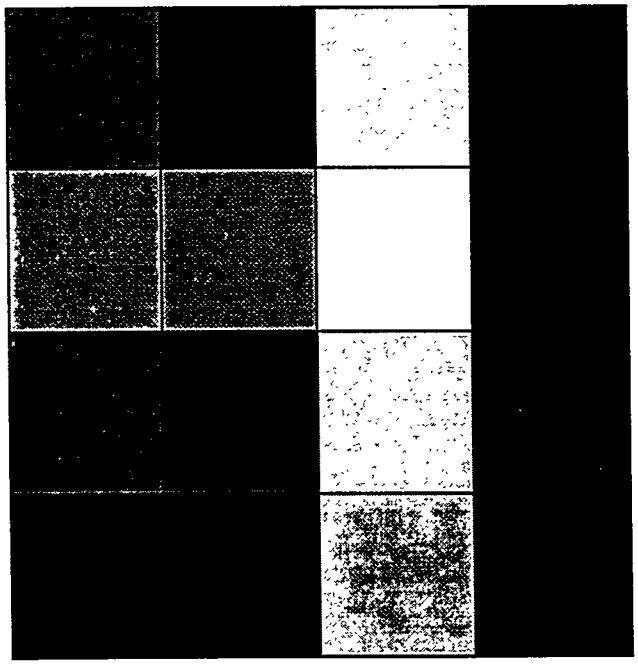
Fig. 2. Example defect image (a) and reference (b)



(a)



(b)



(c)

Fig. 3. Background symbol (a), candidate search image (b), probability matrix (c) (white indicates probability of one, black indicates probability of zero.

3.0 Defect Distribution

3.1 File sorting

A set of 978 high-resolution color TIFF images was received from KLA arranged in single directory with file names in the form of ####.tif, where # is a number. The file name did not correspond to any particular attribute of the image. Within each TIFF file is a ASCII comment header which includes the defect identification number, the coded classification, and the type of image data. A unix script was written to extract the header information, convert the image to gray-scale, and move the resulting TIFF file to designated directory based on the defect classification. The script also generates an ASCII file in each defect class directory that has the comment information for each image in that directory, one image per row. This script was applied to all of the images received from KLA.

3.2 Header extraction

As mentioned above each image contained an ASCII comment field which listed the attributes of the image. A sample of the header is given as: Image Description: "im# 23, def. id# 11 f off.: -632 nm, class#:33 desc:33, comment:m." The information included in the comment is the sequential image number (23), defect identification number (11), focus offset in nm (-632), defect classification code (33), and the type of image (m). There are six types of images that may be associated with each defect: in-focus defect and reference (comment:0, and comment:ref0), positive (above) focus offset defect and reference (comment:p and comment:refp), negative (below) focus defect and reference (comment:m and comment:refm). This comment information will be used in subsequent processing steps to associate file names with defect groupings.

3.3 Defect type statistics

The ASCII files containing the comment information were imported into a spreadsheet and sorted by classification and defect identification. Table 1 shows a portion of the results of this sorting for the large particle classification.

Table 1: Portion of sorted images

Defect id	Image number	focus offset (nm)	classification code	comment
37	70	-631	33	0
37	71	-632	33	p
37	72	-630	33	m
37	73	-630	33	ref 0
37	74	-631	33	ref p

Table 1: Portion of sorted images

Defect id	Image number	focus offset (nm)	classification code	comment
37	75	-630	33	ref m

It was determined that seven of the defect classifications contained reference and defect images for all focus offsets. The total number of unique defects in each classification is listed in Table 2

Table 2: Number of defects per class

Classification	Count
Large particle	35
Poly particle	16
Bridging particle	14
Active area particle	11
Field particle	7
Previous bridging	3
Missing pattern	2

A pareto chart of all defect classes is shown in Figure 4.

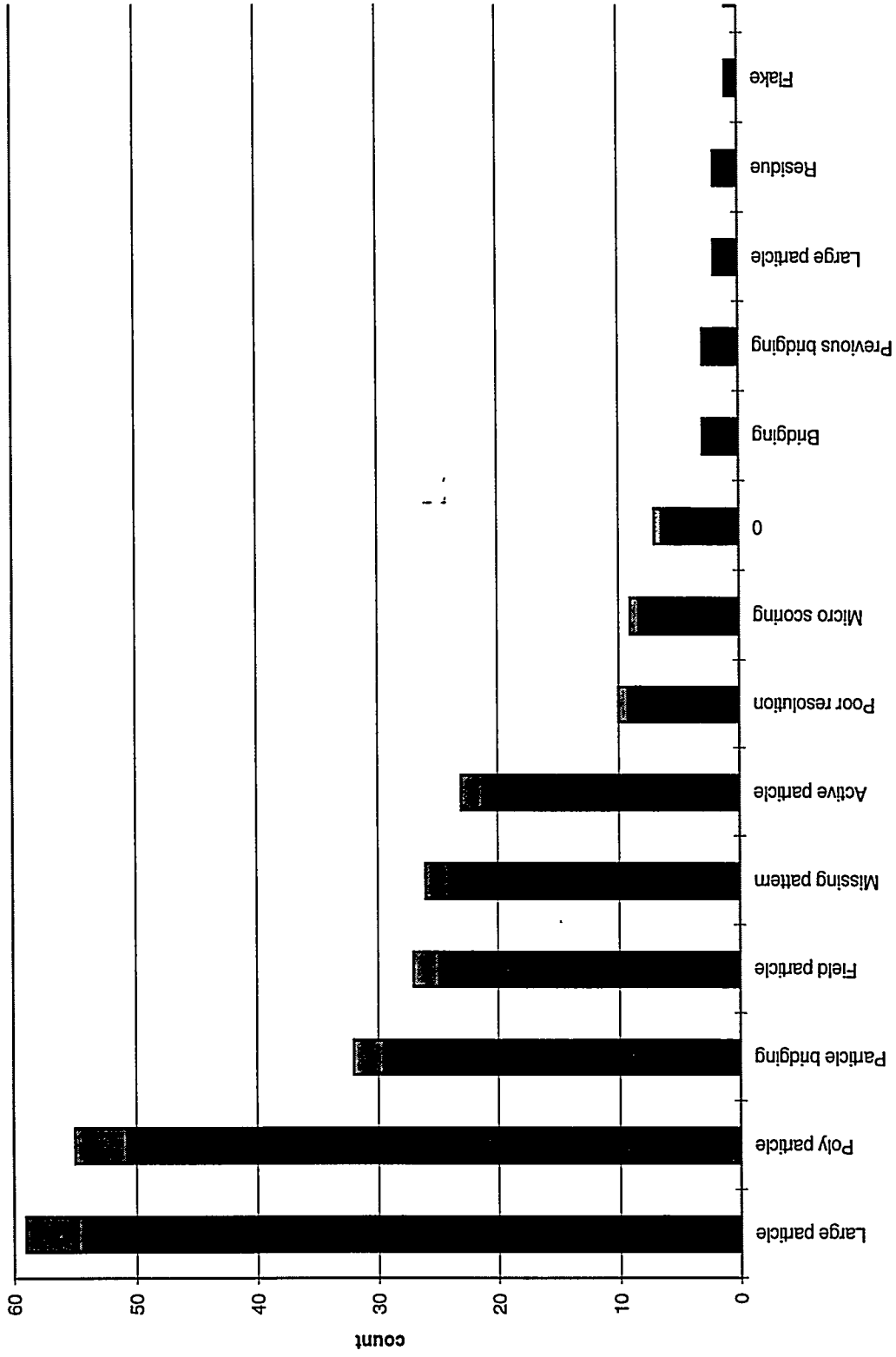


Fig. 4. Pareto chart of defect frequency and percentage

4.0 Defect Segmentation

4.1 Menureg registration of defect images

The development software environment provided by KLA, *menureg,* was initially used to compute the registration parameters for each in-focus image pair and to segment the defect. Several defect images were processed with *menureg,* and it became clear that it would be difficult to use the subpixel translations reported for the in-focus images to register the off-focus images. We were unable to find a direct technique within *menureg* to specify a sub-pixel translation and thus resorted to implementing the sub-pixel translation in the Khoros environment. Another aspect of using *menureg* which was a disadvantage was the manual selection of both the tie point locations and threshold for segmenting the defect. In several cases the threshold had to be raised from a nominal level of 30 to eliminate spurious defects caused by slight misregistration. Based on the need to process many images in a batch mode and the requirement to register the off-focus images using the in-focus sub-pixel translations, a complete defect segmentation procedure was generated in Khoros.

4.2 Defect segmentation and mask generation

A Khoros workspace was developed which segmented the defects in all three focus offsets and generated three defect masks per image in a batch mode of operation. The processing steps of this workspace include automatic selection of candidate tie points, sub-pixel registration, translation of defect image, subtraction, edge filtering, and mask generation. Figure 5 shows a block diagram of the processing steps in the Khoros workspace. A detailed description of each of these functional blocks will be described in detail below.

The input processing list is used to specify which images (TIFF files) are associated with the same defect. This ASCII file is derived from the spreadsheet described in the above section and shown in Table 1. A processing loop is established which iterates through each unique defect in the input list by assigning file name variables for the defect and reference images in all three focus offsets. These file name variables are used in the subsequent processing steps.

The first processing step is the calculation of candidate tie point locations. A gradient of pixel intensity in both the horizon and vertical directions is calculated and coordinate locations which have large magnitude in both orientations are selected as tie points. Two tie points are selected based on having the maximum gradient within an allowable search window. Figure 6 shows the regions of an image where the search for maximum gradient occurs. These two tie points are passed to the sub-pixel registration module.

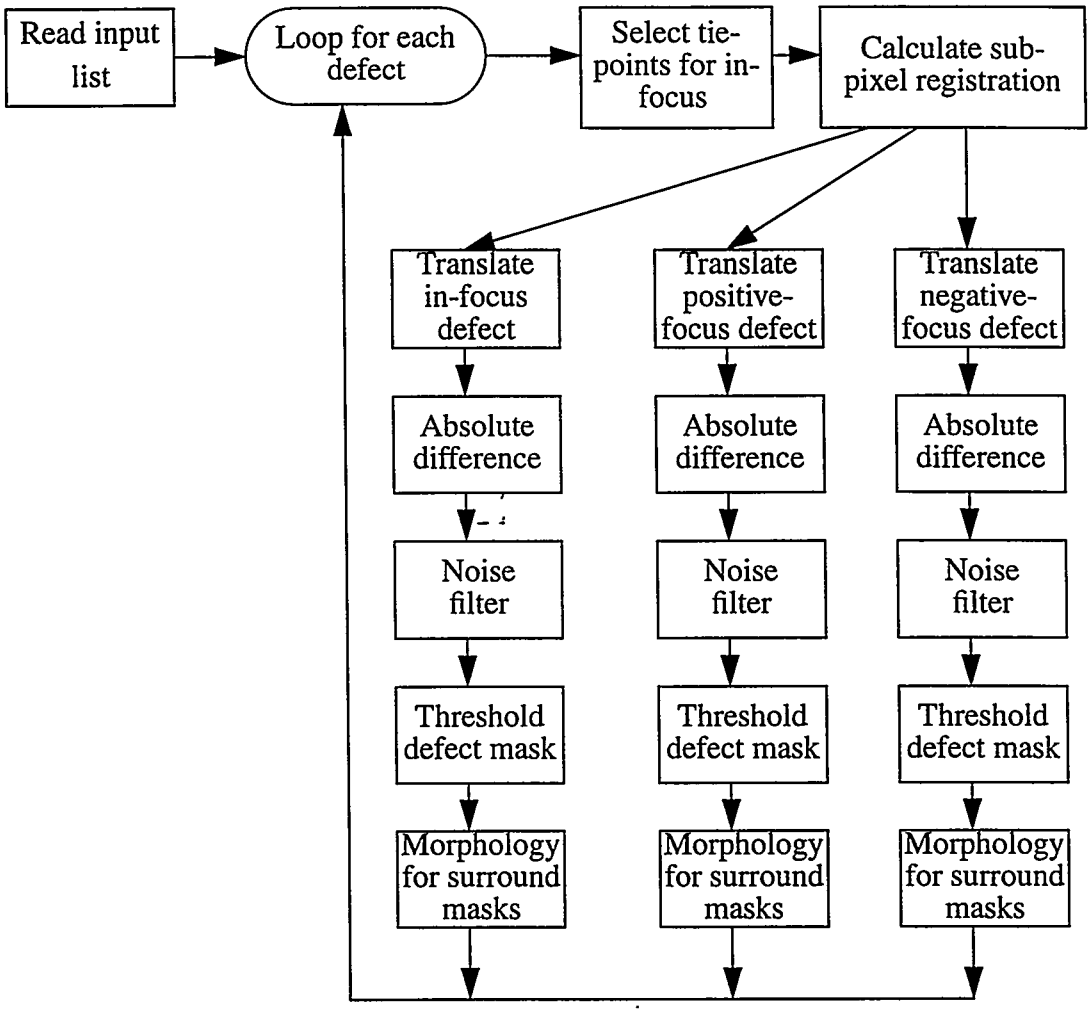


Fig. 5. Block diagram of multi-focus defect segmentation and mask generation.

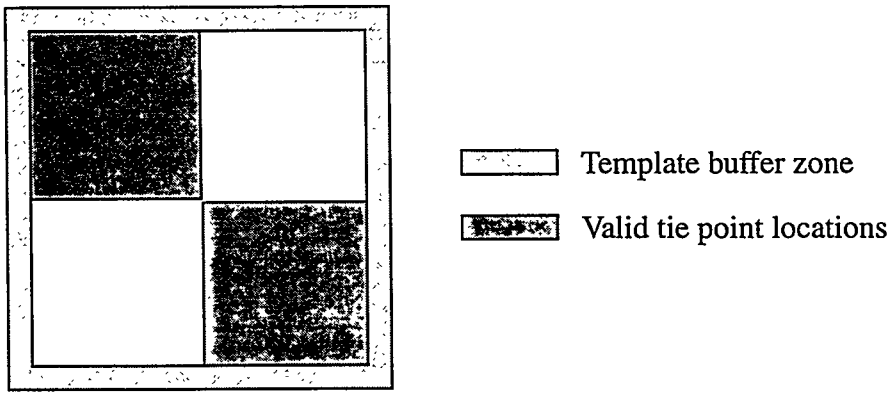
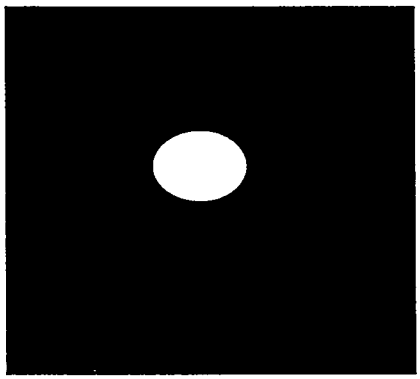


Fig. 6. Tie point selection regions

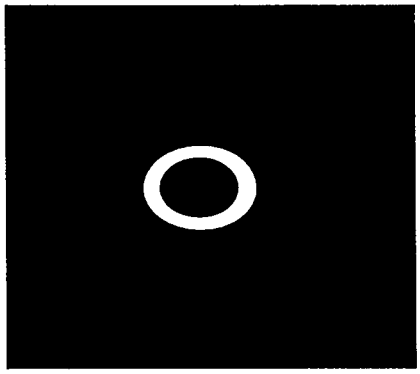
A two-dimensional normalized correlation is the first step performed in the registration module. This is a correlation between a rectangular template area in the reference image, centered on the tie point and larger rectangular search area in the defect image. The results of this correlation are a two-dimensional array of values that indicate the degree of similarity between the reference template and the section of the search area which was compared to the template. The relative integer shift of the defect image with respect to the reference is found by searching for the maximum correlation in this array. Once the maximum is found the second step in the registration module is to estimate the sub-pixel shift between the defect and reference images. This is accomplished by fitting an paraboloid to the surface of correlation values in a three by three region centered on the maximum correlation values. Once the parameters of the paraboloid have been calculated, a derivative of this surface is used to estimate the location of the true maximum value. A complete description of this procedure can be found in [1]. The sub-pixel shifts from each tie-point are averaged and passed to the translation modules.

A parallel processing path follows for each of the focus offsets in which the defect image is translated to match the reference, an absolute difference is computed, a noise filter is applied to the difference image, and masks are generated. The noise filter is composed of two parts: an edge attenuation operation and a median filter. The edge attenuation filter is a multiplication of the difference image by a image constructed as the first derivative of the reference image. This attenuation image has magnitude ranging from zero to one with zero corresponding to a constant intensity area (no attenuation) and one to a maximum contrast area (large first derivative). This attenuation has the effect of suppressing differences that are likely to be the result of slight alignment errors. The median filter which follows removes isolated small point noise. After this filtering the defect is segmented, and a binary mask is generated by applying a threshold to the difference image. A defect perimeter only mask is generated using morphological operations. An erosion and dilation of the defect mask are combined using an exclusive-or operator. The result is a mask that covers an area which just includes the perimeter of the defect. A final mask that is generated is the surround mask which is the exclusive-or of the defect mask and the twice dilated defect mask.

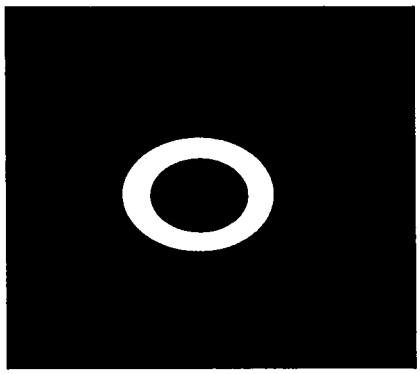
Figure 7 shows a representation of these masks.



(a) Defect mask



(b) Edge mask



(c) Surround mask

Fig. 7. Representation of masks generated from one focus offset defect image.

5.0 Feature Extraction Workspace

The Khoros feature extraction workspace uses the subpixel-aligned (warped) defect image along with its corresponding masks to make some feature measurements on the defect. As illustrated in Figure 7 (a) and (b), the masks used to define regions of interest are the defect mask and the defect edge mask. Only the pixels defined by the mask are considered in all of the following feature calculations. A functional block diagram shown in Figure 8 shows the defect image and the corresponding masks at the input. The same set of features are extracted for each mask, and an OLPARS feature vector is formed at the output of the workspace to be passed on to a classifier. The format of this feature vector is described in detail later in this section. Note that the feature extraction algorithms can be added on to the list without rebuilding the workspace.

5.1 Features Extracted

Currently, only four features have been extracted using the feature extraction workspace. These features are root mean square (RMS), area, contrast, and fractal dimension. As noted in Figure 8 by the "Other..." block, we will be adding additional extraction algorithms to the workspace in the future.

5.1.1 Root Mean Square

The RMS of the defect is given by Equ. (1).

$$RMS = \sqrt{\frac{\sum_{i=0}^{N} x_i^2}{N}} \tag{1}$$

Where N is the number of pixels and x_i is the sample value of x at location i. The RMS value is useful in distinguishing defects which are predominantly dark in intensity from those which are brighter overall or have small bright spots. Typically, fall-on particles tend to be dark overall, while previous layer particles and other defects have some brighter pixels. Because the RMS value sums the squares of the intensity, a few bright pixels can make a significant difference in RMS value.

5.1.2 Area

The size of the defect is simply a pixel count of the area defined by the full defect mask and the edge defect mask. The full defect mask area estimates the size of the defect. The edge defect mask pixel count give an indication of the area and perimeter of the edge of the defect. The ratio of the full defect mask area to the edge defect mask area is an estimation of the edge irregularity or roughness.

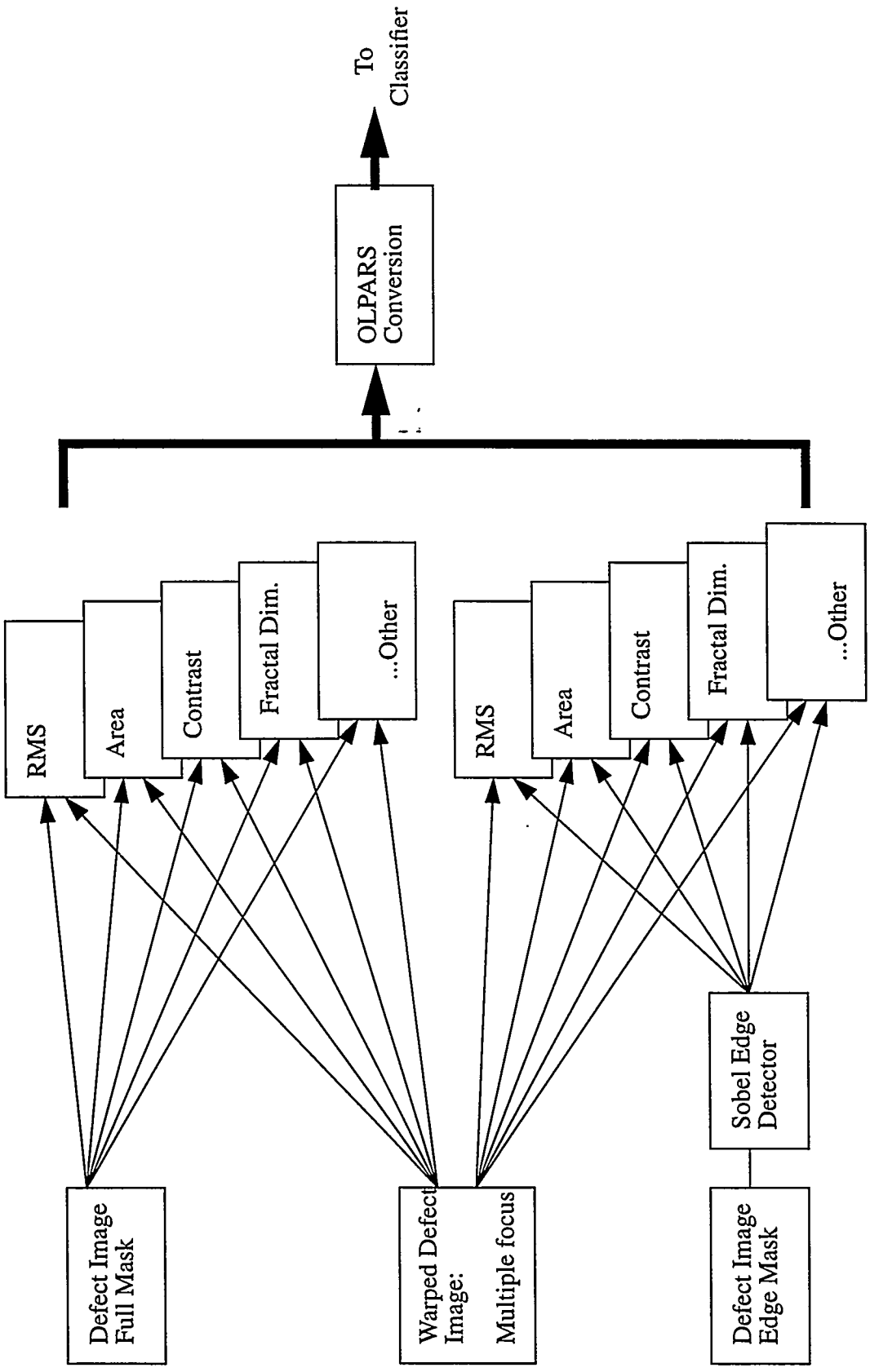


Fig. 8. Feature extraction block diagram

5.1.3 Contrast

The contrast measurement used is given in Equ. (2).

$$C = \sum_{x=0}^{n} \frac{x^2 H(x)}{N} \tag{2}$$

where

$H(x)/N$ = Number of pixels with gray level x normalized by the total number of pixels, N and

n = number of gray levels.

By rearranging Equ. (2), one can see that this implementation of the contrast measurement is simply the expected value of x^2, $E(x^2)$. If the normalized histogram, $H(x)/N$, is spread out over the whole range of gray levels, then this contrast measurement will be relatively high compared to a histogram with a narrow peak. There are other ways to measure image contrast based on gray level extrema that may be turn out to be more effective, but this measurement is being used initially.

5.1.4 Fractal Dimension

The technique used to estimate the fractal dimension is an extension of Mandelbrot's ruler method for measuring lengths of irregular curves. The input mask image is used to find the fractal dimension of only a portion of the image. Any nonzero pixel in the mask indicates that the corresponding pixel in the image will be included in the calculation.

The technique measures the area of the image surface using a "blanket" technique. By varying the blanket thickness, ε, and calculating the corresponding area, area(ε), a log-log plot of log[area(ε)] vs. log(ε) can be generated. A least squares fit of a line through three data points corresponding to ε-1, ε, and ε+1 is calculated. A fractal signature, *Sig(ε)*, is calculated for each set of three data points and epsilon is varied from a value of 1 up to the maximum number of iterations specified by the user. The slope, S, of the line is related to the fractal signature, *Sig(ε)*, by the following equation:

$$Sig(\varepsilon) = 2 - S(\varepsilon) \tag{3}$$

The fractal dimension, D, is calculated as the average value of the fractal signature.

5.2 Feature Vector Description

The four features described in Section 1.1 are extracted for both masks (full and edge) as well as for each of the three focus offsets (positive, zero, and negative offset). This produces a feature vector of length 24 (4 features x 2 masks x 3 focus offsets). After the features were extracted in stored in a Khoros file format, they were converted into OLPARS ascii feature vector files. A special Khoros function (glyph) was written and integrated

into Khoros to build the OLPARS format feature vector files. The structure of the feature vector is shown in Table 3.

