RAPID CAPTURE, ANALYSIS, & STORAGE OF INTEGRATED CIRCUIT IMAGES FOR DEFECT ANALYSIS

By

JONATHAN NATION

A Thesis Submitted to The Honors College
In Partial Fulfillment of the Bachelors degree
With Honors in
Optical Engineering
THE UNIVERSITY OF ARIZONA
MAY 2010

Approved by:

Martha Ostheimer
College of Engineering (Interdisciplinary Advisor)
The University of Arizona Electronic Theses and Dissertations
Reproduction and Distribution Rights Form

<table>
<thead>
<tr>
<th>Name (Last, First, Middle)</th>
<th>Nation, Jonathan, Scott</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree title (eg BA, BS, BSE, BSB, BFA):</td>
<td>BSE</td>
</tr>
<tr>
<td>Honors area (eg Molecular and Cellular Biology, English, Studio Art):</td>
<td>Optical Engineering</td>
</tr>
<tr>
<td>Date thesis submitted to Honors College:</td>
<td>5/5/2010</td>
</tr>
<tr>
<td>Title of Honors thesis:</td>
<td>Rapid Capture, Analysis, and Storage of Integrated Circuit Images for Defect Analysis</td>
</tr>
</tbody>
</table>

I hereby grant to the University of Arizona Library the nonexclusive worldwide right to reproduce and distribute my dissertation or thesis and abstract (herein, the "licensed materials"), in whole or in part, in any and all media of distribution and in any format in existence now or developed in the future. I represent and warrant to the University of Arizona that the licensed materials are my original work, that I am the sole owner of all rights in and to the licensed materials, and that none of the licensed materials infringe or violate the rights of others. I further represent that I have obtained all necessary rights to permit the University of Arizona Library to reproduce and distribute any nonpublic third party software necessary to access, display, run or print my dissertation or thesis. I acknowledge that University of Arizona Library may elect not to distribute my dissertation or thesis in digital format if, in its reasonable judgment, it believes all such rights have not been secured.

Signed: [Signature]
Date: 5/5/2010

Last updated: Nov 15, 2009
Thesis Group Members and Roles:

Jonathan Nation – Optical design engineer. He assisted in the assembly of the test system. He also conducted all the trials on the test system, collecting all the necessary data. He analyzed the defect images and their associated software detection results, and found patterns in defects among the different illumination settings used. He used this information to draw conclusions as to the effect of the different parameters of the illumination on different defect categories, and devise the best illumination settings to detect a given defect. He contributed to several parts of the paper including the system requirements, optical design concepts, hardware system analysis, and the illumination optimization tests, results, and conclusions.

Mark Wiley – Optical design engineer. He machined and built the hardware to use with the test bed assembly. He assisted in the optimization testing to collect data images. He contributed to the multiple parts of the technical report including optical design concepts, system build with regards to hardware mounting, analysis of the optical system, and the proposed design.

Travis Rippstein – Computer engineer. He handled the design and testing of the compression algorithms for compressing defect images. He created a set of Matlab routines and a GUI to allow for easy user interaction with the saved defect images. He contributed to several parts of the technical report, including the system build and testing sections with regards to compression software, the conclusions and design recommendations for software, the top level system design, the introduction, and appendix C.

Maribel Hudson – Computer Engineer. She handled the design and testing of the software algorithms for defect detection on test images. She created a set of Matlab routines to detect the defects on a given image and display the detected defects overlaid on that image. The system integrated with the compression system. She contributed to the technical report in the system requirements, and in the software design, analysis, and test for the defect detection software.

Frederick Chyan – Electrical Engineer. He assisted in the assembly of the test bed system, and in creating an on/off switch for the system. He also assisted the optical engineers in collecting test data for the illumination optimization. He contributed to the technical report in the introduction, system requirements, final system tests, and in the formatting of the document.
Rapid Capture, Analysis, and Storage of Integrated Circuit Images for Defect Analysis

Team Number: 3551
Team Members: Frederick Chyan
               Maribel Hudson
               Jonathan Nation
               Travis Rippstein
               Mark Wiley