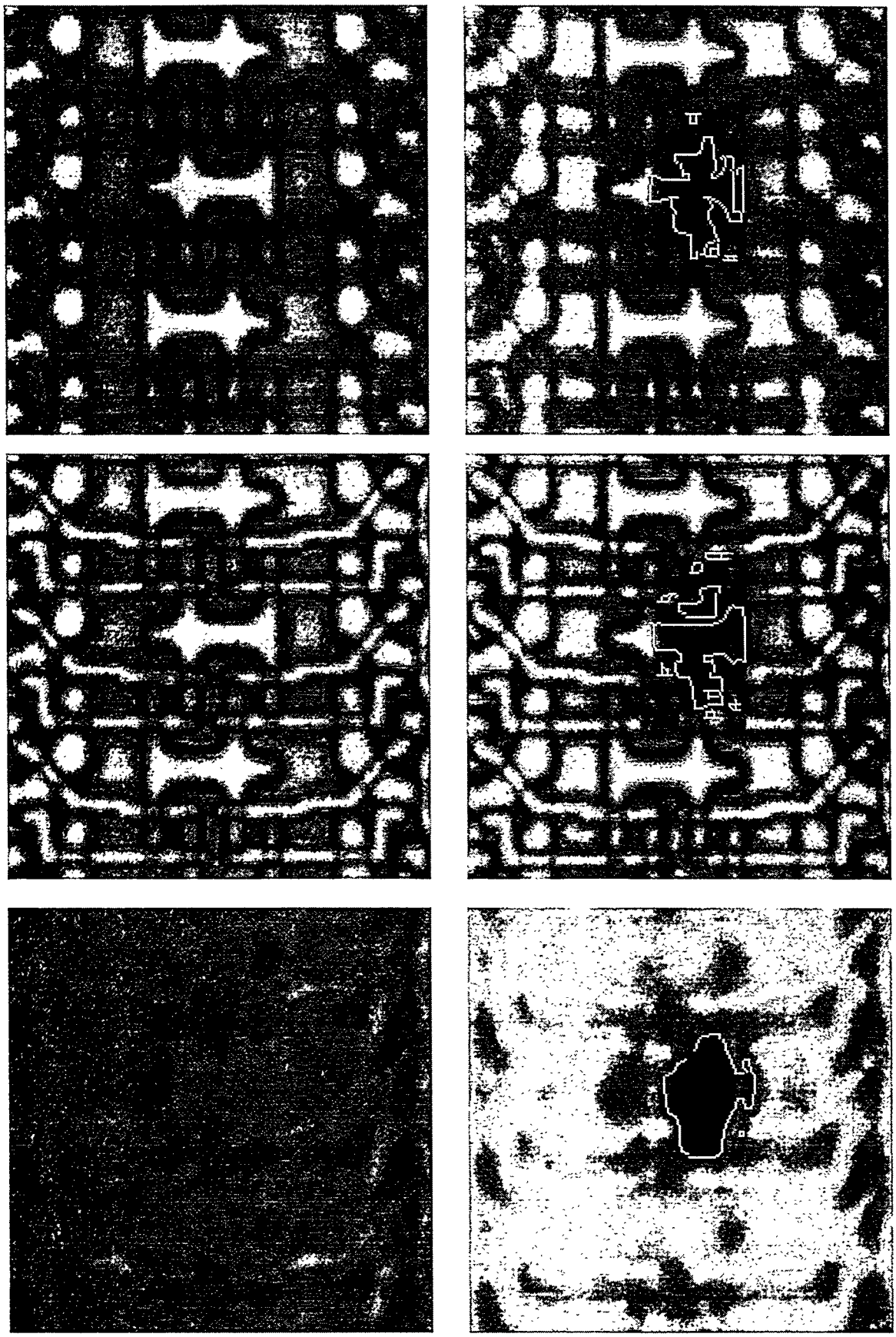
Table 3. Feature Vector Structure

Feature number	Feature Name	Mask Image	Focus Offset
1	RMS	Full	Negative
2	area		
3	contrast		
4	fractal dimension		
5	RMS	Edge	
6	area		
7	contrast		
8	fractal dimension		
9	RMS	Full	Zero
10	area		
11	contrast		
12	fractal dimension		
13	RMS	Edge	
14	area		
15	contrast		
16	fractal dimension		
17	RMS	Full	Positive
18	area		
19	contrast		
20	fractal dimension		
21	RMS	Edge	
22	area		
23	contrast		
24	fractal dimension		

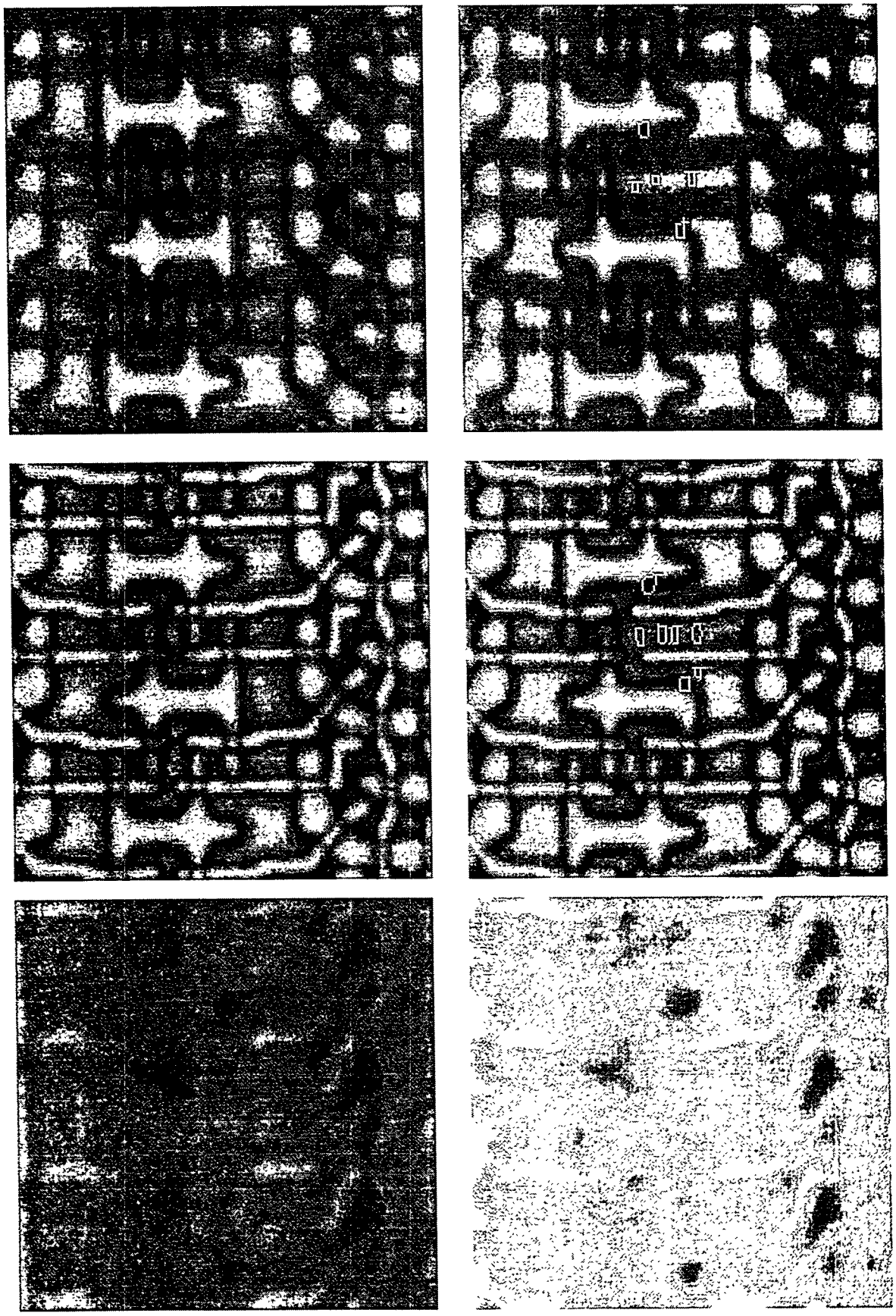
6.0 Viewing Segmented Defects

In order to better relate particular features in a vector to some visual clues in the image, a Khoros workspace was developed which aids in viewing one defect and reference image at various focus offsets. The ability to go back and make a visual study of the images from a given class of defects is extremely useful. For example, a situation that often occurs is the feature vectors from a particular class of defects may cluster tightly except for a small group of outliers. It is beneficial to be able to identify the images from which that small outlier group came from so that they can be studied for visual clues as to why they separated from the rest of the group. Another situation that often occurs is that one may find that a particular algorithm is not measuring the visual clue that it was expected to, or the measurement can be similar for two very different defect types. A very simple example of this is a contrast measure which looks only at the range of intensity values in the image. One defect that has a smooth, continuous gradient from white to black could have the same contrast measurement as a random intensity defect. Going back and looking at the images can indicate that a texture measurement would be more effective to separate classes. This is a simple concept, but it should not be overlooked when developing feature extraction algorithms.

This viewing workspace is an interactive system that allows the user to enter the image number for the defect they wish to view. This number can be read directly from the OLPARS feature vector scatter plots which makes identification and inspection of outlier images very simple. The workspace retrieves the images and displays all focus offsets for both the reference and defect images. Sample outputs from one bridging particles and one missing pattern defect are on the following pages. The white outlines on the defect images mark the perimeter of the defect as detected by the defect segmentation workspace.



~~CONTAINS GRADA PROTECTED INFORMATION~~



7.0 Summary

An overall system design was presented which integrates information about defect context within wafer layers along with defect feature descriptions to make a decision about the defect class. This approach came about as a result of viewing DEC wafers and realizing that the particle classifications were based on location of the particle rather than the characteristics of the particle itself.

A complete defect segmentation procedure based on a subtractive technique was built in the Khoros workspace. The workspace automatically selects registration tie-points and registers the defect and reference images to a subpixel shift value. It then performs a subtraction, noise filters and a threshold to generate a binary mask for all focus offset values. This binary mask is used to generate the three defect masks via morphology which define the full defect, the edge of the defect, and the surrounding area around the defect. This workspace allows us to run the segmentation procedure in batch mode without human intervention to select tie-points and specific thresholds.

A feature extraction workspace analyzes the defects at multiple focus offsets. Features such as RMS, contrast, area, and fractal dimension are being extracted. These features are being converted into OLPARS format for analysis. A set of vectors generated by this workspace on the DEC wafer images has been compiled and delivered to KLA. ORNL will also do some analysis of this data to look for possible automatic classification schemes.

8.0 References

1. Gleason, S. S., Hunt, M. A., Jatko, W. B., "Subpixel image registration based on paraboloid surface fit", SPIE Proceedings, 1989.

2. Peleg, et. al., "Multiple Resolution Texture Analysis and Classification", *IEEE Trans on Patt Anal and Mach Intell*, Vol. PAMI-6, No. 4, July, 1984.

Appendix C

Radial Basis Function Routine Descriptions

June 1994

Radial Basis Function Routine Descriptions

Shaun S. Gleason
Advanced Computation and Machine Vision Group
Oak Ridge National Laboratory

1.0 Introduction

This is documentation for the radial basis function (RBF) C code. This document describes how to install, build, and use the software. The mathematics implemented in the programs are described here so that the user may intelligently apply these routines to their own data as well as interpret the results.

2.0 File Inventory and Installation

The accompanying file *rbf.tar.mail* is a uuencoded, UNIX compressed, and tarred file that contains all of the C files, header files, and sample data. To decode the file type:

```
your_machine% uudecode rbf.tar.mail
```

This will create a file called rbf.tar.Z that can be uncompressed and untarred by typing:

```
your_machine% zcat rbf.tar.Z | tar xvf -
```

A directory called rbf will be created that contains all code and header files. Also, a directory called rbf/olpars will be created that contains sample data files.

The C routines provided with this software are the following:

1. *rbf_train.c*
2. *rbf_eval.c*
3. *gauss.c*
4. *norm.c*
5. *svdcmp.c*
6. *svbksb.c*
7. *pythag.c*
8. *nrutil.c*

The accompanying header files are:

1. *rbf.h*
2. *nrutil.h*
3. *nr.h*

Also included in the distribution is a Makefile with targets for **rbf_train** and **rbf_eval**. The executables are not included in the distribution, so they must be built by typing:

```
your_machine% make all
```

C files 1 through 4 were written specifically for the RBF classifier. The routines in files 5 through 8 were extracted from a collection of software provided as a supplement to the book [*Numerical Recipes in C, The Art of Scientific Computing,* W. H. Press, et. al., second edition, Cambridge University Press, 1992]. The *rbf.h* header file was written specifically for this application, and the *nrutil.h* and *nr.h* files were also provided with the numerical recipes collection.

3.0 Radial Basis Function Math

3.1 rbf_train.c

This function builds one RBF classifier for each class based on a training set of feature vectors. The classifier is constructed by summing a collection of Gaussians (radial basis functions) in the form

$$R_i(x) = e^{-\left(\frac{\|x - c_i\|}{w}\right)^2} \tag{1}$$

where i specifies one Gaussian function with center c_i. To form the RBF classifier, a group of Gaussians must be summed together evaluated at the feature vector, x.

$$RBF(x) = \sum_{i=1}^{n} A_i R_i(x) \tag{2}$$

The unknowns that must be selected to completely specify this family of Gaussians are the centers (c_i), the width (w), and the amplitudes (A_i). The centers of the Gaussians in the accompanying C code were selected as the feature vectors of the training set. This is a common practice when using radial basis functions for data interpolation when a function value is known only at specific coordinates. One obvious problem is that if your training set is large, there are more computations in the RBF evaluation and the determination of the RBF coefficients, A_i. The amount of time to find all of the coefficients can become significant. A better method may be to choose fewer, yet more representative centers to start with. In ["Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", S. Chen, C. F. N. Cowan, P. M. Grant, *IEEE Trans. on Neural Networks*, Vol. 2, No. 2, March, 1991] a method of choosing RBF centers is given that attempts to optimize the number and location of RBF centers automatically. This has not been implemented at this time, but if the RBF technique shows promise, the least squares technique can be implemented.

The width or standard deviation of the Gaussians, w, can be unique for each Gaussian in the RBF combination. The implemented code assumes a uniform width for all Gaussians that make up an RBF classifier for a specific class of data. This width was chosen as half the average distance between neighboring feature vectors within a class. Relating again to function interpolation, if data points are selected as centers, then the width should be wide enough to provide some overlap between the tails of the Gaussians, but not so much width that one Gaussian affects a large range of function values. At the other end of the spectrum, if the Gaussian is too narrow, the function will not provide a smooth interpolation of the points.

The only elements left to be found are the amplitude coefficients, A_i. These coefficients were found by solving the overdetermined system of equations

$$Ea = y \tag{3}$$

where $\boldsymbol{E}$ is an m by n matrix of the form

$$\boldsymbol{E}= \begin{bmatrix} R_1(x_1) & \dots & R_n(x_1) \\ \dots & \dots & \dots \\ R_1(x_m) & \dots & R_n(x_m) \end{bmatrix} , \tag{4}$$

a is an n by 1 coefficient vector of the form

$$a = \begin{bmatrix} A_1 \\ \dots \\ A_n \end{bmatrix}, \tag{5}$$

and y is an n by 1 output vector of the form

$$y = \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix}. \tag{6}$$

The y_i are 1 if the feature vector is in the current class of interest and they are 0 if the current training vector is not in the class. In these matrices, n is the number of centers (or Gaussians in the RBF), and m is the number of feature vectors used in the training set for this particular RBF classifier. In order to have an overdetermined system m must be greater than n.

The training is accomplished by choosing m feature vectors of which some are in the class of interest ($y_i = 1$) and others which are not ($y_i = 0$). The $\boldsymbol{E}$ and y matrices can then be formed, and the coefficients can be found through a least squares technique. The technique implemented in this code is singular value decomposition.

By typing **rbf_train** at the prompt, the following usage statement will be printed:

USAGE: rbf_train [directory] olpars_tree_file output_file

directory: (optional) where the olpars data resides

not necessary if in current directory

olpars_tree_file: name of the olpars tree file

output_file: name of coefficient output file for rbf_eval

The output file will contain the center locations, the widths and all coefficients for the RBFs for each class. This output file must be available to evaluate the classifier using the

rbf_eval function. The output file is an ascii file and has descriptive labels which makes it somewhat readable for interested parties.

3.2 rbf_eval

This function uses all of the information in the output file generated by **rbf_train**. This function takes a list of feature vectors and runs each one through all of the RBF classifiers for all possible classes. The feature vector is assigned to the class associated with the RBF classifier that has the highest output value. This routine forms the $\boldsymbol{E}$ matrix using the widths and centers generated by **rbf_train**. It then multiplies $\boldsymbol{E}$ by the coefficient vector $\boldsymbol{a}$ to form the output vector, $\boldsymbol{y}$.

By typing **rbf_eval** at the prompt, a usage statement is printed out as follows:

USAGE: rbf_eval [directory] olpars_tree_file coeff_file output_file

directory: (optional) where the olpars and coefficient data reside

not necessary if in current directory

olpars_tree_file: name of the olpars tree file

coeff_file: name of coefficient file generated by rbf_train

output_file: name of class output file

The following is a sample excerpt from an output file generated by rbf_eval:

Class 1 feature vectors:

fv# 1: c1: 0.000000 c2: 0.041704 c3: 0.000009 c4: 0.000000 class:2

fv# 2: c1: 0.038991 c2: 0.022343 c3: 0.000000 c4: 0.020794 class:1

fv# 3: c1: 0.001336 c2: 0.000000 c3: 0.000081 c4: 0.000000 class:1

fv# 4: c1: 0.000000 c2: 0.000000 c3: 0.000031 c4: 0.000000 class:3

fv# 5: c1: 0.068704 c2: 0.002496 c3: 0.000002 c4: 0.033830 class:1

fv# 6: c1: 0.069731 c2: 0.002501 c3: 0.000001 c4: 0.032924 class:1

fv# 7: c1: 0.112655 c2: 0.001419 c3: 0.000001 c4: 0.045545 class:1

fv# 8: c1: 0.596007 c2: 0.004173 c3: 0.000001 c4: 0.134130 class:1

fv# 9: c1: 0.010265 c2: 0.000010 c3: 0.000000 c4: 0.000528 class:1

...

Class 1 stats: correct: 73.214287 % incorrect: 26.785713 %

The first column is the feature vector number. Columns 2 through 4 give the RBF score for the feature vector evaluated by each of the RBF classifiers for the four classes in this particular problem. The last column gives the class number assigned to the feature vector based on the highest score. At the end of each class group, the statistics of percent correct and incorrect are given.

3.3 gauss.c

This function takes as input a vector, a Gaussian center, a vector dimension, and a Gaussian width (standard deviation). It evaluate the specified Gaussian (see Eq. 1) and returns the result.

3.4 norm.c

This function calculates the distance between two input vectors of a given dimension. This function is called by *gauss.c* in the exponent evaluation, and it is also called by *rbf_train.c* to calculate the neighbor-to-neighbor distance that is needed for the Gaussian width determination.

3.5 svdcmp.c, svbksb.c, pythag.c, nrutil.c

All of these functions are needed to perform the singular value decomposition on the overdetermined system of equations specified in Eq. 3. The decomposition is performed in *svdcmp.c*, and the calculation of the coefficients is done via back-substitution in *svbksb.c*.

4.0 Sample Data

Some sample data has been provided to test the routines for proper execution. The training data is in an OLPARS format tree file called *gauss2_train.tre*. It contains three classes of three-dimensional feature vectors. Another data set called *gauss2_eval.tre* should be used to evaluate the performance of the classifier. This data is well clustered and should be classified to greater than 99 percent accuracy using **rbf_train** and **rbf_eval**. An example application output is provided here for illustration of proper use. In this example the executables are in the current directory and the data are in the directory ./olpars. All of the output is given here for comparison purposes.

Here are the results of running **rbf_train** on the *gauss2_train.tre* data set.

```
medusa% rbf_train olpars gauss2_train.tre gauss2_coef.dat
Reading OLPARS tree file olpars/gauss2_train.tre...
 reading olpars/gauss2_1_tr.dat
 reading olpars/gauss2_2_tr.dat
 reading olpars/gauss2_3_tr.dat
finished.
Average neighbor-to-neighbor distance: 0.009804
Building RBF matrix for class #1...finished.
Peforming lsq for RBF coefficients for class 1...
 singular value decomposition...
 back substitution...
finished.
Average neighbor-to-neighbor distance: 0.009198
Building RBF matrix for class #2...finished.
Peforming lsq for RBF coefficients for class 2...
 singular value decomposition...
 back substitution...
finished.
Average neighbor-to-neighbor distance: 0.011213
Building RBF matrix for class #3...finished.
Peforming lsq for RBF coefficients for class 3...
 singular value decomposition...
```

```
 back substitution...
finished.
```

At this point we want to run **rbf_eval** on the *gauss2_eval.tre* data set.

```
medusa% rbf_eval olpars gauss2_eval.tre gauss2_coef.dat gauss2_class.dat
Reading OLPARS tree file olpars/gauss2_eval.tre...
 reading olpars/gauss2_1_ev.dat
 reading olpars/gauss2_2_ev.dat
 reading olpars/gauss2_3_ev.dat
finished.
```

Now the results can be viewed in the output file *gauss2_class.dat.*

Appendix D

Semiconductor Yield Improvement Technical Progress Report
August 1994

~~(Protected CRADA Information)~~

ORNL

SEMICONDUCTOR YIELD IMPROVEMENT TECHNICAL PROGRESS REPORT

August, 1994

S. S. Gleason
M. A. Hunt
H. Sari-Sarraf

~~CONTAINS CRADA PROTECTED INFORMATION~~

ORNL

SEMICONDUCTOR YIELD IMPROVEMENT TECHNICAL PROGRESS REPORT

S. S. Gleason M. A. Hunt H. Sari-Sarraf
Instrumentation and Controls Division

August 1994

Prepared for the
ORNL/KLA Automatic Defect Classification CRADA
under CRADA No. ORNL92-0140
between the U.S. Department of Energy, Martin Marietta Energy Systems and
KLA Instruments Corporation

Prepared by
Instrumentation and Controls Division
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6011
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

1.0 Executive Summary

This is a technical progress report written by the ORNL staff for KLA Instruments Corporation as part of the Automatic Defect Classification (ADC) Cooperative Research and Development Agreement (CRADA). The content of the report describes technical approaches applied to defect classification on a set of wafer images supplied by the CRADA partner. The main focus has been on background modelling and segmentation in order to identify defect types (mainly particles) that are classified using information about the location of the defect on the wafer. Because the defect has already been detected and located by a wafer inspection unit before it comes to the ADC system, the particular background pixels that the defect is overlapping can be studied in the spatially registered reference image.

Multi-color space analysis was studied as a potential technique that could be used to identify and label specific types of wafer backgrounds. Representing the full color images in both RGB (red, green, blue) and HSV (hue, saturation, value) color spaces was effective in segmenting various layers in the wafer image. By considering the RGB and HSV values of individual pixels, several classifiers were designed to perform background structure identification based on these six parameters.

In an effort to improve the accuracy of wafer layer classification techniques that are based on color information, a wafer image was converted to an image of connected regions of same layer material. The average color values for all of the pixels in each blob were then used as the representative feature description for the entire connected region. Another improvement in this application was the use of excess color images in which one particular color band in the image is magnified based on its potential for separating different layers.

Another similar approach to background classification was applied which uses an adaptive thresholding technique as a first pass to roughly segment the image based on color information. An independent operation generates another edge-based segmented image to create the connected regions image. A final step fuses the two segmentations using a combination of tests for spatial overlap, logical operations and finally an application of a within-region intensity filter.

ORNL staff spent approximately three man-weeks on-site at KLA Instruments in San Jose, CA during the month of July, 1994 working directly with their technical ADC team. A large amount of technology was transferred to KLA during the visit in the form of information, tools, and software. The visit also benefitted ORNL staff by helping us better understand KLA's overall ADC strategy. ORNL staff was also able to work with the KLA inspection equipment first hand and gather a new set of defect images.

2.0 Contextual-based Defect Classification System

In the May, 1994 progress report an overall ADC system architecture was presented that combines information from the defect itself along with contextual information about the location of the defect to come up with a final class label. The diagram is reprinted in Fig. 1 for reference. Note that the top of the diagram presents defect analysis based on features extracted from the defect itself. The lower portion of Fig. 1 presents a parallel contextual analysis of the defect. For a complete description of the overall approach shown in Fig. 1 refer to the May, 1994 technical progress report.

The efforts described in the following sections focus on the defect context identification. We are assuming for the purposes of the partner-supplied wafer image set that the particle defect can be classified based on their relative location to the various layers on the wafer. This requires automatic identification of the background structures in the wafer image to determine which layer(s) that the defect is affecting. The different layers contained in the images presented in this report consist of active area, poly lines, field areas, and a boundary zone separating one layer of material from another. As an example, a defect that overlaps two separate poly lines is considered a bridging particle. Just as identifying the defect is an object classification problem, so is automatic identification of the background structures. Features of the wafer structures must be extracted and analyzed to determine the types of layers that are affected by a particular defect.

Depending on the specific classification scheme being implemented in an ADC system, certain defect types will need to be classified based on information from the defect itself along with the contextual information around the anomaly described in this report. The technologies discussed here would then provide one essential piece of the classification puzzle.