Submitted to: Anthony King
Date: April 30, 2009

ENGR 498b
College of Engineering
University of Arizona
Abstract

An automated rapid capture, analysis and storage system for identifying chip defects is essential to ensure Intel’s “Copy Exactly” methodology with low labor cost and no human subjectivity. In this system, the software will identify where the defects are with a standardized definition of defects. Constraints include the 80% defect detection rate, and Windows XP compatibility. There should be no defects that can be observed by human eye in the product shipments. Three design concepts were generated and two of them are combined to form the final design. The final design consists of a line scan cameras and two LED arrays with different wavelengths. The software design is discussed under detection and compression algorithms. Based on the software testing done, the software can identify on average 96% of the defects and compress the image up to 22 times over the current bitmap format.
Contents

Abstract ................................................................................................................................................. vi

List of Figures ...................................................................................................................................... ix

List of Tables ........................................................................................................................................ vii

1. Introduction ...................................................................................................................................... 1

2. System Requirements ....................................................................................................................... 2

3.1. Optical Concept 1: One camera, one wavelength, optimized positions ..................................... 4

3.2. Optical Concept 2: One camera, two wavelengths, two passes ................................................. 4

3.3. Optical Concept 3: One camera, one wavelength, two passes .................................................... 4


4.1 Detection Concept 1: Sum of Differences Method .......................................................................... 5

4.2 Detection Concept 2: Contour Analysis of a Threshold Image ....................................................... 5

5. Compression Software Design Concepts ............................................................................................ 8

5.1 Compression Concept 1: MPEG4™ Adaptation of Interprediction ................................................. 8

5.2 Compression Concept 2: Custom Algorithm for Grayscale Image Compression ........................... 9

6. Design Analysis ................................................................................................................................. 10

6.1. Hardware system analysis ............................................................................................................... 10

6.1.1: Camera System Analysis .......................................................................................................... 10

6.1.2: Illumination System Analysis .................................................................................................... 11

6.1.3: Analysis of Reflectivities using two separate sources ............................................................... 11

6.2. Detection software system analysis ............................................................................................... 13

6.3 Compression Software System Analysis .......................................................................................... 15

6.3.1 L-JPEG and LL-JPEG (Lossy and Lossless) .............................................................................. 15

6.3.2 L-Gold and LL-Gold (Lossy and Lossless) ............................................................................... 16

6.3.3 H-JPEG (Hybrid) ....................................................................................................................... 16

7. Proposed Design ............................................................................................................................... 18

8. System Build .................................................................................................................................... 19

8.1. Hardware Mounting: .................................................................................................................... 19

8.2. Hardware Calibration ..................................................................................................................... 21

8.3. Software Algorithms ...................................................................................................................... 22