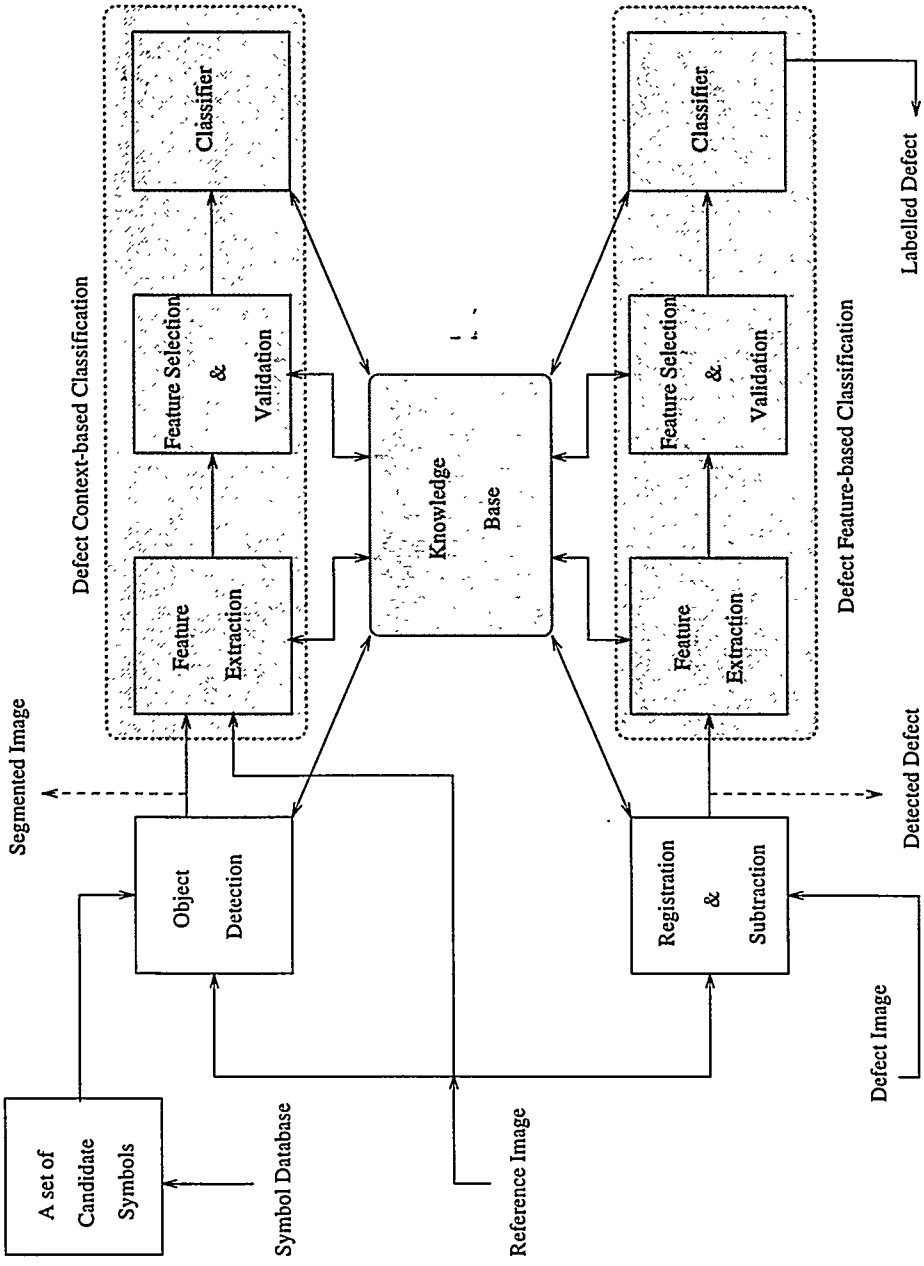


Fig. 1. Automatic Defect Classification System Block Diagram

3.0 Multi-Color Space Analysis

3.1 Color space transforms

The high resolution color images of wafer defect sites are typically represented in three spectral specific bands such as red, green, and blue (RGB). Most electronic color CCD cameras use either three independent CCDs each with a colored filter or a striped filter over a single CCD. For some analysis purposes it is difficult to deal with the triplet color value at each pixel because the physical perception of adjacent pixels is not well represented. The color of a object is usually described as a shade represented on a color wheel, such as red, purple, orange, etc. while the brightness of the color is described as a separate quality. One color transformation converts the RGB triplet into a hue, saturation, and value (HSV) representation. In this space the hue represents the actual color on the color wheel (in radians), saturation represents whether the color is pastel (light, mixed with white) or vivid (dark, single pure color), and value is the overall intensity of the pixel (similar to a single gray level). Figure 2 shows a comparison of the three individual RGB bands and the HSV representation. One issue that becomes apparent in the HSV space is that the Hue is measured in radians and has a discontinuity at the 2Pi boundary (values are modulo 2Pi). An example of this can be seen in the lower left image of Figure 2 where the gray scale intensity switches from black to white in a small local region. This means that all averaging of individual pixel values must be done by summing all the values and then performing a modulo 2Pi operation. Also, distance measures in feature space must be performed by adding the two points then performing a modulo 2Pi operation. The saturation image of Figure 2 (lower row, center) shows a representation which has good contrast and uniformity for the three layers. An advantage to the HSV space is the specific colors in the image may change from wafer to wafer (and thus the RGB values), but the differences between layers will remain fairly consistent in the HSV space.

The excess RGB transformation is another linear transform which enhances the representation of the different layers in a color wafer image. Figure 3 shows the three excess gray scale representations. These excess images are computed by taking two times the primary color (Red in excess Red) and subtracting the other two bands. This transform emphasizes the relative magnitude of one color with respect to the other two primaries. The excess Red image shows a strong contrast between the field layer and other layers. The excess Blue image shows a strong contrast between the poly lines and the other layers. By using these images in the segmentation process, a more accurate and robust algorithm may be developed.

3.2 Pixel-based segmentation

At the lowest level of color analysis, each individual pixel in the image is considered as an element of the wafer background structure. Using an intensity image that was segmented via a zero-crossing algorithm described in Section 3.3, each individual area was hand-labelled with its proper layer by assigning one intensity to each of the different layer materials (active area: black, field: dark gray, boundary zone: light gray, and poly: white). The intensity image and corresponding labelled image are shown in Fig. 4.

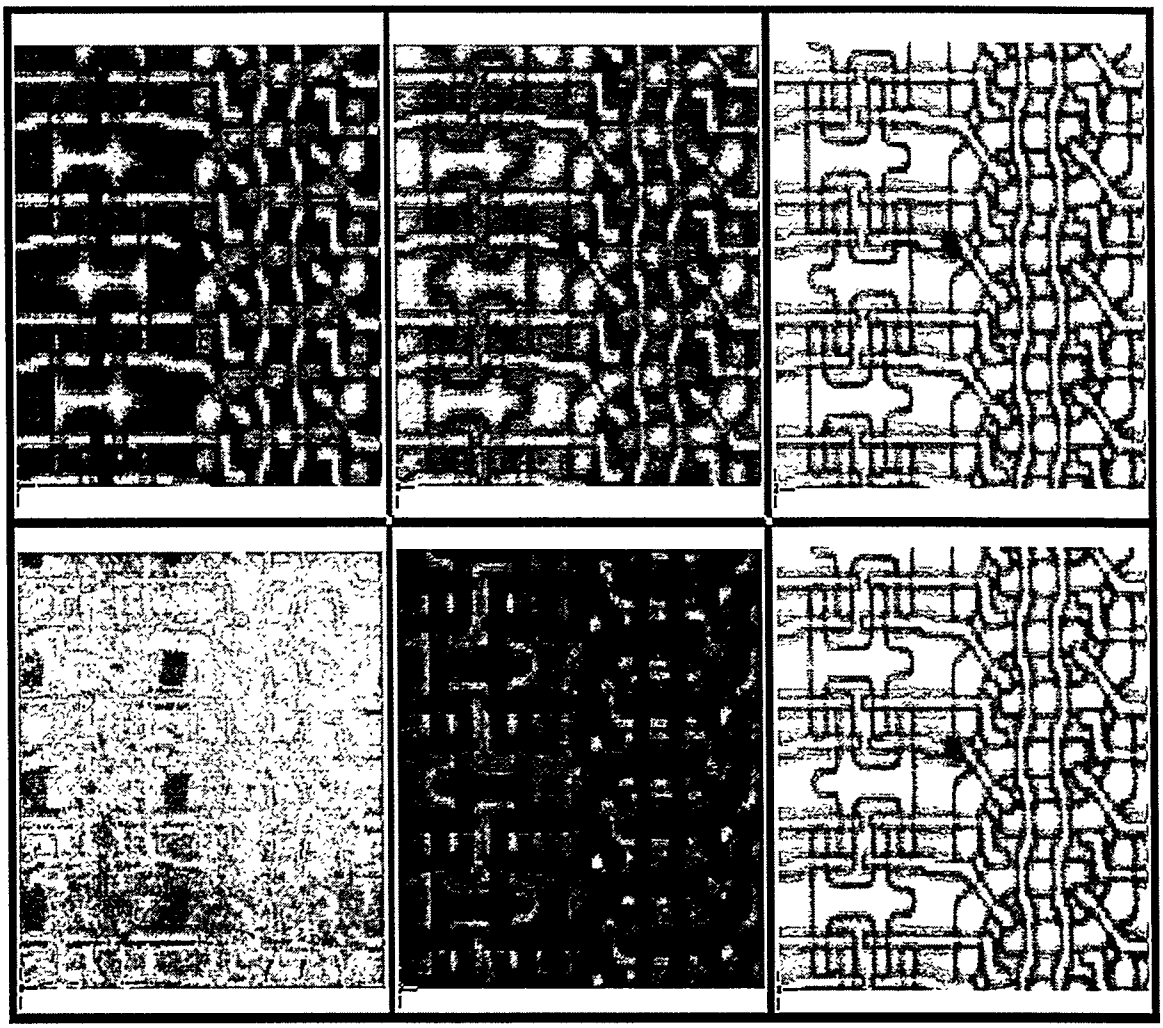


Fig. 2. Gray scale representations of each individual RGB and HSV bands. Top row, left to right: Red, Green, Blue. Bottom row, left to right: Hue, Saturation, Value.

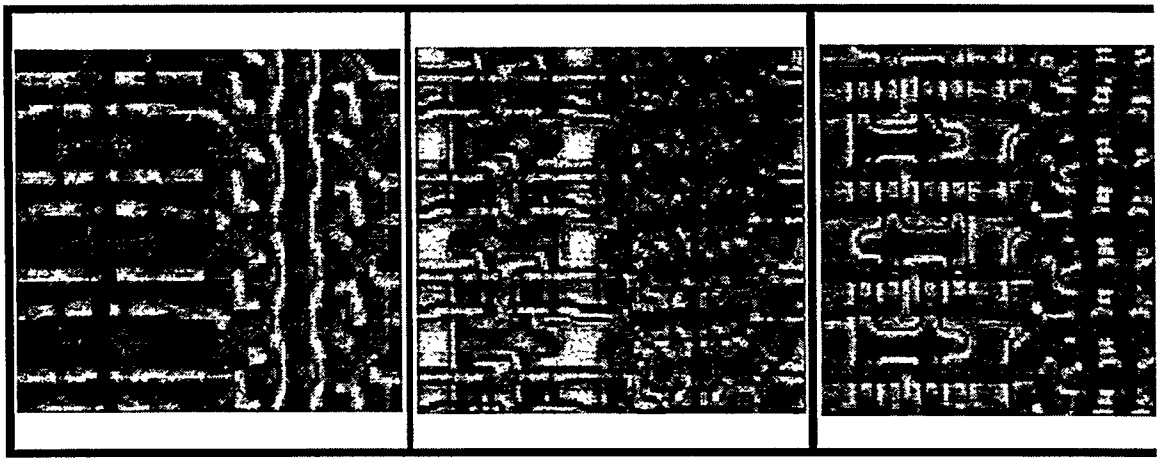
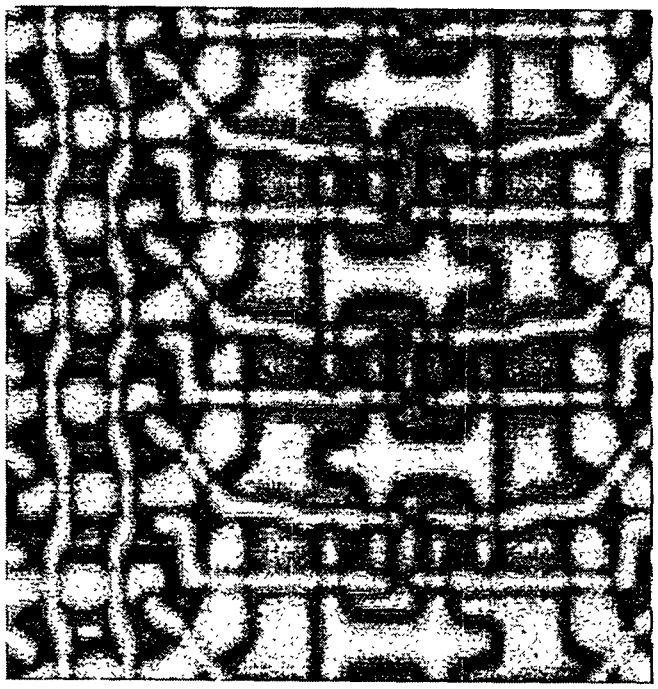
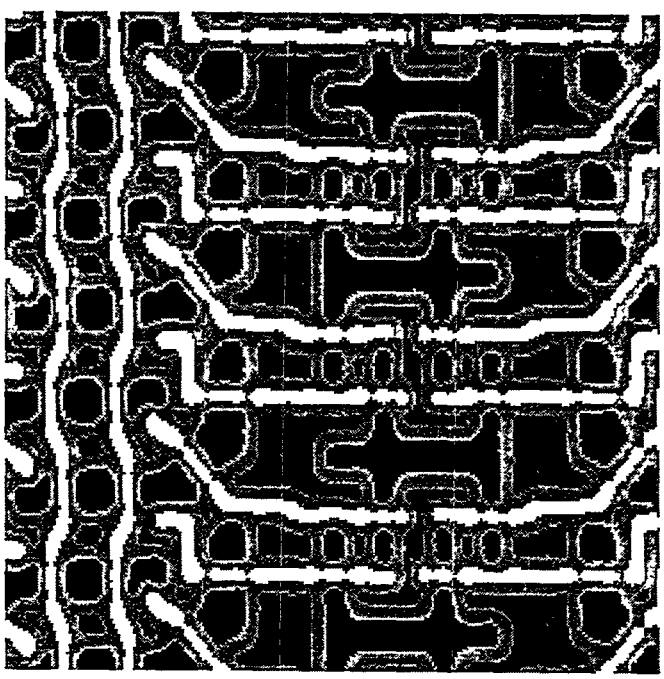


Fig. 3. Gray scale representations of each individual excess RGB band. Left to right: Excess Red, Excess Green, Excess Blue.



(a)



(b)

Fig. 4. (a) Intensity image of wafer. (b) Hand-labelled image of wafer in (a). The layers are labelled via intensity from black to white as follows: active area, field, boundary zone, and poly lines.

Software was written that reads an intensity image and its associated label image, determines the number of regions from the label image, computes the RGB-to-HSV transform, and outputs an OLPARS format file containing feature vectors for each pixel in the image. Each feature vector contains six parameters: red, green, blue, hue, saturation and value. Figure 5 shows a coordinate projection of the data with saturation on the x-axis and hue on the y-axis One can see that the regions do cluster and are separated fairly well in this

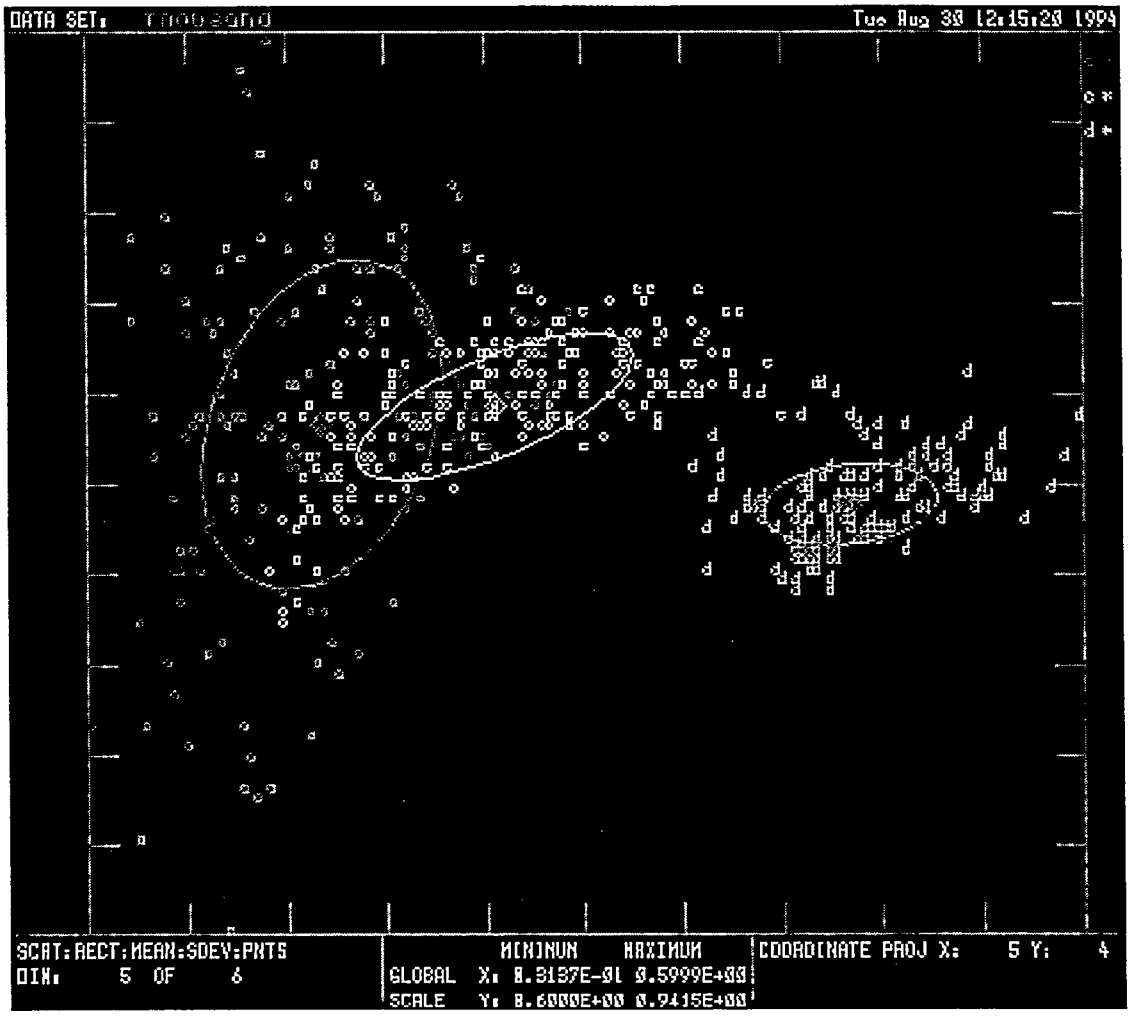


Fig. 5. Coordinate projection of saturation and value for 1000 sampled pixels from a wafer image. The dark gray cluster on the left is poly pixels, the white cluster is active pixels and the light gray cluster on the right is field pixels. The ellipses represent the standard deviation for each class.

coordinate view. There is an overlap as expected because the connected regions in the image corresponding to particular layers are not uniform in color. There are significant color transitions along the edges of the regions which cause the feature vectors for the pixels near the edge to overlap with neighboring clusters. This overlap will increase if the color varies from image to image resulting in some classification ambiguity. This could especially be a problem for the poly and active areas which already have a substantial

overlap. In the next section a technique will be introduced that will make the clusters more compact and create a better separation.

Another OLPARS file that contains a representative sample from each of the different layers in the same image has a total of 1000 feature vectors. Several classifiers were designed using 500 of these 1000 feature vectors. The remaining 500 were used to test the accuracy of each classifier. The results are shown below in Table 1. The neural network classifier

Table 1: Classifier performance (% correct) on pixel-based color segmentation.

Classifier	poly	active	field
Nearest Mean	80	80	100
Fisher Pairwise	86	97	99
Backpropagation Network	92	94	99

has the best overall performance of the three tested.

A contextual defect classification scheme using this approach would use the location and extent of the defect along with the spatially registered reference image to find the pixels that the defect is "covering up". These pixels would then be extracted from the reference image and transformed into HSV space. The RGB and HSV pixel values would then be fed into the pixel-based classifier as a feature vector. The particle would then be labelled with a particular class if a certain pre-defined percentage of the reference image pixels are labelled with that class.

3.3 Region based segmentation and labeling

The correct classification of many semiconductor defects relies on knowledge of what process layer the defect occurs on. As discussed previously, the complete classification of a defect will require both the characteristics of the defect itself and the context in which the defect is located. This section describes a method of segmenting an image of the reference site into connected components (i.e. blobs) and then labeling these components with the correct process layer. Color characteristics described in a previous section will be used as the sole basis for the labeling process. Three steps are required to develop a label for a given location: (1) segmentation of the image into connected regions or blobs; (2) calculation of color based features of each blob; and (3) classification of the entire blob based on the calculated features. The following section will describe each of these steps in sequence.

Looking at semiconductor images reveals an intensity change across the boundary between different layers. Using this fact a segmentation of the input image begins with computing the zero crossing points in the non-directional second derivative. The zero crossing points represent the estimated location of the boundary between two regions with differing intensity. The Laplacian of the Gaussian filter [1] can be used to provide the nec-

essary boundary information from the grey-level reference images. The impulse response of the filter is given as:

$$\nabla^2 G(x, y) = \nabla^2 \left\{ \exp\left(-\frac{x^2 + y^2}{2\sigma^2} \right) \right\}$$

$$= \left(\frac{(x^2 + y^2) - \sigma^2}{\sigma^4} \right) \exp\left(-\frac{x^2 + y^2}{2\sigma^2} \right)$$

This filter is implemented using an algorithm proposed by Marr and Hildreth [2] based on the Difference Of Gaussians (DOG) method. The DOG filter approach to edge detection, or more precisely, its zero-crossings, offers a number of inherent advantages over gradient-based edge detection techniques. These include generation of thin and localized edges, simultaneous reduction of noise, and access to information at different resolutions by controlling the widths of the Gaussian filters.

The lower left image in Figure 6 shows an overlay of the zero crossings of this second derivative on a gray scale intensity representation of the RGB image. The resulting zero crossing points form a continuous line around geometric objects within the image. A single parameter in the DOG method controls the size of the Gaussians and thus the minimum intensity transition which will generate a zero crossing point. The method is very effective in the presence of small, slowly varying intensity regions.

A binary image of the zero crossing points (white) and the background (black) can be used as the input to a connected component algorithm. This algorithm uses 4-connectivity to identify individual pixels as members of the same region. Each separated component or blob is given a unique label (numeric value ranging from one to the total number of blobs). The KBVision task RegLabIm performs this operation. The upper right image of Figure 6 shows a representation of these individual blobs with the intensity of the blob representing the average excess blue value of each blob. After this operation the segmentation step is complete.

The next step is to compute features for each individual blob using the color information derived from the original RGB image. A table is generated which contains a single row for each blob identified in the previous step and several columns which represent calculated features such as the mean and standard deviation of the excess red values for that given blob. Additional columns include the mean and standard deviation for the other excess colors, the HSV values, and several shape descriptors of the blob (e.g. elongation, perimeter-to-area ratio). The lower right image in Figure 6 shows a two-dimensional histogram of the excess red and the excess blue means for all blobs in the image. The number of occurrences of a particular combination of means is represented by the intensity of the grid cell. If no blob has a particular combination of means, then that cell is not filled. Note that the blobs segmented in this image fall into three distinct clusters. Further investigation reveals that each cluster is representative of a given layer. A KBVision data flow graph is shown in Figure 7 and was used to perform the operations discussed in the first two steps. The flow of data is from left to right and top to bottom. Specific tasks include:

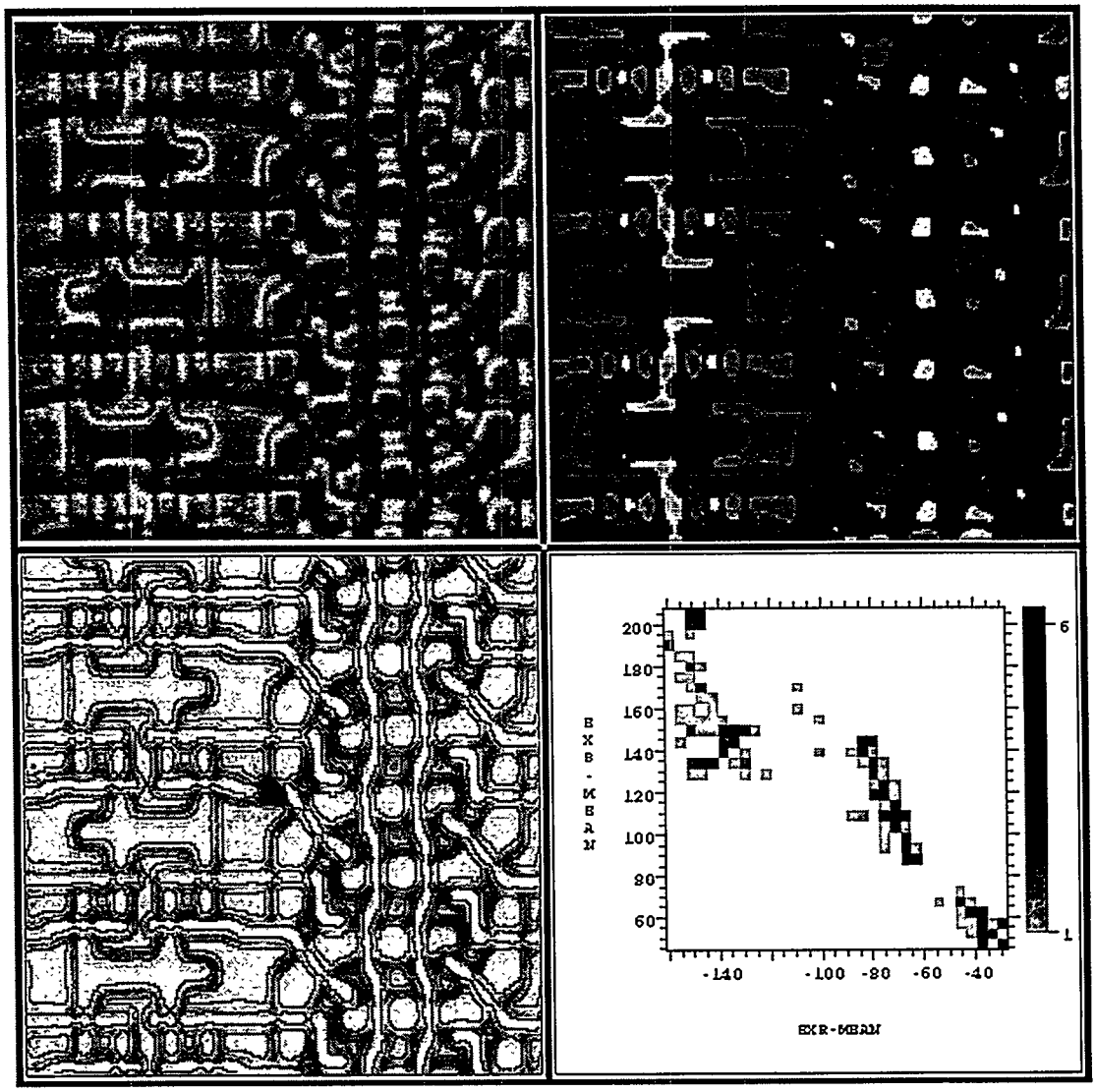


Fig. 6. KBV image examiner window. Lower left: Zero crossing points(black) overlaid on value image. Upper left: excess blue image. Upper right: gray scale representation of the mean excess blue level for each region. Lower right: Two-dimensional histogram of mean excess red and mean excess blue, each cluster represents a particular layer.