8.4. GUI (Graphics User Interface) Guide ........................................................................................... 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9. Illumination Optimization</td>
<td>24</td>
</tr>
<tr>
<td>10. System Testing</td>
<td>28</td>
</tr>
<tr>
<td>10.1. Illumination Settings Tests</td>
<td>28</td>
</tr>
<tr>
<td>10.2. Detection Software Tests</td>
<td>29</td>
</tr>
<tr>
<td>10.3 Compression Software Testing</td>
<td>30</td>
</tr>
<tr>
<td>11. Design Results</td>
<td>31</td>
</tr>
<tr>
<td>11.1 Optimized Illumination Conditions</td>
<td>31</td>
</tr>
<tr>
<td>11.2 Detection Test Results</td>
<td>36</td>
</tr>
<tr>
<td>12. Conclusions and Design Recommendations</td>
<td>39</td>
</tr>
<tr>
<td>13. Acknowledgements</td>
<td>40</td>
</tr>
<tr>
<td>14. References</td>
<td>40</td>
</tr>
<tr>
<td>Appendix A: Contributions</td>
<td>41</td>
</tr>
<tr>
<td>Appendix B: Hardware Models</td>
<td>42</td>
</tr>
<tr>
<td>Appendix C: Software Code</td>
<td>46</td>
</tr>
<tr>
<td>Appendix D: Images for Case Study</td>
<td>58</td>
</tr>
<tr>
<td>Appendix E: Compression Modes / Images</td>
<td>60</td>
</tr>
<tr>
<td>Appendix F: List of Test Images Captured During Optimization</td>
<td>66</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Flow chart of steps taken to find contours in an image .............................................. 6
Figure 2: Threshold of image with defect and the Threshold of image without defect ............... 14
Figure 3: Flexlink conveyor. Aluminum frame with T-slots allows mounting of components using metric bolts and square nuts provided by Flexlink ............................................. 19
Figure 4: Camera mounting Plate .................................................................................................. 20
Figure 5: LED array mounting ..................................................................................................... 21
Figure 6: The Graphical User Interface for processing captured chip images .......................... 23
Figure 7: Image showing over saturation using IR-red combination of wavelengths .......... 27
Figure 8: Output image of defect detection .................................................................................. 29
Figure 9: Red-Red illumination with 70-60 degree angles and 3”-3.75” working distances ...... 32
Figure 10: Blue-Red illumination with 50-50 degree angles and 3” working distances .......... 33
Figure 11: IR-Red illumination with 72-61 degree angles and 3”-5” working distances ............ 34
Figure 12: IR-Blue illumination with 73-54 degree angles and 3.25” working distances .......... 35
Figure 13: Technical Drawings for the Dalsa Piranha 3 camera system ................................. 42
Figure 14: Responsivity through spectrum at 0dB gain for Dalsa Piranha 3 camera system .... 43
Figure 15: Technical drawings for the Cobra linescan LED illumination ................................. 43
Figure 16: Relative Intensity and line thickness vs. the working distance of the Cobra linescan LED arrays ................................................................. 44
Figure 17: Solid Model of the system design chosen. There are two sets of two linescan illuminators and two camera systems, pointed at a conveyor belt where the tray of chip moves along ................................................................. 44
Figure 18: Solid Model top view of the system design. Cameras are centered over the focus line of the illumination system ................................................................. 45
Figure 19: “Color chip” image used for image #1 in case study .................................................. 59
Figure 20: “Altered chip” image used for image #2 in the case study ........................................ 59
Figure 21: The Original 5811KB Image .................................................................................. 60
Figure 22: The L-JPEG Mode image, lossy 257KB ................................................................. 61
Figure 23: The LL-JPEG Mode image, lossless 2425KB ......................................................... 62
Figure 24: The L-Gold Mode “defective” portion, lossy 47KB .................................................. 63
Figure 25: The LL-Gold Mode “defective” portion and the H-JPEG “defective” portion, lossless 422KB .............................................................................................................................. 64
Figure 26: The H-JPEG “non-defective” portion, lossy 222KB .................................................. 65
List of Tables

Table 1: Top Level functional requirements for our design proposal and metrics / target values
for each metric used to gauge system performance................................................................. 2

Table 2: System level constraints that must be adhered to in our design. .............................. 3

Table 3: Interfaces present in our system design..................................................................... 3

Table 4: Sample calculations for DN registered by the system for the case of two different LED
arrays used for a high and a low reflectivity object. The last two columns show the case of Red
illumination used for both high and low illumination............................................................ 13

Table 5: Compression Modes .................................................................................................. 15

Table 6: LEDs used in optimization process .............................................................................. 24

Table 7: Results of testing a 5811KB captured image with different compression modes........ 30

Table 8: Defect detection software results on the test chip using the best settings for each
wavelength pair......................................................................................................................... 31

Table 9: Best illumination settings and relative strengths of each wavelength pair tested........... 36

Table 10: Detection test results.................................................................................................. 37

Table 11: Results Table for detection software ........................................................................ 38

Table 12: Individual Contributions ............................................................................................ 41

Table 13: File Sizes and Pixel Area............................................................................................ 59

Table 14: Serial numbers of the test chips used. Row 1 was the main row of test chips, with
uniform lighting on this row. Row 2 was a secondary row that did not have uniform lighting, so
it was not used for any comparisons....................................................................................... 66

Table 15: Angles of the images captured for stage 1 of the optimization, using only a single LED
on the right side, at a working distance of 3”. ........................................................................... 66

Table 16: Settings for the red-red wavelength pair illumination optimization tests.................. 66

Table 17: Settings for the blue-red wavelength pair illumination optimization tests............... 67

Table 18: Settings for the IR-red wavelength pair illumination optimization tests............... 67

Table 19: Settings for the IR-blue and IR-UV wavelength pair illumination optimization tests. 68
1. Introduction

Intel was founded in 1968 by Gordon Moore and Robert Noyce in Santa Clara. Andy Grove ran the company for much of 1980's and 1990. By the end of the 1990's Intel was one of the largest and most successful companies in the world. They are the world’s largest semiconductor manufacture, and as such, use state-of-the-art manufacturing and packaging processes. They have a large worldwide network of manufacturing, test, and design facilities to support their high volumes of semiconductor production. They produce a wide range of products, including home consumer electronics, general purpose processors, embedded systems and more.