FeatColor, to compute excess and HSV values; RLabToTks, to convert connected components into token representation; TksShape, to calculate shape features; TksFeat, to calculate color based features (mean and standard deviation); and the ImgExam to view results.

The final step involves building a classifier to assign the particular process layer to each blob based on the measured features. This task was accomplished using the algorithms in the OLPARS system. An ascii output of the feature table described above was generated and read into OLPARS for analysis and the generation of a classifier. Figure 6 shows a two-dimensional projection of the excess red and blue mean features. As was evident in Figure 6, these clusters are separated and fairly compact. A nearest mean classifier was

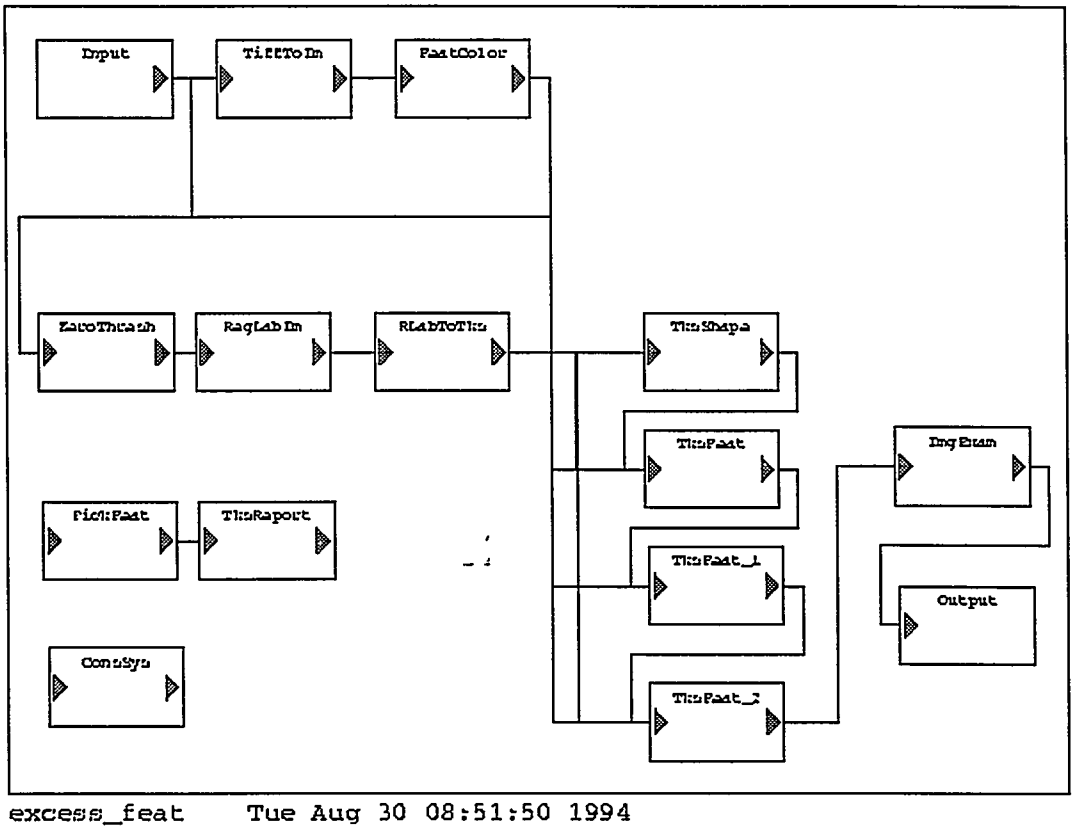


Fig. 7. Figure x4. KBVision data-flow diagram of tasks which segmentations and performs feature calculation for each segment.

generated by using a random selection of half of the blobs present in the image as the training set. The remaining blobs were evaluated with this classifier without any errors.

Application of this classifier to another image from the same wafer yielded the correct classification at an overall rate of 92%. Similar classification results were obtained using the mean hue and saturation values. A coordinate projection of saturation vs. hue is shown in Fig. 9. Compare the clustering in this plot to the individual pixel hue vs. saturation plot shown in Fig. 5. Note that by using the average hue and saturation values over uniform material regions shown in Fig. 6 the clusters become more compact and better separated. The accuracy of the classifier could possibly be improved if additional color and shape features were included in the classifier design. Additionally, the classifier could be trained using features from several images which would tend to increase the capacity of the classifier to handle slight image to image variations.

Overall, it appears that labeling of process layers is possible based on region features. The use of alternate color spaces (Excess and HSV) enhanced the ability to extract features which characterized specific layers. By using statistical features such as the mean, the individual clusters formed by each layer are more compact and better separated from one another.

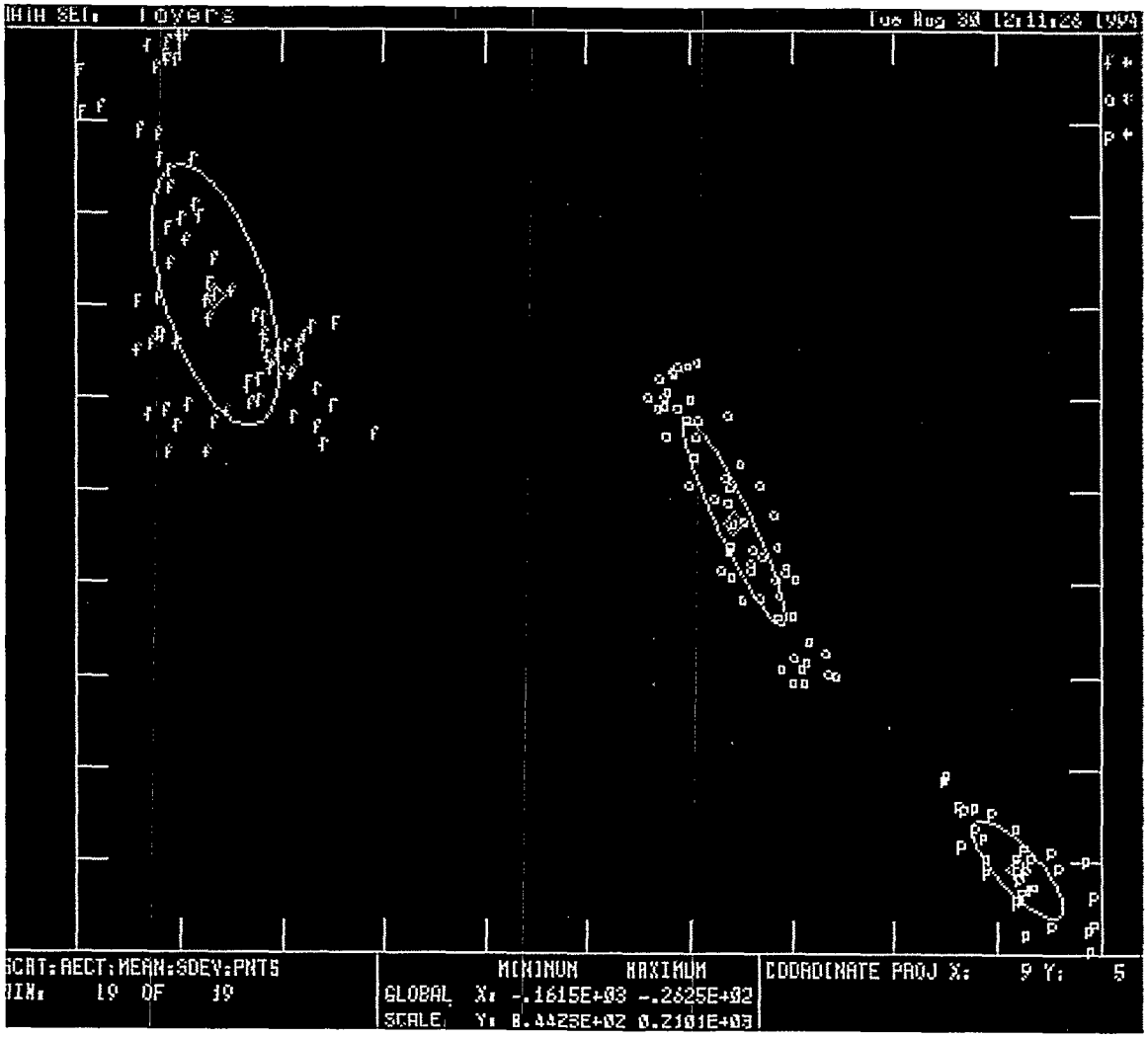


Fig. 8. OLPARS two-dimensional projection of excess red and excess blue means for all blobs. The symbols f, a, p represent field, active, and poly areas.

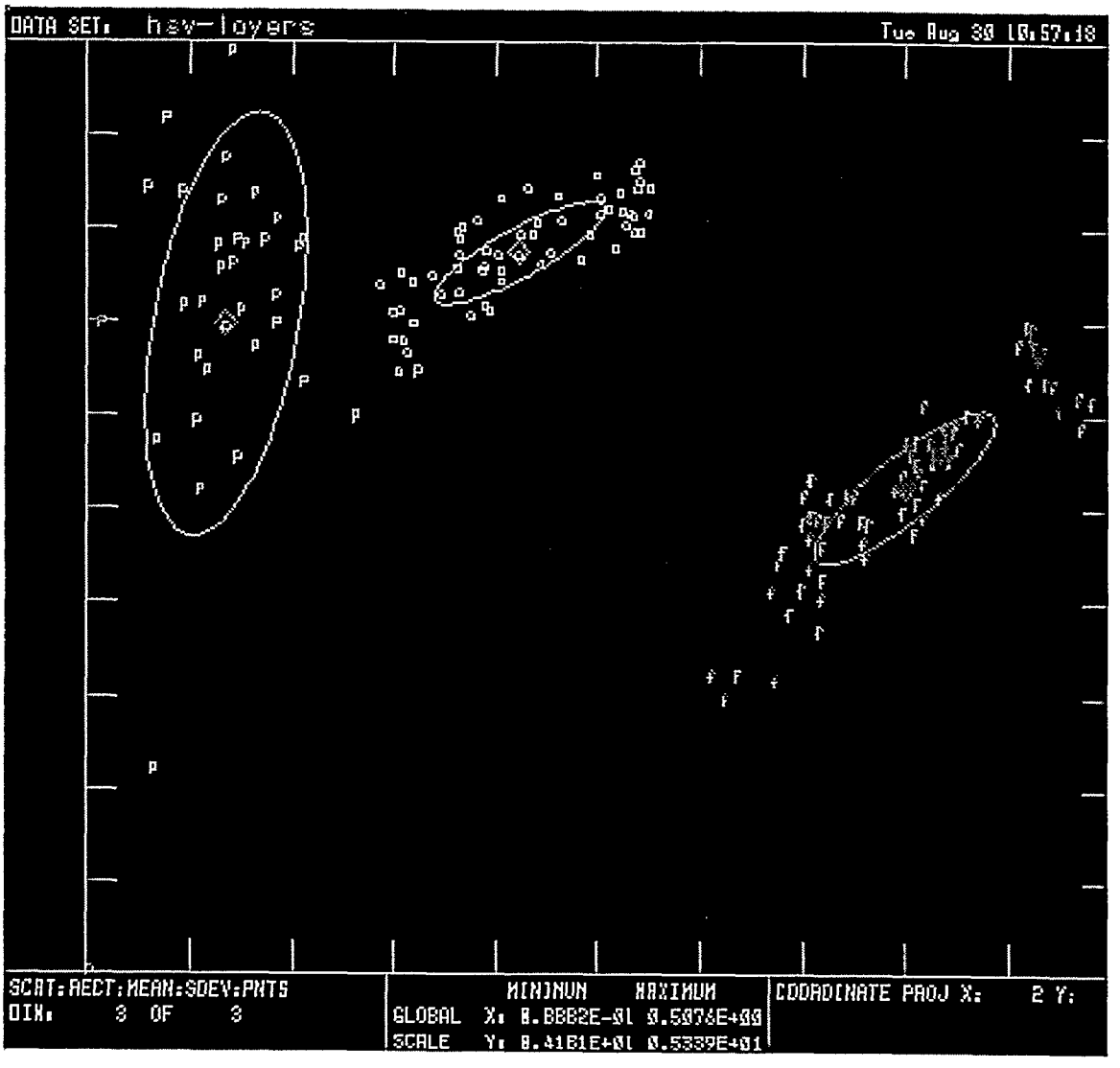


Fig. 9. OLPARS coordinate projection of saturation (x-axis) and hue (y-axis).

3.4 Accurate segmentation based on fusion of pixel and region data

This section describes our preliminary efforts in segmenting and recognizing the background in high-resolution color images of wafers using a combined pixel and region approach. The backgrounds of the images under consideration consist of three layers, namely poly, active area, and field. Furthermore, we define a fourth region (not a physical layer) which represents the boundary zone between the aforementioned layers as shown in Figure 4. In what follows, we describe a specific technique for segmentation and recognition of the field layer.

3.4.1 Segmentation

As evident from the visual inspection of the images, color plays an important role as an inter-layer discriminator. More precisely, it can be observed that the field layer is, almost always, distinguishable from all the others solely based on color related features as has been seen in sections 3.2 and 3.3. This observation is the basis for the following field layer segmentation and recognition technique. This method is described below in terms of its three major components:

Color-based data generation: This part of the proposed segmentation technique relies heavily on the fact that in almost all of the images under consideration, the field layer, unlike the others, lacks one of the three components of the RGB color coordinates; usually either red or blue. This characteristic, described in section 3.1 as the excess color, is exploited by computing the following measurement:

$$g(x,y) = n\, f_R(x,y) - \{f_G(x,y) + f_B(x,y)\}$$

where f_R, f_G, and f_B are the RGB components of the input reference image, respectively; n is an integer (usually equal to 2); and g(x,y) is the output image. It is noted that because the color component that is lacking in the field layer (but not in the others) is weighted heavier than the other two, the resulting image, g(x,y), will possess a histogram that is inherently bimodal. Hence, an optimal thresholding technique [1] may be employed to automatically compute a threshold value, T, such that:

$$b_1(x,y) = \begin{cases} 1 & \text{if } g(x,y) \le T \text{; implying that } g(x,y) \in \text{Field Layer} \\ 0 & \text{if } g(x,y) > T \text{; implying that } g(x,y) \notin \text{Field Layer} \end{cases}$$

Due to the expected variations in color, especially near the layer boundaries, the selected regions in $b_1(x,y)$ may exhibit one or all of the following undesirable characteristics (see examples in Figure 10):

- Too small, *i.e.*, the region size is below the minimum defect size.
- Contain holes and small protrusions.
- Fragmented, especially evident in long and narrow field-layer regions

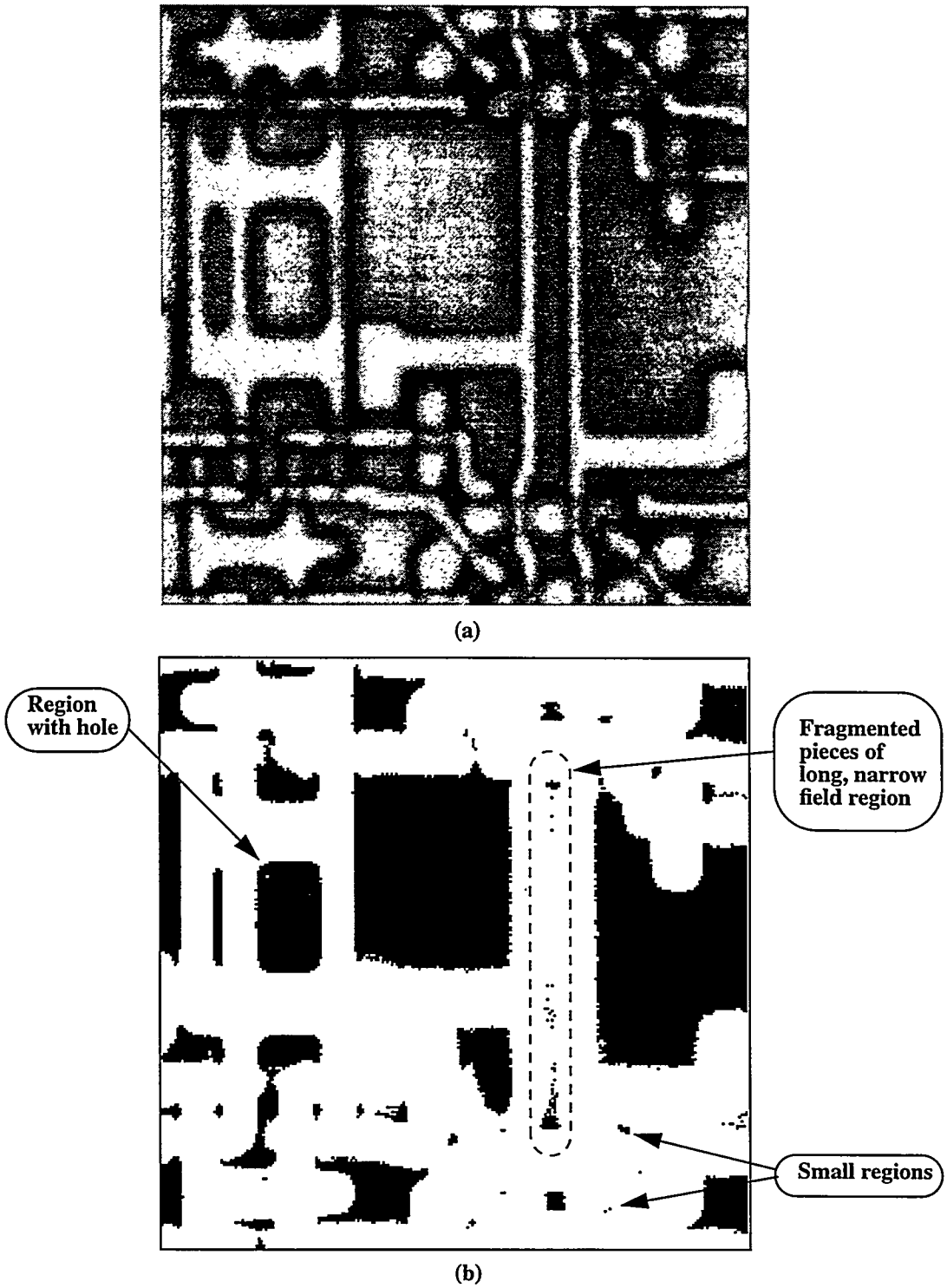


Fig. 10. Examples of undesirable information after the excess-color threshold operation. The original image is in (a). The image in (b) shows small regions, regions with holes and protrusions, and fragmented regions that must be handled in subsequent processing steps.

The first two problems are readily treated by utilizing two nonlinear filters; first, a size-sensitive filter that passes those regions in $b_1(x,y)$ whose size in pixels $\in (10, \infty)$, and the second, a filter which implements the morphological closing operation. The remaining components of the proposed segmentation and recognition technique are designed to handle the most challenging problem of fragmented regions.

Edge-based data generation: The utility of this component is to generate connected field-layer regions by performing an edge-based segmentation of the overall background using the intensity component grey-level reference image as the input (color to gray scale conversion method is the same as described in previous reports). The reason for considering the edge-based information is that the fourth region in the wafer images, i.e., the boundaries between the three physical layers, seems to always constitute a strong discontinuity between the field and other two layers.

The connected field-layer regions, together with regions corresponding to the other layers, are generated in this component by thresholding the output of the DOG operator (see Section 3.3) with a threshold value of zero. An example result is shown in Figure 11.

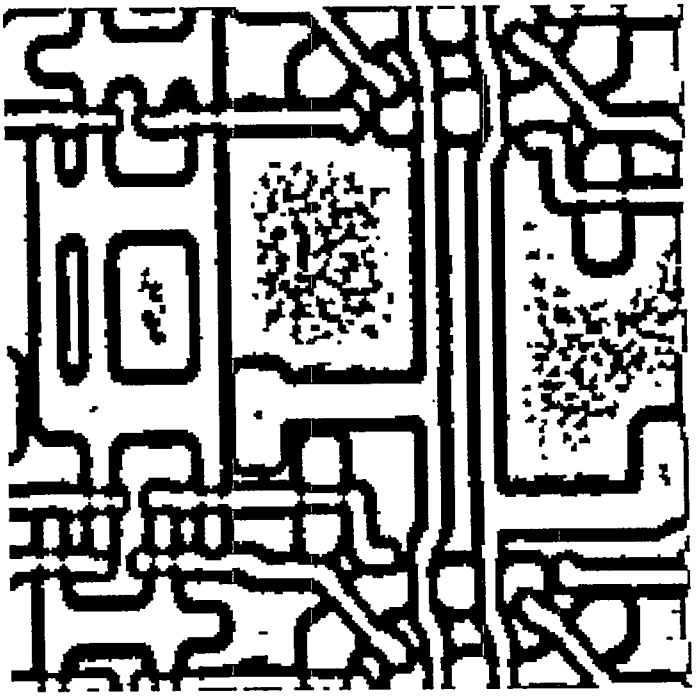


Fig. 11. Segmented result after applying the difference of Gaussians filter to a wafer image

The recognition of the field-layer regions among the others is accomplished through data fusion which is described next.

Data/region fusion: To generate an accurate segmentation of the field-layer regions, this component of the proposed technique fuses the regions obtained from the previous two components in accordance with the diagram shown in Figure 12.

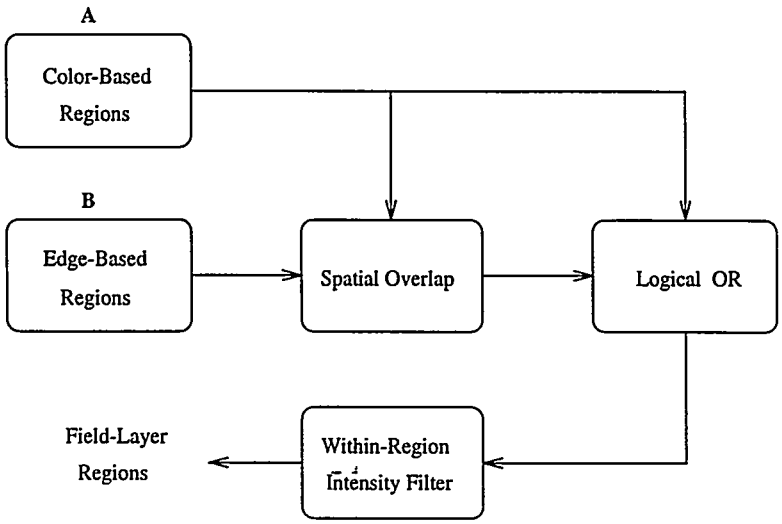


Fig. 12. Region fusion strategy that combines edge-based segmentation result with color threshold segmentation result

The functionality of these blocks are described as follows. The images in Figure 13 show a complete start-to-finish example of this procedure.

- Spatial Overlap: For every region from block B, if its spatial boundary overlaps that of at least one region from block A, then it is passed to the output; else it is rejected. This block reconstructs the fragmented regions which appear in the output of the first component.
- Logical OR: The regions obtained from the above block are ORed with that of block A, to ensure that the field-layer regions are solid, and as large as possible without overstepping their boundaries.
- Within-Region Intensity Filter: This filter performs a final check on the output regions, and ensures that they, in fact, correspond to the field layer. This is accomplished by reevaluating the color content within each region to assure its compliance with the criterion which was established in the first component.

The described method of field-layer region segmentation and recognition has performed favorably when confronted with a variety of high-resolution color images of semiconductor wafers. Efforts to quantify its performance in terms of success, false positive, and false negative rates are currently underway.

3.4.2 Utility of Field Layer Segmentation in Defect Classification

In this section, we outline a procedure for automatic defect classification of the "field particle" flaw. Here, we make the reasonable assumptions that the defect is contained wholly within the field layer, and that it is visually distinguishable from its surroundings. It is

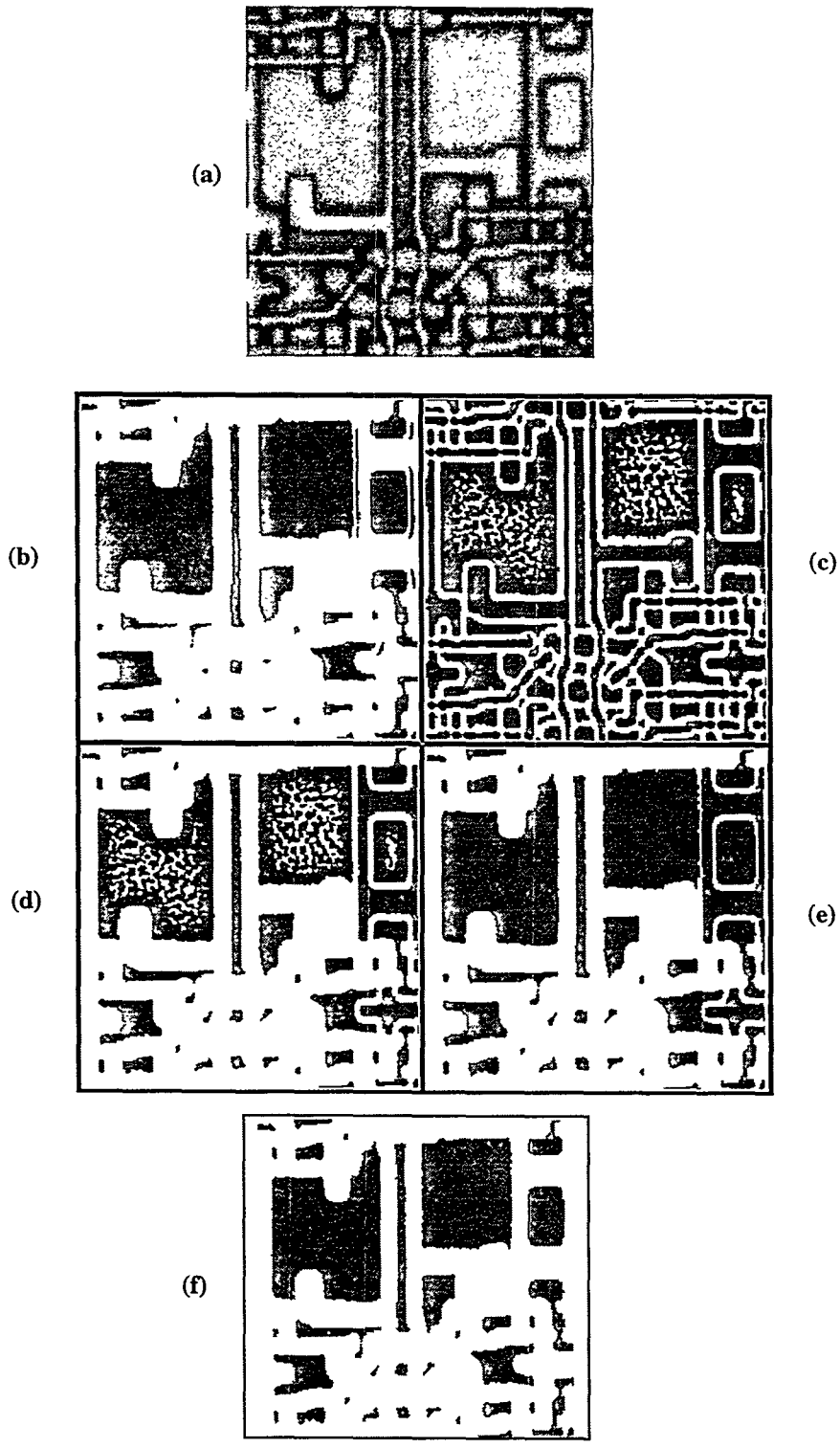


Fig. 13. Example image sequence for region fusion followed by intensity filtering. The original image is in (a). Images (b) and (c) are the color segmented and edge-segmented region images, respectively. The spatial overlap of (b) and (c) is shown in (d). The result of the OR operation between (d) and (b) is displayed in (e). By comparing the regions in (e) with the original image (a) and filtering areas based on the region color content, the final segmented field image was generated and is shown in (f).

worth noting that this approach deviates, ever so slightly, from the proposed architecture of the Automatic Defect Classification System in that it attempts to analyze the background of the defect as well as the reference image. The procedure includes three simple steps:

1. The defect and the corresponding reference image (after registration) are both subjected to the segmentation technique which was presented in the previous section.
2. Shape features, such as compactness, orientation of the major axis, no of holes, etc., are measured from every region of the two segmented images. It is recommended that the selected features be size-invariant.
3. The derived features from one field-layer region in the reference image are compared with features derived from the corresponding field-layer region in the defect image (correspondence is established through spatial information, e.g., centroid).

A number of conclusions can be reached based on this simple procedure. For instance, discrepancy in the values of the "number of holes" feature between corresponding regions is a necessary condition for the existence of a wholly contained "field particle" or some types of "missing pattern" flaws. Figures 14 and 15 show examples of how a "field particle" and a "missing pattern" defect cause changes in the structure of the field regions. Also, if there are only insignificant deviations among shape feature values in corresponding regions, then one could deduce that the defect under consideration is an active area and/or poly type flaw.

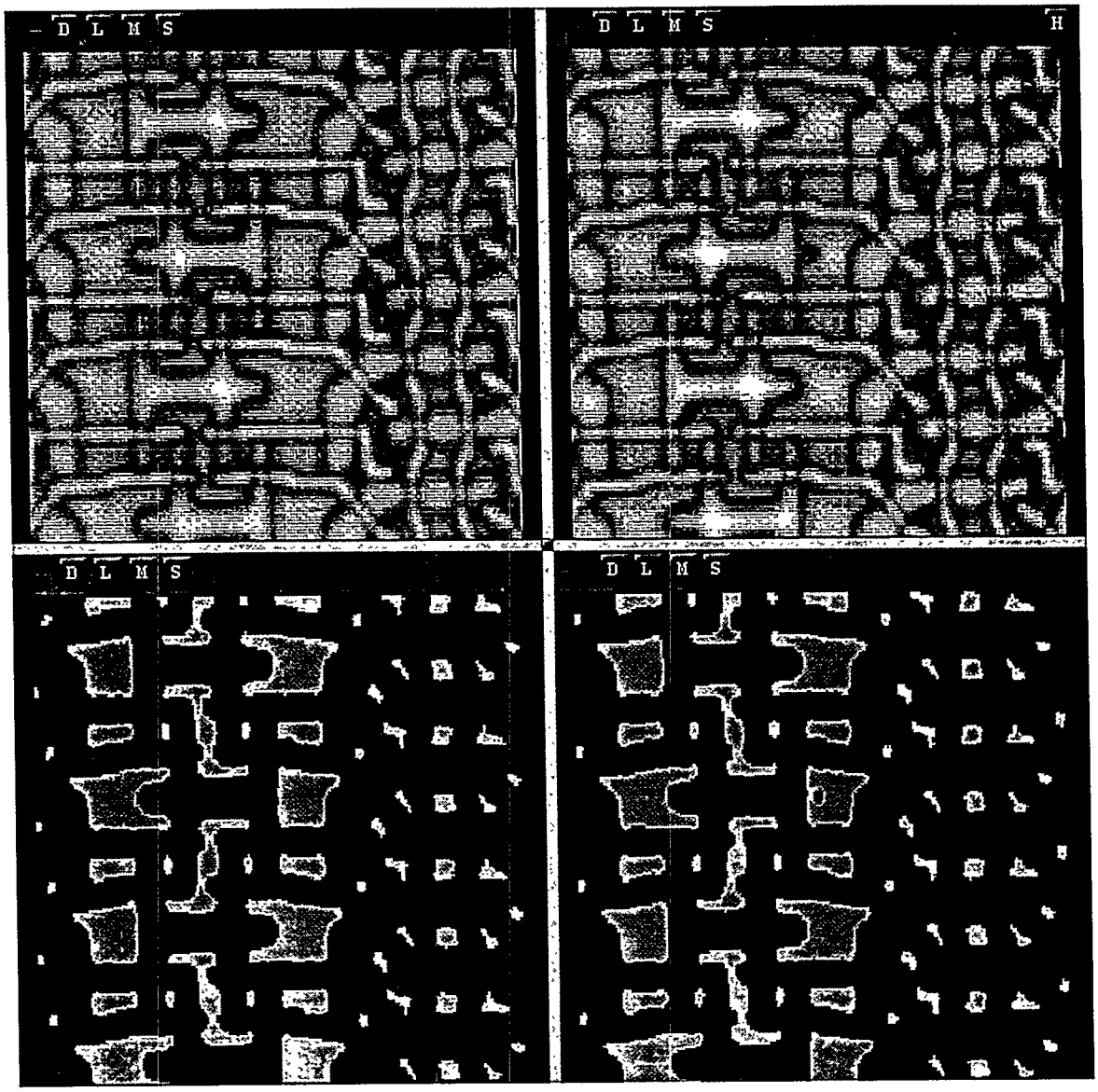


Fig. 14. Example of region-based approach on a particle defect. The original image and the segmented field regions are shown on the left. The defect image and its corresponding field regions are shown on the right. Note that a defect is detected by observing the number of holes in each region.

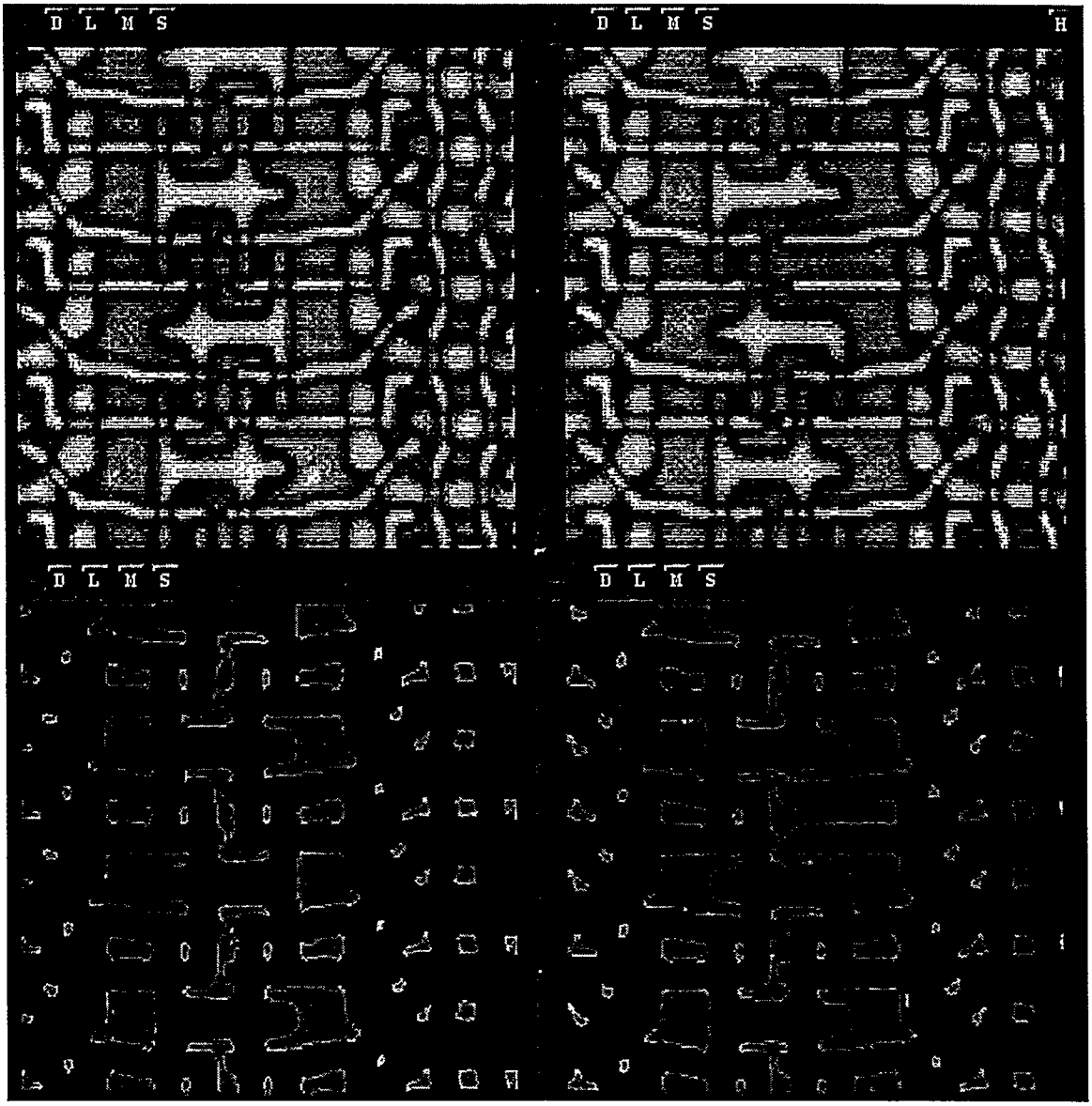


Fig. 16. Example of region-based approach on a missing pattern defect. The original image and the segmented field regions are shown on the left. The defect image and its corresponding field regions are shown on the right. Note that a defect is detected by simply comparing the number of regions in each field image.

4.0 Technology Transfer

4.1 Information Exchange

Two members of the ORNL team spent three man-weeks on-site at KLA Instruments in San Jose, California. The purpose of the visit was to have an extended ORNL presence at KLA headquarters to promote the exchange of information between the technical teams. Communication via telephone, electronic mail, and technical report updates is an effective way to stay in touch with what each team is contributing, but working together under the same roof is invaluable in creating an atmosphere where ideas can be freely exchanged.

Several technical discussions were held to make KLA staff completely aware of the approaches that the ORNL team is pursuing. The combined approach of defect classification through defect feature analysis and defect contextual analysis was discussed in detail.

During the visit KLA's new strategy for integrating ADC into their on-line inspection system (KLA 21XX) was described to the ORNL team. In order to accomplish the ADC on-line, a data reduction step is necessary to eliminate a large portion of the initially detected defects. KLA provided the ORNL staff with information on their plans to do global defect map clustering and subsequent statistical sampling in order to reduce the original defect set down to a manageable size.

4.2 Data Gathering

Each of the two ORNL team members spent time gathering wafer images from the DEC wafer set. This allowed them to become more familiar with the operation of the operation of the 26XX data review station. The ORNL team has a better understanding of the difficulties in acquiring quality defect data. Acquiring a defect image and a reference image at several different focus offsets is a challenging and time-consuming task. Several multi-focus images were gathered from the DEC wafers consisting of several defect types listed below in Table 2..

Table 2: Number of collected defects from DEC wafers.

Defect type	Number collected
missing pattern	9
micro-scoring	7
active area particle	14
poly particle	1
bridging particle	2
residue	2

4.3 Tools and Software

ORNL has benefitted from the use of several in-house tools that perform various image processing, feature extraction, pattern recognition, and classification/clustering algorithms. Two tools that ORNL has used extensively on this CRADA are Khoros and KB Vision. Both of these software packages were introduced to the KLA technical team during the visit.

Khoros is a free-ware general signal and image data processing package written by members of the electrical engineering department at the University of New Mexico. Algorithms are implemented in Khoros by placing function "glyphs" on a workspace palette and connecting the inputs and outputs of the glyphs to form a specific data processing path. ORNL uses Khoros to implement many of the feature extraction routines described in previous reports. Khoros, along with all of the ORNL-created workspaces, was delivered and installed at the KLA site. KLA staff was shown how to run Khoros, how to load ORNL workspaces and modify them, and how to add their own functions to the Khoros menus.

KB Vision provides a multitude of high-level image analysis and image recognition algorithms. Algorithms can also be implemented in the form of data-flow workspaces similar to Khoros. Much of the wafer segmentation and background structure identification research has been performed using KB Vision. ORNL staff installed a temporary KB Vision license at KLA headquarters and showed some of the ORNL workspaces implemented to perform background segmentation.

5.0 References

1. Gonzales, R.C. and Wintz, P., *Digital Image Processing,* 2nd edition, p. 81., Addison-Wesley Publishing Co., Reading, MA, 1987

2. Marr, D. and Hildreth. E., "Theory of Edge Detection", *Proc. Royal Society of London*, B.207:187-217, 1980.

Appendix E

ORNL's Off-Focus Feature Extraction

October 1994

~~(Protected CRADA Information~~)

ORNL'S OFF-FOCUS IMAGE FEATURE EXTRACTION

1.0 Introduction

ORNL's effort on feature extraction from off-focus images is progressing. We are currently working with the latest set of defect images delivered by KLA. The current defect set contains bright field images of defects and reference images taken at three different focus offsets (in-focus, above-the-plane, below-the-plane). One of our main tasks is to look at a defect at various focus offsets to determine if any features can be extracted that would allow automatic defect classification (ADC). Several software routines have been written to perform registration and defect mask generation on off-focus image pairs.

2.0 Current Image Set

The current set of defect images represents approximately 260 different defects. Of the set, seven of the classes have a significant number of defect samples (> 10). The "missing pattern" class is the only one of this seven that has examples of non-particle type defects. One problem with this set is that only two of the 26 defect samples have off-focus images available for study.

The other six classes are particles classed which seem to be based on their location on the wafer, not their material composition. Particle defects make up approximately 75 percent of the data set. This defect set should be a representative sample for looking at off-focus features of particles, but more samples of other non-particle defects will be necessary to thoroughly analyze their specific features.

3.0 Defect Extraction Software

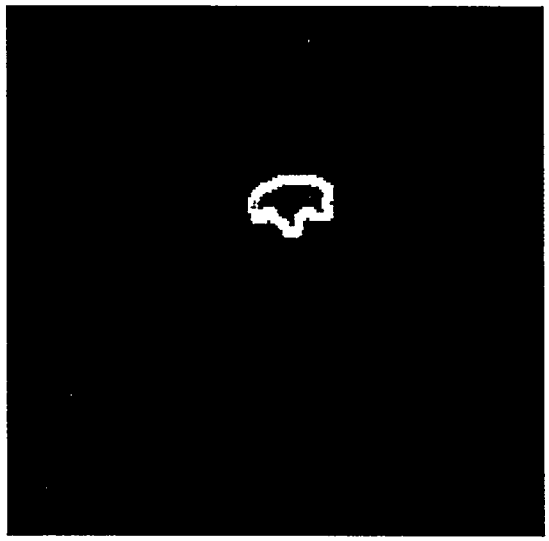
3.1 Khoros Defect Registration Program

We have been using KLA's defect extraction software (menureg) to do defect segmentation on in-focus images only. One issue that KLA has requested we study is performing subpixel image registration and warping on off-focus images because menureg does not have this capability. We have developed an algorithm in the Khoros software environment that uses the subpixel registration found by the menureg program. The off-focus defect images are warped (translated by subpixel distance) and the difference images are then generated in Khoros.

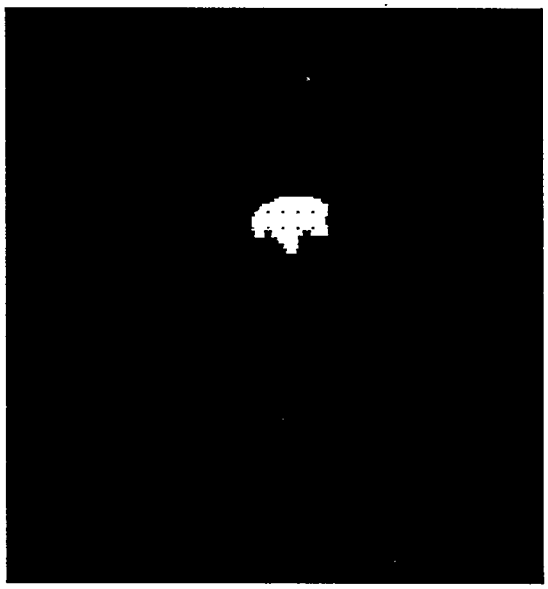
We have added one more step in the defect segmentation process to suppress spurious defects caused by "ghosting" where edges in images do not line up exactly even after subpixel registration. This occurs commonly where there are sharp edge transitions in the original reference image. We apply an edge suppression mask to the difference image before thresholding to remove this noise. The application of this filter was nessary due to the wide range fixed thresholds encountered in the defect segmentation.

3.2 Mask Generation

Another software routine was written which generates two different types of defect masks. On mask is created that covers the entire area of the defect, and the other mask highlights only the edges of the defect. Examples of each type of mask are shown below:



Edge Mask



Full Defect Mask

3.3 Feature Extraction for Off-Focus Images

The algorithm described in Section 3.2 generates a defect mask (binary) image that is used to mask out all areas in the defective image except for the defect itself. Software was written to extract several features of the off-focus images:

1. root mean square (rms) of defect intensity
2. area of defect
3. contrast of defect area
4. fractal dimension (roughness) of defect
5. rms in neighborhood of edge
6. length of defect edge
7. contrast in neighborhood of defect edge
8. fractal dimension in neighborhood of defect edge

Attached Tables 1 and 2 show the results of this feature extraction software on five "large particle" defects and two examples of "missing pattern" defects. We have not examined enough samples of defect types to make conclusive decisions about the usefulness of the off-focus information. We hope that the three-dimensional properties of particles will show up in off-focus images through the feature set described above.

Appendix F

Semiconductor Yield Improvement Technical Progress Report
November 1994

~~(Protected CRADA Information~~)

ORNL

SEMICONDUCTOR YIELD IMPROVEMENT TECHNICAL PROGRESS REPORT

November, 1994

S. S. Gleason
M. A. Hunt
H. Sari-Sarraf

ORNL

SEMICONDUCTOR YIELD IMPROVEMENT TECHNICAL PROGRESS REPORT

S. S. Gleason M. A. Hunt H. Sari-Sarraf
Instrumentation and Controls Division

November 1994

Prepared for the
ORNL/KLA Automatic Defect Classification CRADA
under CRADA No. ORNL92-0140
between the U.S. Department of Energy, Martin Marietta Energy Systems and
KLA Instruments Corporation

Prepared by
Instrumentation and Controls Division
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6011
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

1.0 Executive Summary

This is a technical progress report written by the ORNL staff for KLA Instruments Corporation as part of the Automatic Defect Classification (ADC) Cooperative Research and Development Agreement (CRADA). The content of the report describes technical approaches applied to ADC through the month of November, 1994

The set of images that was collected by ORNL staff at KLA headquarters in July was analyzed. The images consisted of approximately 40 defects, half of which are particle and half non-particle (missing pattern, residue, etc.).

An additional feature was added to radial basis function (RBF) classifier developed in the Spring. In the original RBF classifier implementation, the number of RBF centers (Gaussian centers) was arbitrarily selected to be the same value as the number of feature vectors in the class of interest. An orthogonal least-squares technique was implemented that calculates the contribution of each center in the classifier to the classifier's output. This allows one to rank the contributions of each center, and to subsequently choose intelligently some number of representative centers from the entire set of feature vectors. This software was delivered to KLA for further testing and evaluation.

The work on wafer background segmentation was extended to include some non-color based feature measurements. These attributes of wafer structures (active area, field, poly, etc.) were extracted and ranked according to their discriminatory power. Additional work on a token-level connectivity analysis is also reported. This work addresses the problem of identifying contiguous wafer structures that have been broken up by other overlapping structures in the digitized image.

2.0 ORNL Image Set Evaluation

During the three-week visit by ORNL staff member to KLA headquarters in San Jose, CA, a set of 35 images was gathered from the DEC wafer set in an attempt to gather a good sample of non-particle defects. A distribution of the number and types of all defects gathered is given below in Table I. This data was gathered on a 2108 wafer inspection workstation..

Table 1: Number of collected defects from DEC wafers.

Defect type	Number collected
missing pattern	9
micro-scoring	7
active area particle	14
poly particle	1
bridging particle	2
residue	2

The images acquired at KLA had two problems which prevented them from being very useful in the ADC technical development. Focus changes that occurred when switching from the reference to defect image resulted in sharpness differences between the images. The auto-focus feature of the 2108 did not consistently adjust the focal plane to exist on the same layer of material in the reference and defect images. This problem carried over into the multiple-focus images that were gathered. For each defect image captured, three different focus-offsets were applied. There is not a consistent way to move the focal plane of the 2108 optics by a specific distance, so ORNL-staff gathering the images had to manually adjust the focus above and below the plane of the wafer. This resulted in inconsistent focus offsets between reference and defect images.

Another difficulty encountered in gathering data on the 2108 was discovered when the image set was analyzed. In order to capture a reference image on the 2108, the machine will automatically position itself on the same spot on the neighboring die. The accuracy of this automatic repositioning was not accurate enough to consistently perform a digital registration of the two images, The translation offset between reference and defect images was typically greater than 30 pixels. This inaccurate positioning mechanism may have been a problem particular to the workstation used for this data collection.

In summary, data collected in the future should be acquired using the 2608 review station that has a programmable focus offset specification as well as a consistently accurate positioning mechanism.