Currently Intel hand inspects all products for physical and cosmetic defects before being shipped to customers. This method of identifying defects is inconsistent due to the subjectivity of inspectors in identifying the defect. As the world’s largest semiconductor company, Intel is constantly striving for greater consistency in the quality of their products. This project will produce a prototype for a system to identify defects using an image capture system and detections software. The product is expected to detect at least 80% of defects in an image. Images of defective chips will be analyzed to identify the defects and compressed to a format which is a minimum of ten times smaller than the current format. The scope of the project consists of optimizing the illumination system for defect detection, developing software to detect defects on the chip die from an image, and implementing a method of defect image storage that is significantly more efficient than the current system.
2. System Requirements

This section describes our technical specifications and constraints. Table 1 lists the functional requirements and metrics with target values used to design our system. Our final system design was the design best suited to achieving these target values for our functional requirements. Table 2 lists the system level constraints given by the customer. Most of the constraints did not have to be designed towards specifically, because they came naturally from parts of the pre-existing design given to us. The only constraint we really had to design towards was the cost. The interfaces present in our design are listed in Table 3. The process variables overlap between some design parameters because some of our process variables are used to solve several problems at once, such as the conveyor causing the motion of the trays, the speed of the system, etc.

<table>
<thead>
<tr>
<th>Top Level Functional Requirement</th>
<th>Customer Importance</th>
<th>Critical To Quality (CTQ) (metric)</th>
<th>Target Value &amp; Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect 100% of Defects</td>
<td>5</td>
<td>% detection accuracy</td>
<td>80% necessary</td>
</tr>
<tr>
<td>Compress Images</td>
<td>5</td>
<td>Compression rate %</td>
<td>90% compression rate over JPEG target (10x)</td>
</tr>
<tr>
<td>Light image uniformly</td>
<td>4</td>
<td>% uniformity across image</td>
<td>Minimize light loss between middle and sides of tray (better than 80%)</td>
</tr>
<tr>
<td>Image a tray in 15 seconds</td>
<td>3</td>
<td>Seconds per tray</td>
<td>Target at least 15sec, less is better</td>
</tr>
<tr>
<td>Categorize defects</td>
<td>4</td>
<td>% defects categorized correctly</td>
<td>80% target, checked by visual inspection</td>
</tr>
</tbody>
</table>

Table 1: Top Level functional requirements for our design proposal and metrics / target values for each metric used to gauge system performance.
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software must run on Windows XP®.</td>
</tr>
<tr>
<td>2</td>
<td>Overall system cost must be less than $80,000.</td>
</tr>
<tr>
<td>3</td>
<td>Must operate on 110V power supply.</td>
</tr>
<tr>
<td>4</td>
<td>Must image standard size Intel die trays.</td>
</tr>
<tr>
<td>5</td>
<td>Must maintain 3” spacing between hazardous parts and operating area.</td>
</tr>
<tr>
<td>6</td>
<td>Must retain functionality of identifying serial numbers in images.</td>
</tr>
</tbody>
</table>

**Table 2:** System level constraints that must be adhered to in our design.

| Camera to Image Capture board interface (MDR26 Camera Link connector) |
|---|---|
| Image Capture board to PC interface (PCI-bus connection) |
| Detection Software to Compression Software interface (detection software gives defect pixel position, which is used by compression software) |
| Detection Software to Categorization Software interface (detection software gives defect position and characteristics to categorization software) |

**Table 3:** Interfaces present in our system design.
3. Optical System Design Concepts

3.1. Optical Concept 1: One camera, one wavelength, optimized positions

The first concept involved the use of a multispectral camera, or two cameras operating using different wavelengths of illuminations. This design concept would require two separate wavelengths of light, and only one pass through the system. The multispectral camera would capture one image with 3 values per pixel, corresponding to a red, green, and blue intensity level. Using this information, the red/green/blue components of the image could be examined separately to find the defects unique to each color image. Also, the illumination in each wavelength could be at different intensity levels, optimized to few different parts of the chip with better contrast. Using two grayscale cameras and one pass, the cameras would be inline with each other, and each would have a unique illumination wavelength and intensity. The two pictures taken by the two cameras would be analyzed for defects separately, with each wavelength/camera combo being optimized for a certain set of defects on the chips.

3.2. Optical Concept 2: One camera, two wavelengths, two passes

The second concept for illumination involved two separate wavelengths of light, and a single camera. In order to achieve optimal illumination of materials with different optical properties, two separate images would be taken and recombined into a single image. Each image would require its own pass under the camera, and illumination adjustments would be done in-between each pass. The first pass would be with one wavelength and the second pass would use the other wavelength. This option provides the most flexibility, but requires double the image capture time of a single pass, manual adjustment in-between passes, and increased software processing time. An optional method of achieving this flexibility would be to have a two-camera system with different illumination on each camera. This would eliminate the need for two separate passes and manual adjustment for each pass, but would substantially increase cost because the camera and LED arrays are large contributors to the overall cost of the system.

3.3. Optical Concept 3: One camera, one wavelength, two passes

This concept is very similar to the second concept, as it involves two passes through the system, with an adjustment made in-between passes. The difference is that the adjustment would be the orientation and intensity of the light source, rather than the wavelength of the source. This option also provides some flexibility, but it requires that all of the defects will be able to be imaged at one chosen wavelength. This concept would be relatively inexpensive compared to a two wavelength, two camera system, however it would be limited to detect defects that stand out at the particular wavelength chosen. Also, the adjustment made in-between each pass may increase the total time required to complete the imaging, and may not be a very convenient solution.
4. Detection Software Design Concepts

Two important factors of the projects are defect detection and defect categorization. These are both software based problems that have to be solved. After a tray is passed through the camera system an image is captured and detection and categorization of chip defects is done. Two approaches where chosen for the detection and categorization. Sum of difference and contour analysis of a threshold image

4.1 Detection Concept 1: Sum of Differences Method

The sum of differences method involves taking an ideal image and comparing that to the images taken by the system. From there an analysis of the difference between the two images is preformed. If an image differs in a certain location from the ideal image it is said to have a defective. This method could also give the system an easy way to categorize the defect type by comparing to the different type of defects and deciding which is more closely related to the defects on file. The accuracy sum of difference method is influenced largely by images with difference level of lighting and shifts in the image. For the sum of difference method to work best the images must first be normalized and scaled to ensure the images have average values which are similar. The next step which has to be done is to center the image to ensure the shift is done correctly. Once these alterations are preformed the pixels in the images are compared. The differences in pixels are analyzed to find if there is a specific location where a large number of differences are found. These locations will correspond to defects in the chips.

4.2 Detection Concept 2: Contour Analysis of a Threshold Image

The chosen method for implement is contour analysis of a threshold image. The contour analysis of a threshold image can be accomplished by running different processes on the images. Fig. 1 shows the flow of steps taken in the image processing stage to find the contours in an image. The image is normalized to ensure the grayscale values span the full range of 0-255. Normalizing helps more accurately calculate the threshold value of the image because it guarantees the image will have the largest contrast possible. The next part in the image processing is to section off the image. To accurate perform a threshold analysis the different parts of the chip have to be identified. Once the image is sectioned off the threshold can be found on just the die and using that value to convert the grayscale image into a binary image. All the values below the threshold value are changed to 0 and other values are converted to 1.
Figure 1: Flow chart of steps taken to find contours in an image.

The threshold implementation converts the grayscale image into a black and white. Though the conversion process is relatively simple, finding the threshold value is significantly more complicated. Otsu describes a mathematical process for finding the threshold value of an image in [2]. The Otsu threshold method for gray value images is the method chosen to threshold the images because of the success shown in the paper.