3.0 Orthogonal Least-Squares Approach to RBF Center Selection

Included below is some information from the "Radial Basis Function Routine Descriptions" technical document delivered with the first version of the RBF software. For more information regarding details on the amplitude and width parameter selection for the Gaussian basis functions, see the aforementioned technical document. This section focuses on the new approach to Gaussian center parameter selection.

3.1 Gaussian Specification

The function rbf_train.c builds one RBF classifier for each class based on a training set of feature vectors. The classifier is constructed by summing a collection of Gaussians (radial basis functions) in the form

$$R_i(\mathbf{x}) = e^{-\left(\frac{\|\mathbf{x}-\mathbf{c}_i\|}{w}\right)^2} \tag{1}$$

where i specifies one Gaussian function with center $\boldsymbol{c}_i$. To form the RBF classifier, a group of Gaussians must be summed together evaluated at the feature vector, $\boldsymbol{x}$.

$$\mathrm{RBF}(\mathbf{x}) = \sum_{i=1}^{n} A_i R_i(\mathbf{x}) \tag{2}$$

The unknowns that must be selected to completely specify this family of Gaussians are the centers ($\boldsymbol{c}_i$), the width (w), and the amplitudes (A_i). Equation (2) can be rewritten as

$$\mathbf{Ea} = \mathbf{y} \tag{3}$$

where $\boldsymbol{E}$ is an m by n matrix of the form

$$\mathbf{E} = \begin{bmatrix} R_1(\mathbf{x}_1) & \cdots & R_n(\mathbf{x}_1) \\ \cdots & \cdots & \cdots \\ R_1(\mathbf{x}_m) & \cdots & R_n(\mathbf{x}_m) \end{bmatrix}, \tag{4}$$

$\boldsymbol{a}$ is an n by 1 coefficient vector of the form

$$\mathbf{a} = \begin{bmatrix} A_1 \\ \cdots \\ A_n \end{bmatrix}, \tag{5}$$

and $\boldsymbol{y}$ is an n by 1 output vector of the form

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix}. \tag{6}$$

The y_i are 1 if the feature vector is in the current class of interest and they are 0 if the current training vector is not in the class. In these matrices, n is the number of centers (or Gaussians in the RBF), and m is the number of feature vectors used in the training set for this particular RBF classifier.

The centers of the Gaussians in the original version of rbf_train.c were selected as the feature vectors of the training set. This is a common practice when using radial basis functions for data interpolation when a function value is known only at specific coordinates. One obvious problem is that if your training set is large, there are more computations in the RBF evaluation and the determination of the RBF coefficients, A_i. The amount of time to find all of the coefficients can become significant. A better method may be to choose fewer, yet more representative centers to start with. In [1] a method of choosing RBF centers is given that attempts to optimize the number and location of RBF centers automatically.

This orthogonal least-square (OLS) approach is a technique which chooses the centers from the original feature vector data set intelligently. In equation (2), **y** is the desired output, **A** is the amplitude vector, and the columns of matrix **E** are the regressors. Also, each column of **E** corresponds to one particular Gaussian with a unique center. In general the contributions of each regressor to the output, **y**, are correlated, so the individual contributions of each regressor cannot be measured. By performing an orthogonalization procedure such as Gramm-Schmidt [2] on the columns of **E**, the contribution of each column to the output value can be measured. Each regressor (or Gaussian center) can then be ranked according to its contribution to the output value. This ranking then allows one to select a subset of centers for the classifier construction.

There is a built in a ill-conditioned matrix check, so linearly dependent feature vectors will be left out of consideration. The terminating condition, ρ, is either defined in rbf.h (#define RHO 0.003) or it can be defined on the command line of the rbf_train executable with the -r switch. As the value of ρ increases, the number of centers selected decreases. The paper in reference [1] describes a way to automatically determine ρ during the training, but this feature is not currently implemented.

3.2 Results of OLS Decomposition

The data sets that this algorithm was tested with were not large enough to allow one to quantify how well the center selection procedure works. The performance was inconsistent from one class to the next In one particular class the centers were reduced from 24 to 8 and the accuracy of the classifier fell from 98% to 95%. In another class the number of centers was reduced from 51 to 40 and the accuracy improved from 79% to 81%. Finally,

in another class, the centers were reduced from 64 to 53 and the accuracy of the classifier fell from 82% to 65%.

The data used for testing was the original low-res feature set sent by KLA last Spring. The feature vectors are 51 features long and the number of vectors in each class ranges from 24 to 64. That is really not enough samples per class given the long length of the feature vector. The final version of the algorithm was delivered to KLA at the beginning of October for further testing and evaluation.

4.0 Feature-Based Background Segmentation

The previous technical report for August, 1994 focused on using color information in the wafer image to segment the wafer background structure into its various components. For example, it was illustrated that a particular wafer could be segmented into active area, poly lines, and field area through a multi-step process consisting of a color transformation, edge detection, blob analysis, intensity average per blob, and finally, a clustering approach to component classification. This section describes another approach to background segmentation based on image features other than color values. Relying entirely on color for image segmentation is risky in that the colors imaged from the wafer are due more to optical effects of the light interacting with very thin layers of material than they are to selective reflection of specific frequencies from the wafer surface.

Connected region images of the excess-red, excess-green, and excess-blue versions of the original were generated using the same techniques described in the August, 1994 technical report. In addition to extracting features of average intensity from each region as was done previously, all of the available shape and intensity features provided by KB Vision were extracted. This comprehensive set of features was output to a text file and formatted to be compatible with OLPARS.

Table 2: Background Structure Feature List

Feature Name	Numeric
Bounding-rectangle-fill	1
Bounding-rectangle-to-perimeter	2
Compactness	3
Elongation	4
Excess-Blue mean	5
Excess-Blue std. dev.	6
Excess-Green mean	7
Excess-Green std. dev.	8
Excess-Red mean	9
Excess-Red std. dev.	10
Height	11
Log of Height-to-width	12
Major axis	13
Number of Blobs	14
Number of Holes	15
Perimeter	16
Pixel Count	17
Token index	18
Width	19

OLPARS was used to perform a feature ranking to see which features most discriminated between layers. The following lists are the feature rankings using the Fisher discriminant and the probability-of-confusion ranks. The "Class" column identifies the class which is most readily separated from all other classes using the current feature. The "Pair" column shows the two classes which are most separated by the given feature. This column is given because both the Fisher and probability-of-confusion techniques use a pair-wise class comparison method to extract the feature rank.

Fisher Discriminant rank:

Rank	Feature	Value	Class	Pair
1	9	5.4836	f	fp
2	5	1.0188	p	fp
3	10	0.4662	p	fp
4	3	0.4145	p	ap
5	4	0.3116	p	ap
6	8	0.2427	p	fp
7	6	0.2097	p	fp
8	7	0.1859	a	ap
9	16	0.1513	p	fp
10	1	0.1312	p	ap
11	14	0.0920	p	fp
12	2	0.0735	p	ap
13	15	0.0357	p	fp
14	12	0.0354	p	ap
15	13	0.0211	a	fa
16	11	0.0191	p	fp

f - field, p - poly, a - active

The Excess-Red mean feature (9) did the best job between field and poly and separating the field from all other classes. The Excess-Blue mean (5) was best for separating the poly

from all others. The Compactness (3) and Elongation (4) features were best at separating the active from the poly.

Probability of Confusion:

Rank	Feature	Value	Class	Pair
1	9	0.00	f	fa
2	5	0.6310	p	fp
3	4	1.3343	p	ap
4	10	1.4216	p	fp
5	14	1.4732	p	ap
6	7	1.5427	a	ap
7	3	1.6062	p	ap
8	16	1.6667	p	ap
9	15	1.8313	p	fp
10	13	1.9206	a	ap
11	8	2.0030	p	fp
12	11	2.1161	a	ap
13	6	2.1776	p	fp
14	1	2.1915	p	ap
15	12	2.6230	p	fp
16	2	2.6577	p	ap

Notice again the similar ranking of the top five or six features. One notable difference is that elongation (4) ranked higher than compactness (3) for discriminating poly using the probability-of-confusion metric.

5.0 Background Segmentation

This section describes the continuation of our preliminary efforts in segmenting and recognizing the background in high-resolution color images of wafers. The backgrounds of the images under consideration consist of three layers, namely poly, active area, and field. Furthermore, we have, in the past, defined a fourth region (not a physical layer) which represents the boundaries between the aforementioned layers. As described and demonstrated in our previous report, an accurate segmentation and recognition of the field and poly layers can be achieved through the extraction and fusion of color-based and/or edge-based information. Here, we shall propose a different approach for the segmentation and recognition of the active area layer.

5.1 Accurate Segmentation of Active Area Layer

A major challenge in accomplishing the task of active area segmentation and recognition is the fact that, due to the nature of the manufacturing and image acquisition processes, every active area region appears to be fragmented in the image space. This poses a serious problem for approaches that utilize edge-based segmentation methods, because each active area region is effectively represented as a collage of noncontiguous fragments, see Figures 1 (a) and (b). Furthermore, some of these fragments can be relatively small in size, hence, utilizing features such as color or shape may be neither meaningful nor discriminating.

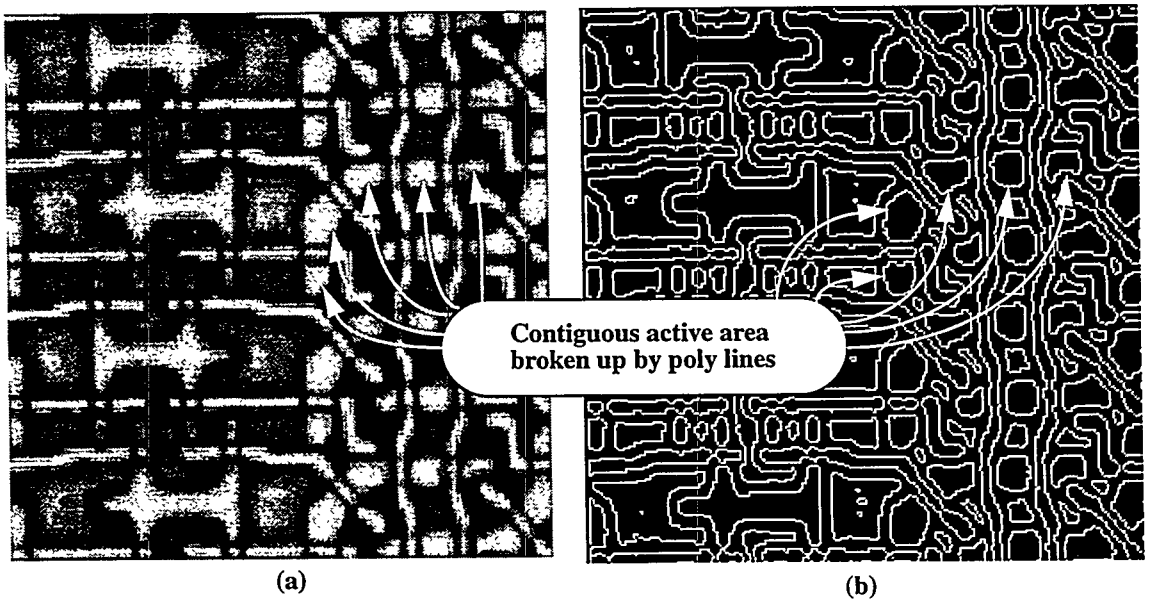


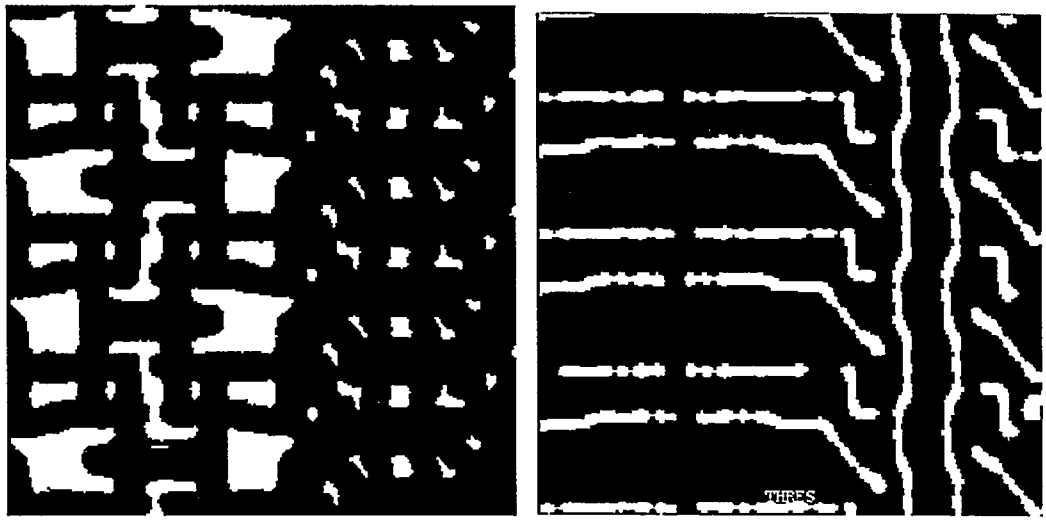
Fig. 1. (a) The original grayscale image and (b) edge-segmented image

In this section, we present an approach for the segmentation and recognition of the active area regions that does not rely on color or shape information alone. Rather, this task is accomplished by utilizing some knowledge of the problem domain, and by considering

structural relationships that exist between different regions in the images under consideration. Before describing our technique, it should be mentioned that the implementation and the testing of what is to follow are at their preliminary stages, and thus, there are details that are either not reported, or are yet to be worked out.

Close inspection of the images under consideration reveals that the fragmentation of the active area regions is due to the intersection of these regions by poly lines. As stated above, in our approach this problem-domain knowledge is used to establish structural relationships between the active area regions and the poly lines. The proposed approach can be described in terms of the following steps:

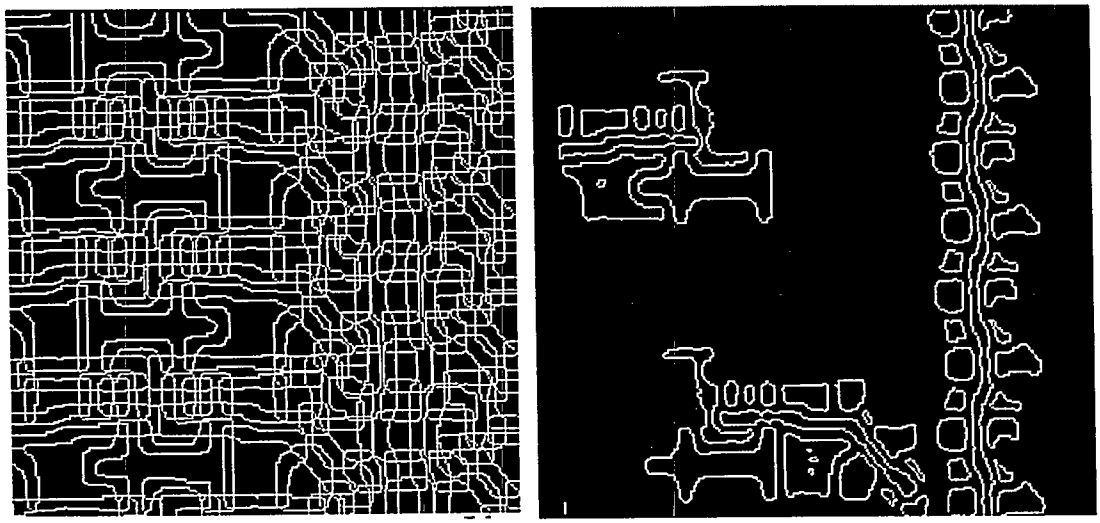
1. We start by performing an edge-based segmentation of the overall background by convolving the grey-level reference image with the Laplacian of the Gaussian (LoG) filter, and extracting the zero-crossings of the obtained image as shown in Figure 1 (b). Furthermore, we identify the poly and field regions by the applications of the appropriate color-based segmentation algorithms, as described in the previous report (Figure 2).



(a) (b)

Fig. 2. Field area (a) and poly-line (b) segmented images.

2. For every poly line in the given image,
   a. its neighboring regions are determined. This is accomplished by dilating all of the regions by a specified amount (Figure 3 (a)), and selecting those regions whose spatial extents overlap (Figure 3 (b)).



(a) (b)

Fig. 3. Dilated regions (a) and spatially overlapped regions (b).

b. its skeletal representation is produced. This is achieved by thresholding the output from the LoG filter, and subsequently, performing thinning or morphological skeletonization on the resultant binary image, Figure 4.

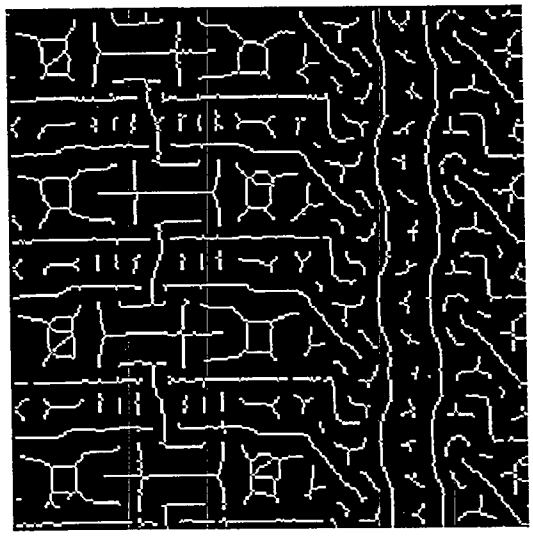


Fig. 4. Skeletal representation.

c. a straight line is fitted to the first n points of its skeletal representation. If the error of fit is not less than or equal to some ε, further processing ceases, and the next n points are considered. For small errors of fit, the perpendicular bisector of the fitted line is obtained, and the regions that it intersects are recorded. At this point, without the ben-

efit of actual implementation and testing, we conjecture that two intersected regions that are unlabeled (they are neither field nor poly), are the noncontiguous fragments of the same active area region. This process continues for the next n points, until all of the points on the skeletal representation of the poly line have been considered.

The outlined procedure is a high-level connnectivity analysis that operates on tokens (collections of pixels) rather than individual pixels. Given the layered structure of semiconductors, the accurate identification of contiguous regions in the midst of many other overlapping structures becomes a critical and challenging task. In those defect classification schemes that require automatic recognition of the particular layer or structure that a defect occurs on, this high-level token connnectivity approach has a potential for success. Further development of this idea on multiple wafer types is recommended. Unfortunately, there are not enough resources to continue this reasearch while the construction of the ADC prototype tool (Section 6.0) is in progress.

6.0 ADC Tool Development

An ADC prototype tool is under development as the final task for the CRADA. The ADC tool will have an X-windows, Motif, and UNIX-based graphical user interface (GUI). The code is being developed using C++.

6.1 Graphical User Interface

All of the major steps needed for this implementation of ADC will be included on the main window of the tool's GUI. These major elements include (1) Data Input/Output, (2) Defect Detection, (3) Feature Extraction, and (4) Classification. All four steps can be executed at this level by an operator, but the tool will also have detailed configuration parameter windows which can be accessed by the operator if this is deemed necessary. For example, the defect detection configuration window has various parameters to be defined such as template window location and size, search window location and size, automatic or manual tie-point selection. The data input/output configuration window will allow the operator to read in individual images or image lists of defect and reference image files. Once these details are configured by the operator, all of them can be written to a configuration file. This file will be read at start-up to configure the ADC tool for the same setup as specified in the saved configuration file.

6.2 C++ Code Definitions

The class structure for the storage and processing of images has been defined in a C++ class hierarchy. Figure 5 gives a high-level description of the image class. A similar class structure is being considered for feature vector data and member functions, but there are not enough details to report on at this time.

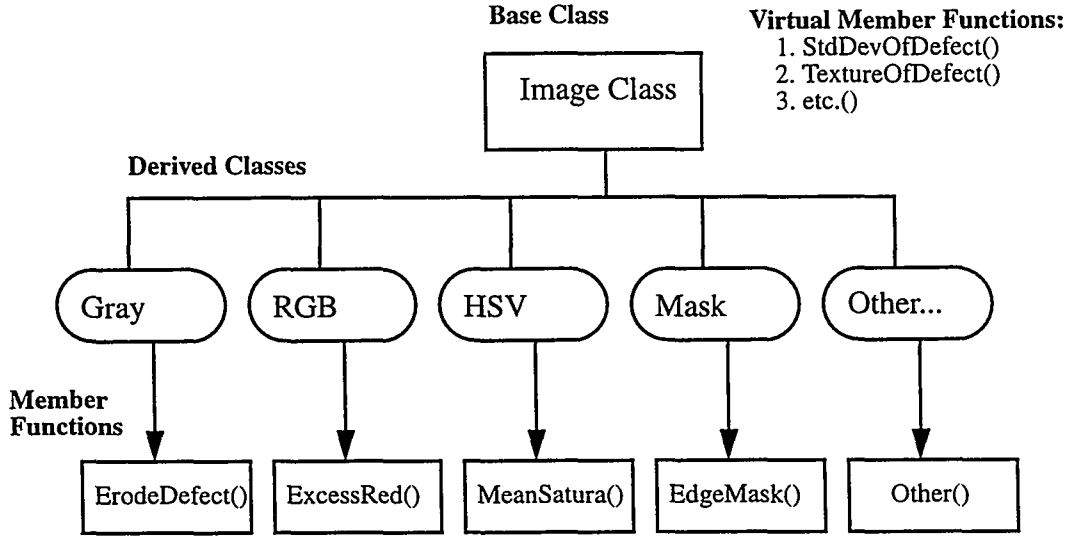


Fig. 5. Image class structure

7.0 References

1. S. Chen, C. F. N. Cowan, P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Trans. on Neural Networks*, Vol. 2, No. 2, March, 1991.

2. R. L. Burden and J. D. Faires, *Numerical Analysis*, p. 445, PWS-Kent Publishing Co., Boston MA, 1989.

Appendix G

ADCtool User's Guide

August 1995

(~~Protected CRADA Information~~)

ADC Tool User's Guide

1. Introduction

This document describes the functionality of the automatic defect classification (ADC) tool prototype, ADCtool, and provides instruction for a user on how to access all available capabilities of the tool. The tool can be described in terms of four functional modules: (1) Data I/O Module, (2) Defect Detection Module, (3) Feature Extraction Module, and (4) Classification Module. The rest of this document describes these modules and their graphical user interface (GUI) in detail.

2. Starting ADC tool

In order to start ADCtool, you must set up one environment variable called ADCTMPDIR. This variable should be set to the directory where you wish the temporary files to be created that ADCtool generates. For example, to set the current directory to be the temporary file repository, use the command "setenv ADCTMPDIR ." in the shell where ADCtool will be started. Next, make sure ADCtool is in your path and type "ADCtool" to start the program.

3. Main ADCtool Interface

The initial screen that appears when the tool is invoked provides access to all four of the main modules in the tool (see Fig.1). This main interface is set up to allow the user to access as little or as much of the configuration details as they require. Each of the modules is briefly mentioned here, but is described in more detail in later sections.

3.1 Data I/O

This button displays the module that allows the user to import image and other types of data into the tool for processing.

3.2 Defect Detection Configuration

This button allows the user access to all of the detailed configuration parameters for the defect detection module. This detailed setup can be configured once by the user, and then batch detection mode can be subsequently used.

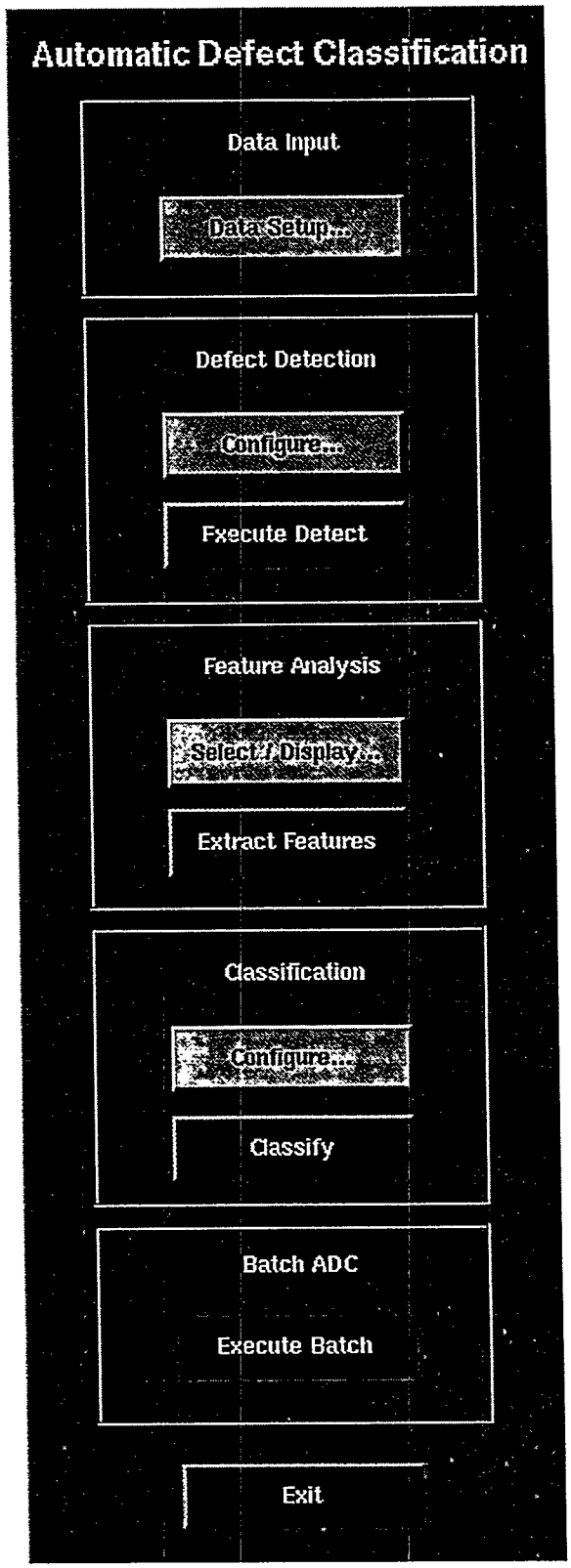


Fig. 1. ADC Tool main interface.