After the threshold image is created the salt and pepper noise is removed using a median filter described in [3]. The median filter reduces the noise and the chance that it could later be interpreted as very small defects in the image. After the image is in binary objects can be processed by finding contours in the image. Finding the contours involves looking at the binary image and locating enclosed areas. These could include the outline of the chip or defects that could be found in the chips. These contours are cataloged as a list of pixel values that make up the contour and this list can be interpreted as a function of x and y that can be used later to analyze the contours. The process of finding image contours is implemented in different ways by the system.

Now that the contours in the image are found an analysis that can be done on them is a finding the moment information of the contours. This step will not be implemented every time it will only be used to find the characteristics of the chip defect. The method to find the moment information is described in [4]. Some of the information that can be extracted using moment analysis includes mass, angle of orientation and center of mass. The mass can be used to identify
the chip size. The center of mass will indicate where the chip and the die are located in the image. The equations for moment extraction are:

\[ m_{p,q} = \iint x^p y^q f(x, y) dxdy \quad (4.1) \]

\[ \mu_{p,q} = \iint (x - x_c)^p (y - y_c)^q f(x, y) dxdy \quad (4.2) \]

\[ f(x, y) = \text{contour of image} \quad (4.3) \]

The calculation of the zero order moment gives the area of defect

\[ \text{Area} = m_{0,0} \quad (4.4) \]

A combination of the zero and first order gives the center of mass

\[ x_c = \frac{m_{1,0}}{m_{0,0}} \quad (4.5) \]

\[ y_c = \frac{m_{0,1}}{m_{0,0}} \quad (4.6) \]

Finally the angle of orientation is found using the second order moments

\[ \theta = \frac{1}{2} \arctan \frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}} \quad (4.7) \]
5. Compression Software Design Concepts

Another key requirement taken into account during the design of the system was compression. Specifically, the goal was to obtain a compression ratio over the current format by a factor of 10x. The size of the raw images captured were large, roughly 160MB in size. The archival of thousands of defective chip images becomes expensive without proper compression, making this an important point in the design’s functionality. This section will briefly discuss the design concepts that were considered.

5.1 Compression Concept 1: MPEG4™ Adaptation of Interprediction

The first method for compression was an adaptation of the MPEG4™ (Moving Pictures Expert Group) video format. This design concept was not used. Note that the author(s) have experience developing video decoders, so no source information was necessary regarding the MPEG4™ format.

MPEG4™ is a widely used video and audio format for compression. This adaptation of MPEG4™’s format to still image compression was to focus on the way MPEG4™ decodes video streams. Like other video formats, a frame in a video encoded in MPEG4™ uses a something called interprediction to accelerate the process of decoding and reduce file sizes. The complex algorithm that can estimate the Luma and Chroma values based on the pixel block values in previous frames would have been the target information to extract from the MPEG4™ format for use in this image compression.

Using the algorithm for prediction between frames in MPEG4™, a compression method for single images was formed. An image can be saved in terms of the differences between it and a similar image, with regards to this predictive scheme. However, this method was not chosen for several reasons. The complexity of the prediction algorithm was deemed problematic for a project with very limiting time constraints. More notably, this method of compression would introduce compatibility issues – without a special program that understood this new image format, the files would be not be viewable. Though the complexity of designing a viewer for the new file format would have been trivial, it was regarded as too inconvenient for the images to be archived in a non-standard image format.
5.2 Compression Concept 2: Custom Algorithm for Grayscale Image Compression

The second method for compression was a custom algorithm using existing JPEG™ (Joint Photographic Experts Group) compression for grayscale images. This design concept was used and expanded upon.

This compression design concept focuses on using grayscale JPEG™ compression, both lossless and lossy. Lossless JPEG™ formats lose no image quality in compression due to the use of a type of exponential-golomb encoding. The basic idea behind this encoding is that values that appear more frequently in an image are assigned very small binary code numbers. Less frequently occurring values within the image are encoded with larger binary values. This creates a system that guarantees moderate compression with no data loss. Lossy JPEG™ encodes images in such a way that small changes in color and brightness across the image are lost, but compresses the image much more efficiently (Wikipedia, “JPEG”).