3.3 Batch Defect Detection

Pressing this button simple executes the defect detection step on all image pairs that have been imported and then selected for processing. It uses the detailed configuration parameters that can be set up through the defect detection configuration interface.

3.4 Feature Extraction Configuration

All of the features that are available for extraction and subsequent classification can be accessed by pressing this button without bringing up the detailed configuration window.

3.5 Batch Feature Extraction

After the features are all selected for extraction, this button allows the user to perform the feature extraction from the main window based on the features chosen in the configuration window.

3.6 Classification Configuration

The type of classifier, the classifier configuration, and the data set to be classified can all be set up by pressing this button to display the classifier configuration window.

3.7 Batch Classification

This button simply classifies all imported, selected image pairs that have been through the defect detection process.

3.8 Batch ADC

This function is inactive at this time. Its intended purpose is to perform the entire process of defect detection, feature extraction, and classification on the imported data set, and to display the final classification of each defect on the screen.

4. Data I/O Module

This module allows the user to import image data into the tool for processing (see Fig. 2). All of the data that is associated with one particular defect ID is part of a processing unit.

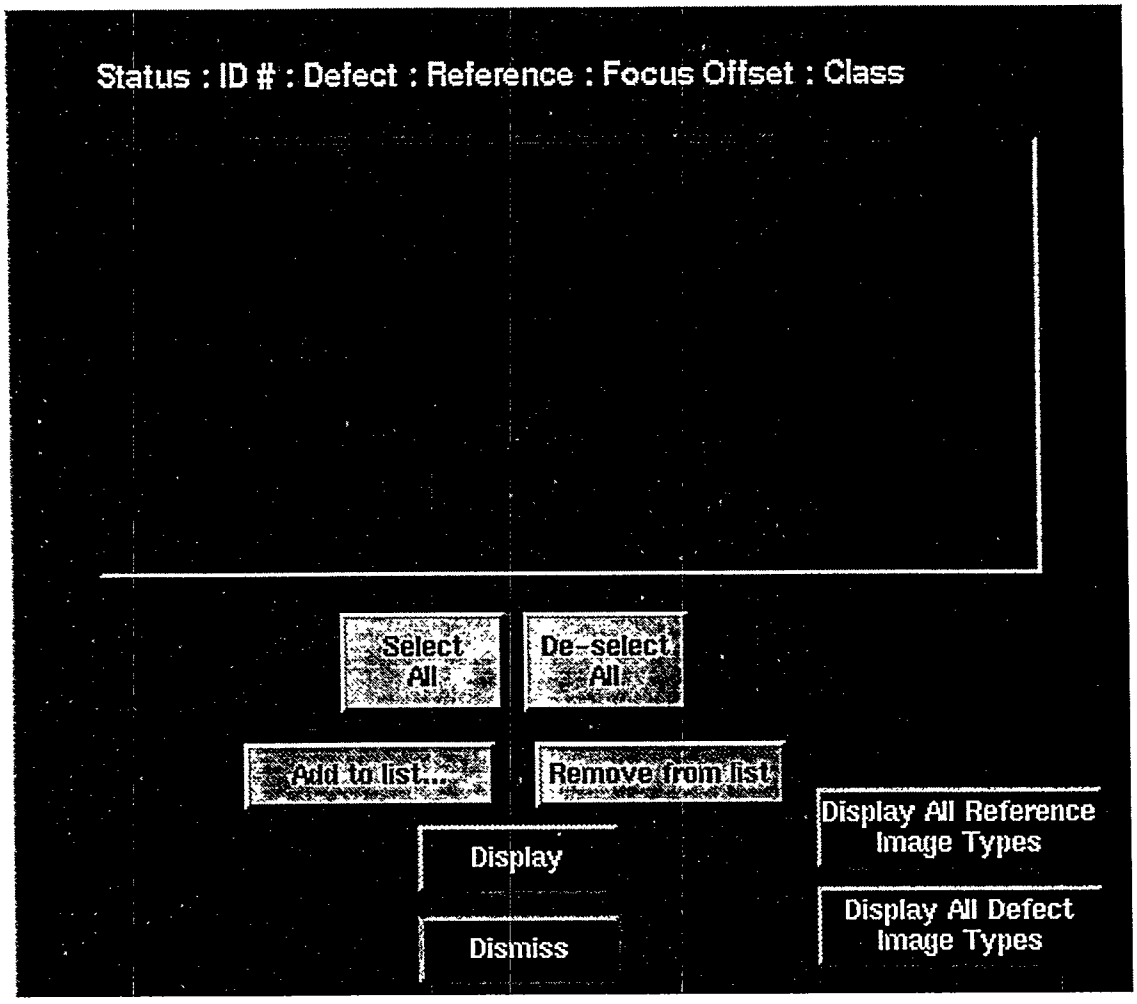


Fig. 2. Data I/O module of the ADC Tool.

4.1 Processing Unit

The processing unit is the basic data block used in the ADC tool. The information about the processing units is displayed in a list format in the Data I/O module. There is a key above the list that identifies all of the processing unit parameters. The descriptors are separated by colons and include the following pieces of information:

Processing Status: This field will be filled with the "-" symbol if the processing unit has not been through the defect detection process. The "!" symbol indicates a processing unit has been partially through the defect detection process. The "*" symbol indicates that the processing unit has completed the defect detection process.

Defect ID: This is a user-assigned, unique identification number that identifies the current defect.

Defect and Reference Images: The defect image file name of a TIFF format, color image that has a defect or anomaly in the scene. The reference image file name of a corresponding TIFF format, color image of the same scene that does not have the anomaly.

Focus Offset: A number which identifies a unique focus offset condition under which the image was acquired. This feature allows several image pairs of the same defect (same defect ID) to be imported with different focus offsets.

Classification (optional): The user-assigned classification for the defect.

4.2 Processing Unit Selection

In order to add processing units to the list one must choose the "Add to list..." button to bring up a file browser (see Fig. 3). There are two different modes of accessing data: (1) individual file access (reference or defect) and (2) collection file access. The operator can choose "Defect Image" or "Reference Image" from the toggle buttons at the top of the file browser to read in individual defect or reference files. Alternatively, one can choose the "Collection" toggle button to read multiple processing units at one time from a prepared ascii file. The first and second lines in the file are the directories for the defect and reference images, respectively. The items starting in the third line are defect id, defect file name, reference file name, focus offset, and classification. An example of a collection file is:

```
/projects/kla/color_set2
/projects/kla/color_set2
0 0014.tif 0018.tif 0.0 PAR
1 0102.tif 0105.tif 0.0 PAR
2 0114.tif 0117.tif 0.0 PAR
3 0212.tif 0215.tif 0.0 FLA
4 0243.tif 0246.tif 0.0 FLA
```

If one is inputting individual files, then after each individual file is added using the "Add" button on the file browser, a window will appear (see Fig. 4) requesting, defect ID, focus offset, and classification information.

After some processing units are on the processing list, they can be selected or deselected individually for processing by clicking on them with the left mouse button. A processing unit selected for processing will be highlighted in white. The "Select All" and "De-select All" buttons are used to select or deselect all processing units on the list.

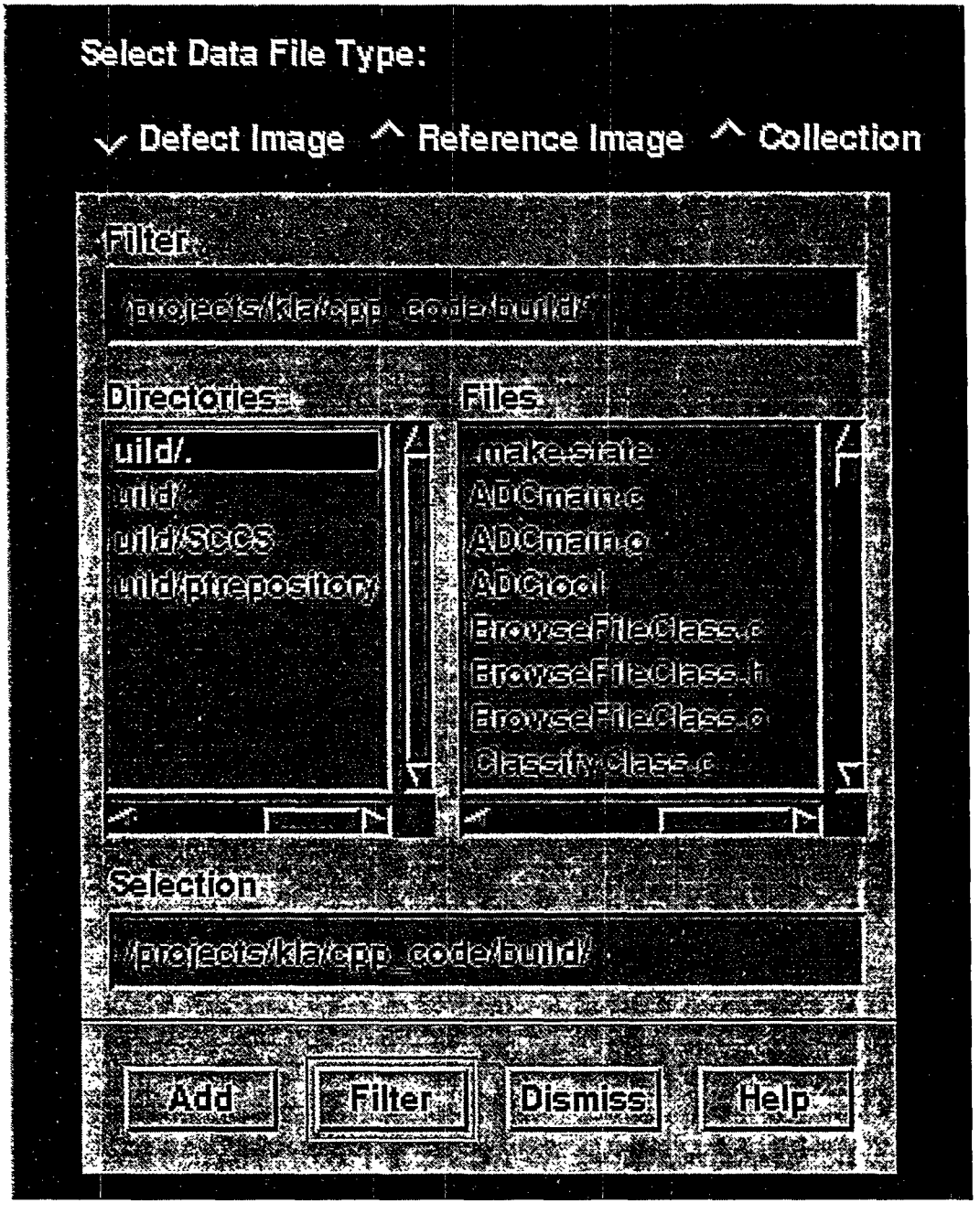


Fig. 3. File browser for selection of image and collection files

To remove a processing unit from the list, simple highlight the unit(s) to be removed and select the "Remove from list" push button.

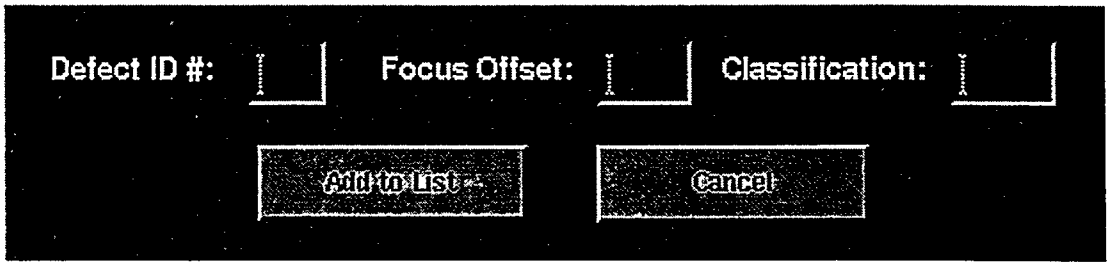


Fig. 4. ADC Tool window requesting defect ID, focus offset, and classification.

4.3 Image Display

The "Display" button will display both color images (defect and reference) for the currently selected processing unit. Only one unit can be selected for display at any one time. The buttons "Display All Reference Image Types" and "Display All Defect Image Types" will display all image types that have been created for the current processing unit (e.g., color image, grayscale image, hue image, green image, etc.). Each image will come up in its own pop-up window using the ImageMagik image display software. For any of the ADCtool image display functions to work properly, the operator must have the "display" program in his or her executable path.

5. Defect Detection Configuration

This window allows the operator to configure the details of the defect detection algorithm (see Fig. 5).

5.1 Current Processing Unit Information

The upper-left corner of the window displays all of the processing unit information for the current unit that is loaded into the defect detection module. Only one processing unit can be accessed at any given time. To change the current processing unit the buttons "Previous Image Pair" and "Next Image Pair" can be selected. This will move through the list in the data I/O module.

5.2 Correlation Parameters

These parameters are used to align the defect and reference images before they are subtracted from one another to isolate the differences between the two.

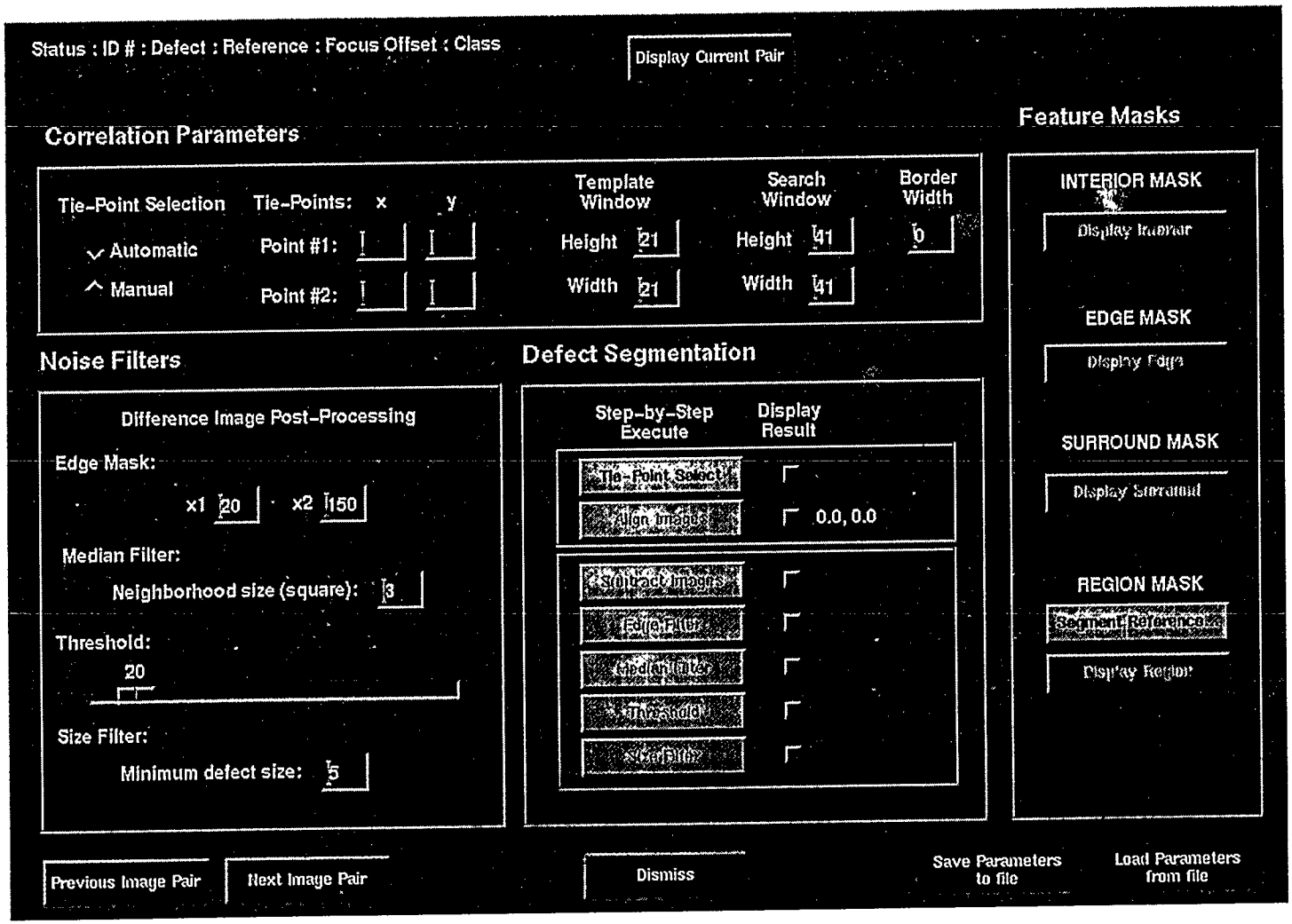


Fig. 5. Defect Detection Configuration module of the ADC Tool.

The tiepoint selection can be either automatic or manual. The automatic mode will use a Hough transform method to find a candidate high-contrast feature in the image for correlation. In manual mode, the operator simply types in the tiepoint coordinates by hand. The template window height and width can be set based on how large the operator wants the tiepoint feature to be. The search window height and width should be set according to how much misalignment will exist between defect and reference images. For example, if the template window size was 21 by 21, and one expected 10 pixels possible misalignment in both X and Y, then the search window size would be 41 by 41 (21 + 10 + 10). The border width parameter will be used to determine how much of the original image border to cut off before processing. This is convenient when the borders of the original image are not needed.

5.3 Noise Filters

This area of the defect detection configuration window allows the operator to adjust the various noise filtering operations applied to the difference image to remove spurious defects.

Edge Mask. The "x1" and "x2" fields of the edge mask define the width of the mask at edge transitions. The edge mask suppresses noise along the edges of the image where there is a high potential for noise caused by slight misalignments in the two images.

Median Filter. The median filter neighborhood size defines the size of the square neighborhood used in this filtering step. This filter will remove speckle noise from the difference image.

Threshold. The threshold value is the pixel level used in the binarization of the difference image to create the defect mask. This mask will be white where the defect is located and black where there is no defect information.

Size Filter. The size filter can be set to remove any defects from the defect mask image that are below a certain size (in pixels).

5.4 Defect Segmentation

This area of the defect detection module is where the actual defect detection process get executed. The steps of the defect detection process can be performed by the user one at a time. If the "Display Result" toggle button is selected when a particular step of the detection operation is performed, then the result of that operation will be displayed for the oper-

ator in a window. This allows detailed monitoring of each step in the detection process. The numeric fields to the right of the "Align Images" button will show the pixel shift required to align the defect and reference images.

5.5 Feature Masks

This section of the GUI allows the operator to view the four different types of defect masks generated by the defect detection process.

Interior Mask. The interior mask is the result of the defect detection process and it defines where the interior of the defect is located within the image. The mask will be white where the defect is located and black elsewhere.

Edge Mask. The edge mask is created by eroding the interior mask and subtracting it from a dilated version of the interior mask. The result is a mask that is white only on the edges of a defect.

Surround Mask. The surround mask is created by subtracting the interior mask from a dilated version of the interior mask. The mask covers the area around the defect, and provides context information about what type of material the defect is located on.

Region Mask. This mask is defined by the region of common material that the defect is located on. It cannot be created before pressing the "Segment Reference..." button to automatically segment the reference image into contiguous areas of common material. The segment reference configuration window (See Fig. 6) allows the operator to set the widths of the Gaussian kernels used in the difference-of-Gaussians edge detection algorithm. The "Segment Reference" button performs automatic reference image segmentation. The 'Display Segmented" button will display a pseudocolor image of the contiguous regions of common material found by the segmentation process.

5.6 Parameter Files

The button "Save Parameters to File" will put all of the user-selectable information into an ascii file. This file can be reloaded at a later time by selecting "Load Parameters from File".

6. Feature Extraction Configuration

The ADCtool provides a very flexible feature extraction environment (see Fig. 7).

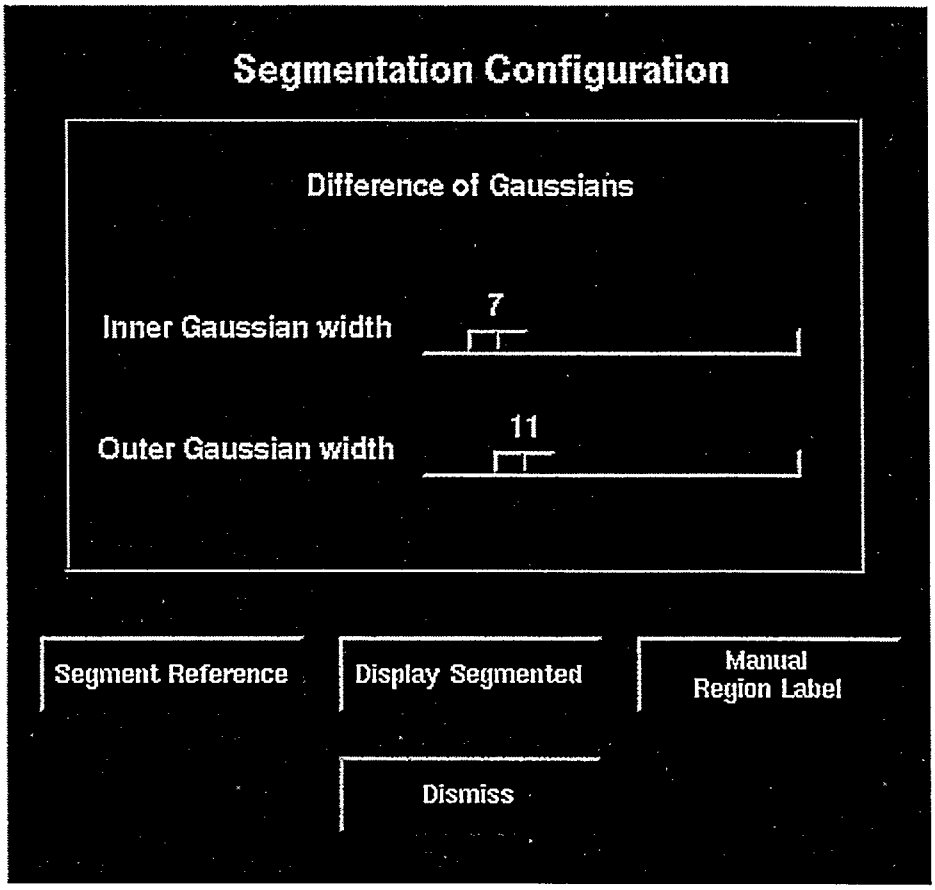


Fig. 6. Reference image segmantation module of the ADC Tool.

6.1 Feature List Type

There are three types of defect feature lists that can be generated using the ADC tool. These can be selected by using the "Feature List Type:" option menu.

Defect List. The defect list selection will apply the user-selected feature extraction algorithms only to the defect image. This feature list will be numerical descriptors of defect features only.

Reference Region List. Choosing this type of feature list will result in all of the feature extraction algorithms being applied to the reference region area. The reference region area is the region of common material in which the defect is located. If this option is selected, then the "Import Labeled Reference..." button will become active, and it must be used to import a labeled reference image. See the instructions on creating a segmented reference image in the Defect Detection Configuration section.

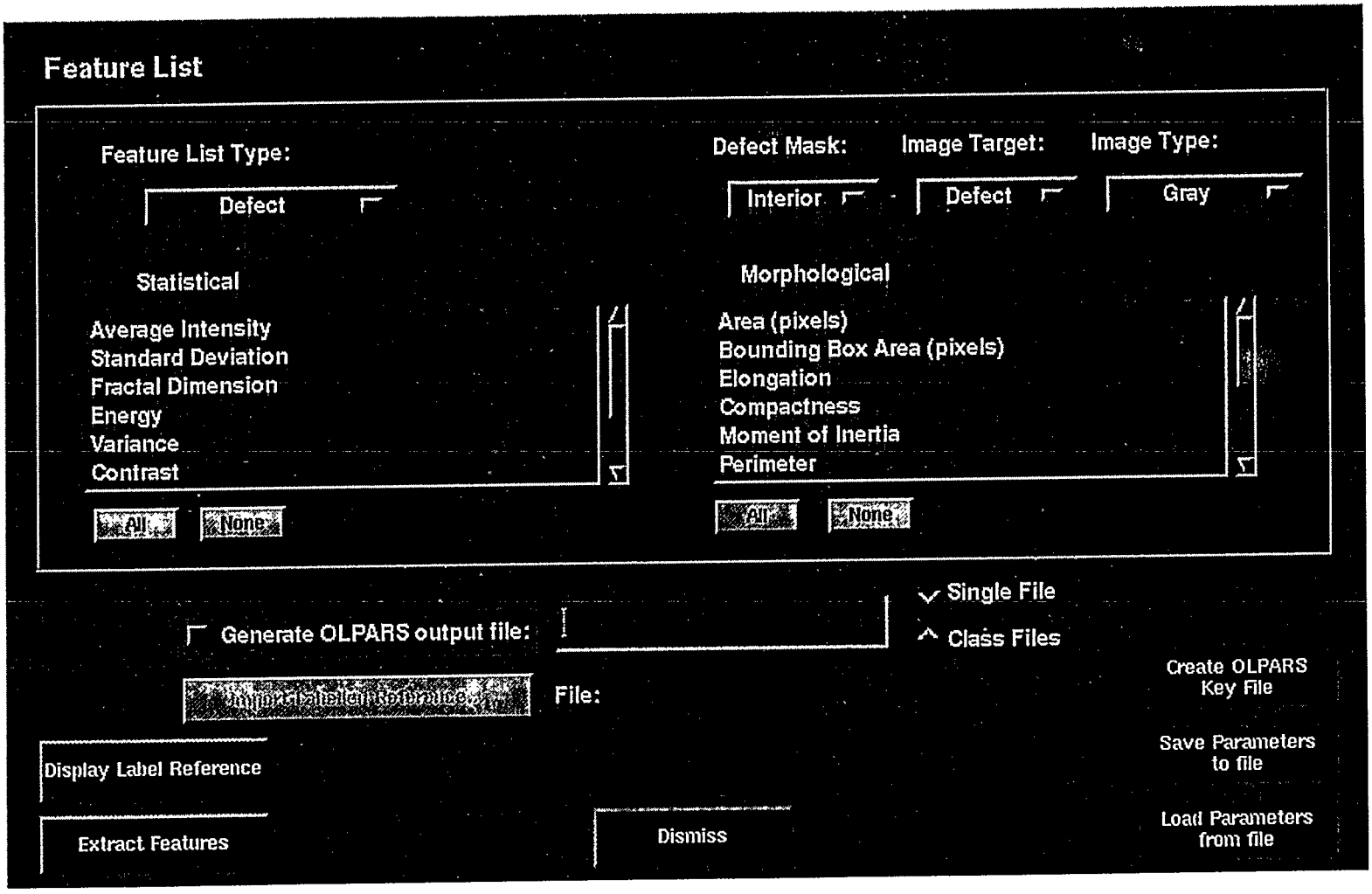


Fig. 7. Feature extraction configuration module for the ADC Tool.

Combined List. This list allows the operator to create one list of features which includes features about both the defect itself and the reference region in which the defect is located.

6.2 Defect Mask

The "Defect Mask" option menu selects one of the four defect masks that define where feature extraction algorithms are to be applied. The four masks are: (1) interior, (2) edge, (3) surround, and (4) region.

6.3 Image Target

In addition to selecting the appropriate mask, the operator can apply the feature selection algorithms to one of three image targets: (1) Defect, (2) Reference, and (3) Difference by setting up the "Image Target" option menu.

6.4 Image Type

Once the image target is selected, the "Image Type" option menu allows the operator to specify whether the feature extraction algorithms will be applied to a Gray, Red, Green, Blue, Excess Red, Excess Green, Excess Blue, Hue, or Saturation version of the image target.

6.5 Features

A variety of features can be extracted from the target image. The features are divided into statistical features and morphological (shape) features. The features can be selected or unselected for extraction by clicking on them with the left mouse button, or by using the "All" or "None" buttons below each list. After the desired features are all selected, then the "Extract Features" button will calculate the highlighted features for all of the selected processing units according to the Data I/O module.

6.5.1 Statistical

This list of features uses intensity values of the pixels in the target image to determine the feature value. The features included in the list are:

Average Intensity
Standard Deviation
Fractal Dimension
Energy
Variance

Contrast

Angular 2nd Moment

Entropy

Dispersion

6.5.2 Morphological

This list of features is based on the shape of the defect mask selected by the operator. These shape features are:

Area (pixels)

Bounding Box Area (pixels)

Elongation

Compactness

Moment of Inertia

Perimeter

Box-to-Perimeter

Major Axis

Minor Axis

Number of Blobs

Width

Height

6.6 OLPARS Interface

The ADC tool will generate feature vector output files in a format the is readable by the On-Line Pattern Analysis and Recognition System (OLPARS) software package. OLPARS allows visualization of feature vector data, and also contains several classifiers that can be tested for applicability to a feature data set.

By selecting the "Generate OLPARS output file:" toggle button and entering a file name, the ADC tool will write out all feature values into the user-assigned output file name. If one intends to test unsupervised classification techniques (i.e., unclassified defect feature data), then the "Single File" toggle button option should be selected. This puts all feature vectors for all defects into the same file. On the other hand, if one has classified defect data amenable to supervised classification techniques, then "Class Files" should be chosen. This will put all defect feature vectors of the same class into one file.

OLPARS has a limitation in that it will not be able to read the actual feature names from the OLPARS file ("Energy" or "Contrast", for example). If the operator selects the "Create OLPARS Key File" button, a text file will be generated which lists the names of the

features in the order that they are written into the OLPARS file. This file can be used to identify the name of each feature while analyzing the data in the OLPARS environment.

6.7 Parameter Files

Similar to the Defect Detection Configuration module, the feature extraction configuration information concerning features, feature list type, defect mask, image target, and image type can be saved to a text file. An operator can then reload their set of features from a previous session with the ADC tool.

7. Classifier Configuration

This window contains configuration and execution options for all available classifiers in the tool (see Fig. 8). At the time of this document, only one classifier is available for use within the ADCtool, the Fisher Pairwise classifier. Other classifiers are available in the OLPARS environment.

The feature list type options are 'Defect", "Reference Regions", and "Combined". The operator should select the same type of feature list as was selected in the Feature Extraction Configuration module. Refer to the Feature Extraction Configuration section for details on each of the feature list types available.

The classifier configuration file contains all of the coefficients, etc. required to fully define the selected classifier type. The tool currently reads a Fisher Pairwise classifier configuration file generated by the OLPARS software package. Selecting the "Load Classifier Configuration File..." button will allow the operator to import such a configuration file.

When the operator has specified the classifier, the "Classify Current Selections" button will prompt the user for an output results file name and then will perform the classification.

The last section in this module is the "Classifier Specific Parameters" area. Depending on the design of future classifiers for this ADCtool, several classifier configuration fields may be present to allow the operator to easily adjust classifier parameters for better performance. At this time, these fields are inactive and for example purposes only.

The "OLPARS..." button will start the OLPARS software package as a separate task.

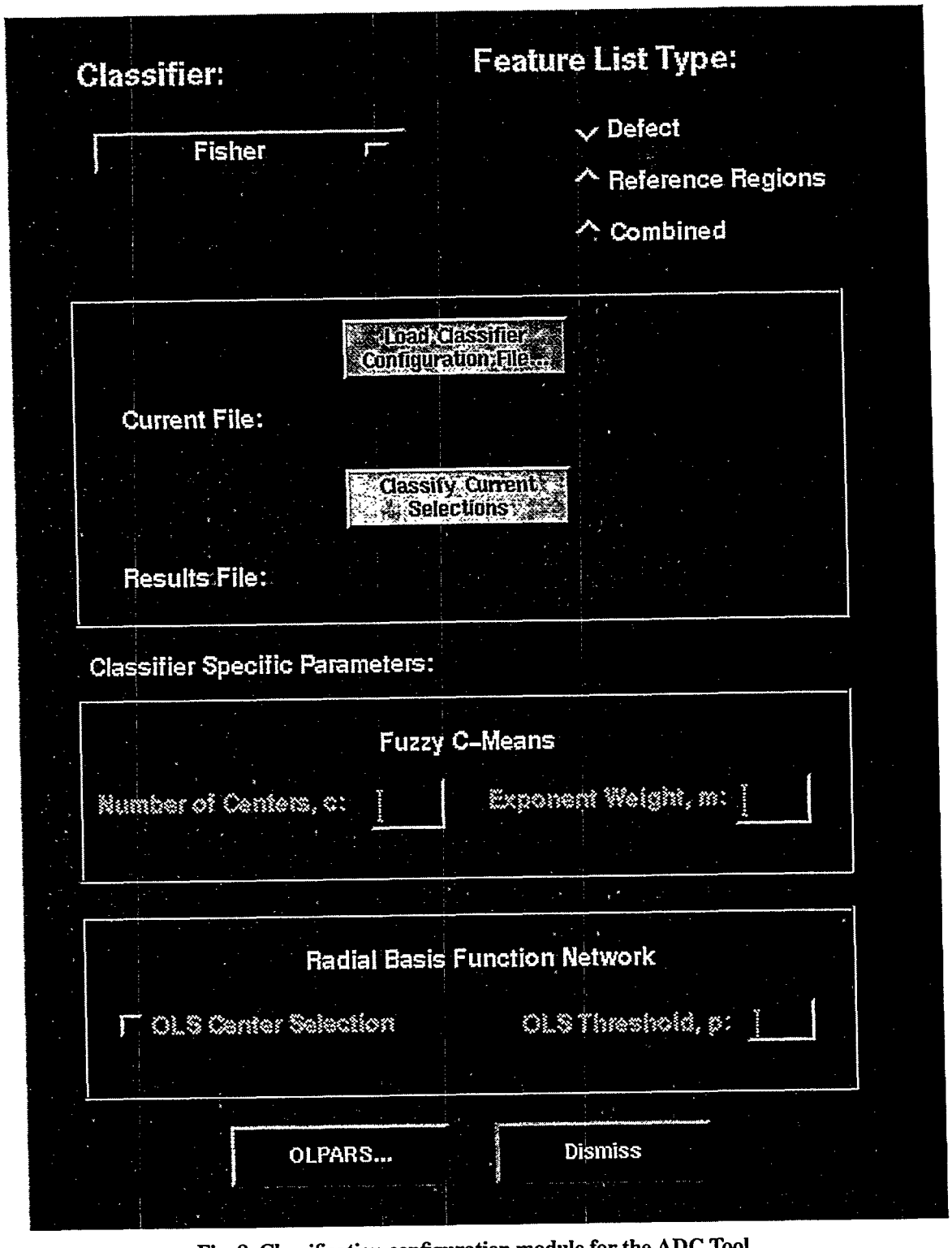


Fig. 8. Classification configuration module for the ADC Tool.

8. Batch ADC

This module is not currently available, but when implemented will allow an operator to read in a list of processing units, and then classify all of the detected defects while reporting results back to the operator as each defect is analyzed.

Appendix H

ADCtool Code Technical Documentation

August 1995

(~~Protected CRADA Information~~)

ADCtool Code Technical Documentation

The top level class in the ADCtool software is the *ProcUnit.* This class represents all the information associated with a given defect site and contains member functions to detect the defect and extract features. A defect site will have two high resolution color images associated with it at the beginning of the defect classification process which are contained in the ColorImage data member within *ProcUnit.* As the processing progresses, additional data members in *ProcUnit* are calculated from the base color images. Finally, the characteristic features of the defect are stored in a *Feature* class. Figure 1. shows a hierarchy of the various software objects in the ADCtool. At the top level is a dynamically linked list of pointers to *ProcUnit* objects. This list based approach enables batch processing of many defect sites. Each of the relevant classes shown in Fig. 1. will be described in additional detail below.

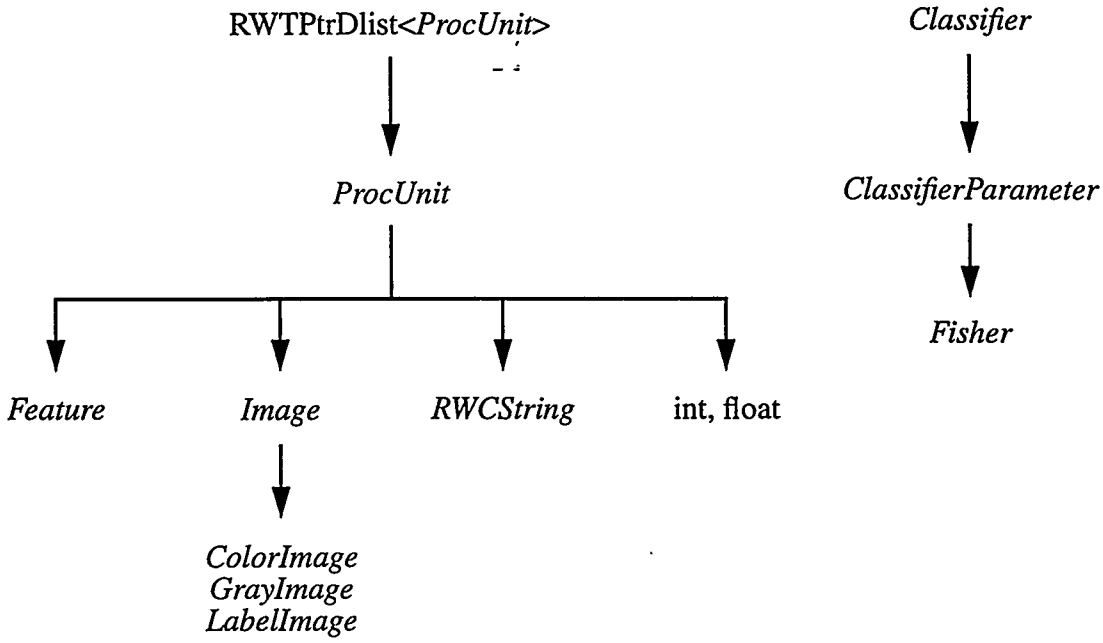


Fig. 1. Overall structure of software objects. Arrows indicate lower level of detail.

Another top level element is the *Classifier* class which contains the techniques and parameters required to assign a classification to the given defect site based on the features contained in a *ProcUnit.* The *Classifier* object contains data members which specify the classification algorithm and the necessary parameters to compute the defect classification based on a given feature set. The detail of the classes related to the classification process will be given below.

The *ProcUnit* class contains the data members shown in Fig. 2. This class is the "glue" for the information necessary for the processing associated with a defect and reference color image. The significant data members are the various image classes and the linked list of *Feature* objects. Each of the specific data members and member functions are described in the header file, proc_unit.h, for this class.

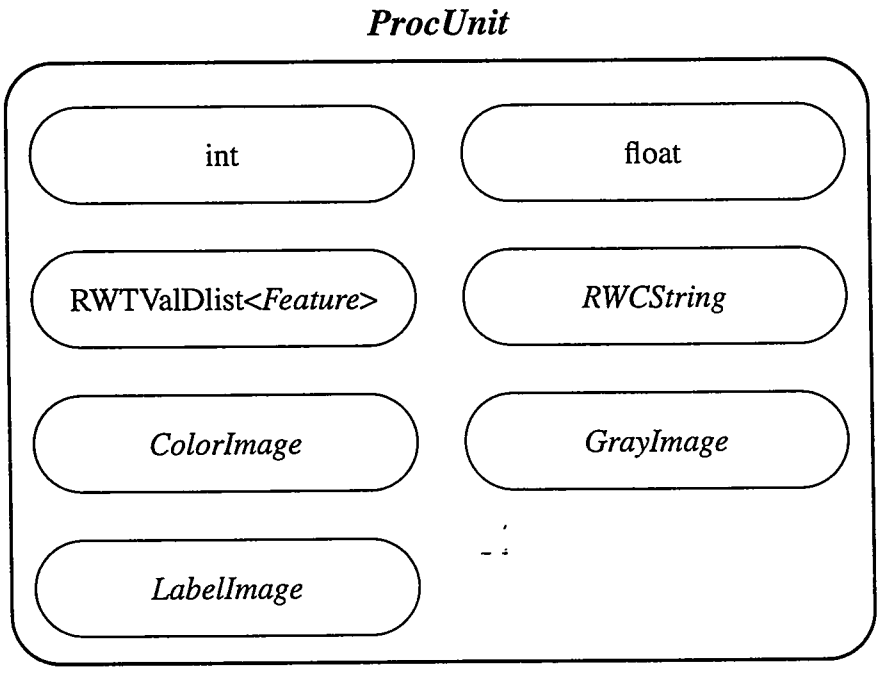


Fig. 2. Data member types in the *ProcUnit* class.

The *Feature* class is a simple encapsulation of the name and value of a feature. In addition the various possible defect masks, source images, and color types are contained in this class. Figure 3. shows the data members of the *Feature* class. Member functions in this class computer the feature value based on the images contained in the *ProcUnit* class. The specific description of each data member and member function is listed in the header file, features.h, for this class.

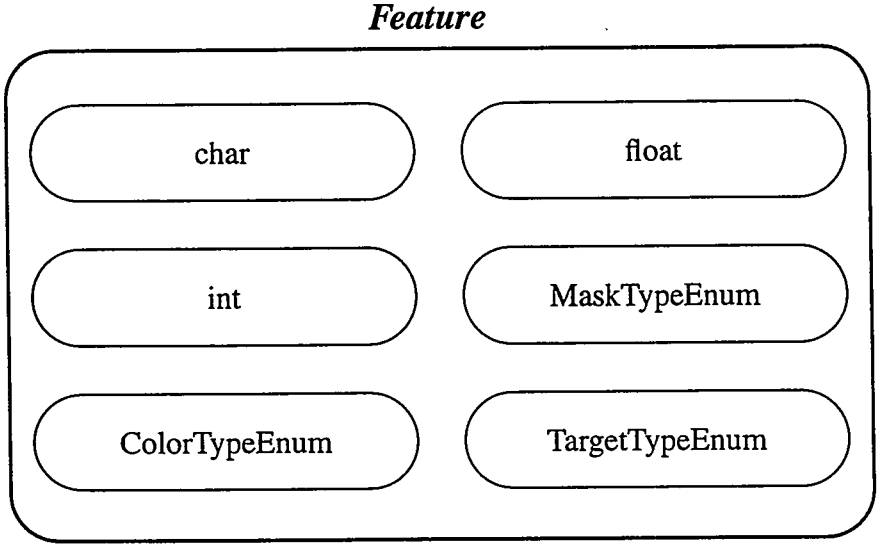


Fig. 3. Data member types in the *Feature* class.

The *Image* class is the base class for representation of two dimensional intensity data. This class has as its primary data member a pointer to the Khoros data structure xvimage. Since many of the processing algorithms utilized Khoros functions, the fairly complete structure of the xvimage was utilized. The basic data members of the *Image* class are shown in Fig. 4 and also described in the appropriate header file, image.h.

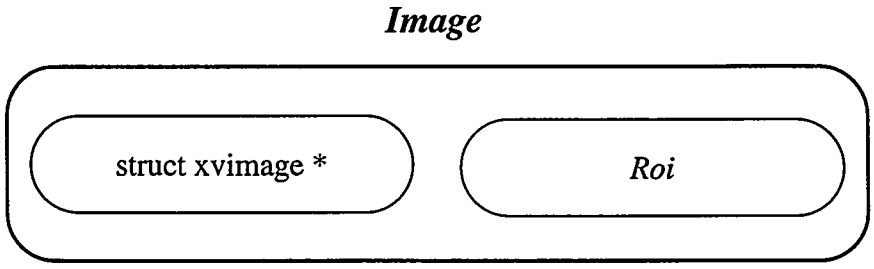


Fig. 4. Data member types in the *Image* class.

Several classes are derived from the base *Image* class including the *ColorImage* and *GrayImage*. These classes are basically identical with the *ColorImage* class having three unsigned char data members and the *GrayImage* having one. The member functions for these classes are primarily responsible for reading and writing data from/to files. Figure 5 shows the inheritance relationship and data member of both the *ColorImage* and *GrayImage* classes. Additional details are given in the color.h and gray.h header files.

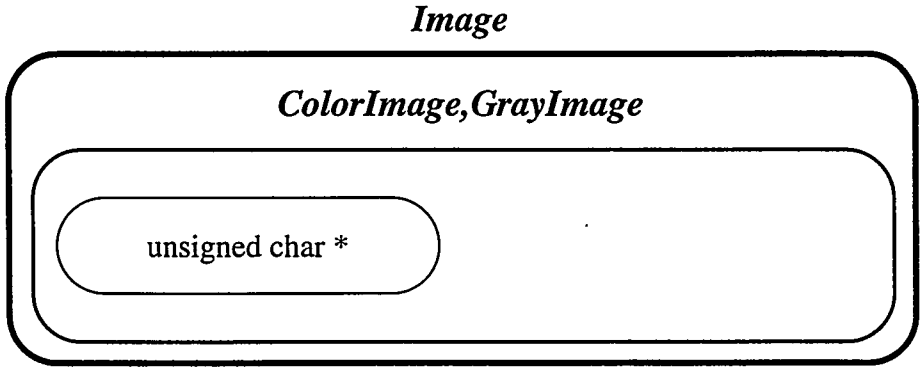


Fig. 5. Data member types and inheritance in the *ColorImage* and *GrayImage* classes.

A similar class to the *GrayImage* is the *LabelImage* class. This class differs only in that the data member is an integer pointer which enables the label image to contain more than 256 discrete labeled regions. The header file label.h describes the data members and member functions.

The *Classifier* class is designed to represent the information associated with the defect classification operation. This class is totally separate from the classes related to the *ProcUnit* class. Where the *ProcUnit* class deals with detecting and characterizing individual defects, the *Classifier* class applies the classification algorithm represented by the class to any defect. The key data members are shown in figure 6. The pointer to the linked list of *ProcUnits* is the link to the individual defects and the data member class *ClassifierParameter* contains the necessary parameters of the

classifier. Definitions of the data members and member functions are given in the header file classifier.h.

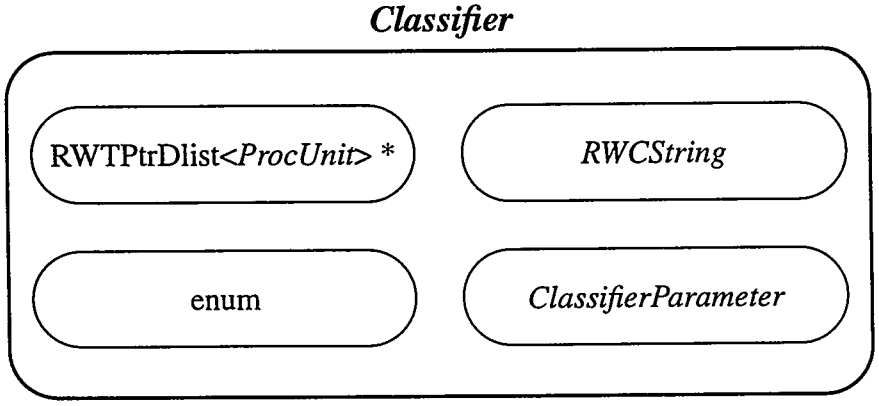


Fig. 6. Data member types in the *Classifier* class.

The *ClassifierParameter* class is a base class which contains information common across different classification algorithms. This base class contains virtual functions which actual compute the class assignment of a defect with a given classification algorithm. Figure 7 shows the basic data member of this class and additional details are given in the class_param.h file.

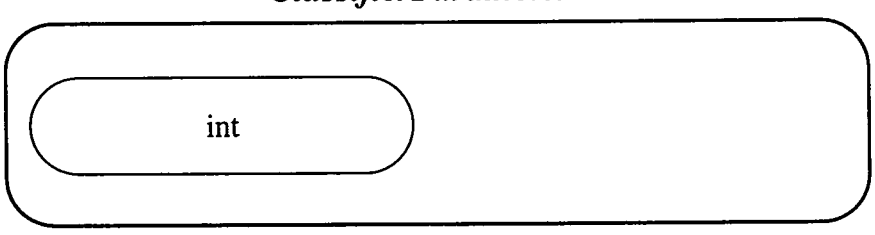


Fig. 7. Data member types in the *ClassifierParameter* class.

An example of a classification algorithm is the Fisher pairwise discriminate method. This algorithm requires an n-dimensional function for each pairwise class comparison. The Fisher class contains the specific coefficients compute the distance measures. Figure 8 shows the data members and inheritance relationship of this class. Specific descriptions of the data members and member functions are given in the fisher.h header file.

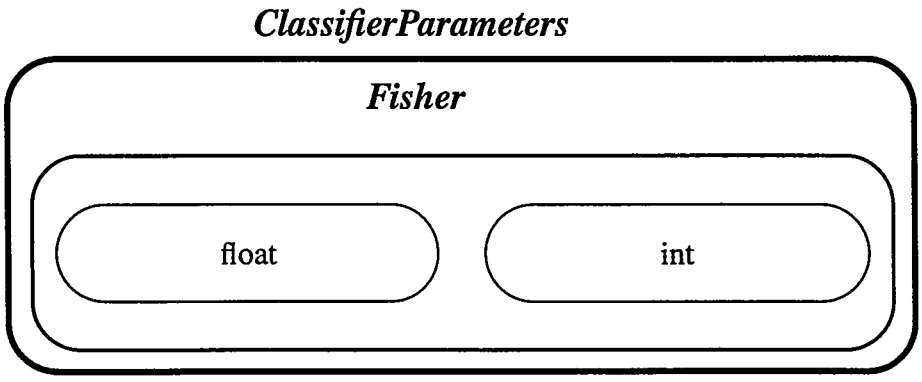


Fig. 8. Data member types and inheritance in the *Fisher* class.

Distribution:

Lockheed Martin Energy Systems, Inc./DOE-OR
S. S. Gleason, MS-6011, 3546
D. W. McDonald, MS-6005, 3500
K. W. Tobin, MS-6011, 3546
M. A. Hunt, MS-6011, 3546
H. Sari-Sarraf, MS-6011, 3546
Brian B. Bovee, MS-8242, 701SCA
Erwin Epple, MS-8084, 9203
William P. Painter, MS-6416, 5002
A. K. Lee/DOE-OSTI, MS-8175, 9983-30 (2 copies)
DOE Office of Patent Counsel, FOB

KLA Instruments Corporation
Ashok Kulkarni, P.O. Box 49055, San Jose, CA 95161-9055 (5 copies)