Originally, the custom algorithm was to exploit the properties of JPEG™ by pre-processing the image in a manner that altered small blocks of pixels. The blocks of pixels would have their Luma (essentially brightness) values normalized and replaced in areas that were not considered defective. This assumed that the coordinates where defects had been detected were available from the defect detection software. The efficiency of this algorithm was crudely measured in the case study found in the critical design review and is included in Appendix D. While the compression results were significantly exceeding the 10x goal, the loss of information in image quality was noticeable. When real image samples from the system became available, this concept was revised slightly. It became unnecessary to normalize the Luma values and suffer noticeable data loss. The captured images easily met the 10x compression criteria by simply being converted from bitmap to JPEG. Due to the simplicity of this solution, the compression system would now have five different options for compression that offer different levels of quality and compression, though most meet or exceed the functional requirements. Section 6.3 discusses the specific options available when compressing the images.
6. Design Analysis

The overall system performance is determined by the individual performance of each of the subsystems. The ultimate purpose of the system is to identify defects and store images of the defects at high compression. The ability to identify defects depends on the illumination and the defect detection/categorization algorithms. The compression relies only on the compression algorithm/method. We have determined through analysis that our system design meets the requirements for detection, categorization, compression, and time. Our design of the illumination and camera system allows enough flexibility so that it may be optimized to acquire images that show the defects our sponsor is concerned about.

6.1. Hardware system analysis

6.1.1: Camera System Analysis:

The decisions for which camera setup to use were made based on availability of equipment from the client, and how well the system would meet the illumination and image capturing specifications. The main issue in the current system that we are trying to solve in this system is that objects of varying reflectivities need to be imaged together. This creates a problem because using a high illumination level necessary to image a low reflectivity surface would saturate the detector when imaging a high reflectivity surface. Conversely, using the lower illumination level necessary to image the high reflectivity surface would result in a lack of light on the low reflectivity surfaces. The three PDR designs were evaluated based on their ability to solve this problem, as well as their availability in mind. The three designs presented in the PDR are as follows:

1. Using a multispectral camera or multiple cameras with two different wavelengths of illumination.
2. Using a single camera but two passes per tray, with adjustable illumination intensity in between passes.
3. Using a single camera and a single pass, and optimizing the illumination angle and intensity to best image the range of chip types.

Design 3 was most similar to the client’s current design, and would retain the same problem of a lack of contrast when viewing chips with highly varying reflectivities. Design 3 was eliminated because of its inability to resolve the reflectivity requirement to a satisfactory level. Both designs 1 and 2 would offer a more robust solution to imaging objects of differing reflectivities. The multispectral camera that would be used in design 1 was only available in a 4k resolution version from the preferred vendor Dalsa, so two would need to be used to image the
whole width of the tray. This would add complexity because the images would have to be spliced together and there would need to be some pixel overlap between the two cameras to run a relative calibration between them. Also, the client currently does not have these cameras available to lend us, and their cost is outside our team’s budget. Using two 8k cameras in design 1 is feasible, but it is uncertain whether we will be able to obtain two image capture boards necessary to run both cameras at once, or whether we will be able to obtain two LED arrays of two different wavelengths. Our design is most similar to design 2, and will also optimize the working distance, intensity, and angle of the LED arrays to best image the whole range of object reflectivities that are required. A sample calculation demonstrating the need for a two wavelength / two camera or a two pass system in order to solve the reflectivity problem is given in Section 6.1.3.

6.1.2: Illumination System Analysis:

As described in the camera section 6.1.1, the 3 camera designs from the PDR also specified an accompanying illumination system. Design 1 used a set of 2-4 LEDs of 2 wavelengths with fixed positions, while design 2 used 2 sets of 2 wavelengths of LEDs with variable positions. Design 3 used only one wavelength of light, but also offered adjustment to the position to the LEDs in between passes. Design 2 was chosen for several reasons. The benefit of using two different wavelengths over one is that the detection algorithm gains extra data to work with, because some defects, specifically stains and foreign material, will show up better under one color illumination but not another. The benefit of using two sets of two LED arrays is that each set can have an incident intensity on the tray optimized for a different range of reflectivities, and an orientation angle optimized for a certain set of defect types and positions. It should be readily apparent that these benefits can also be seen in a system with only one set of two LEDs but with multiple scans, between which the parameters of the LEDs can be varied. Having two separate incident intensities on the chip during two different passes or under two different cameras, one intensity can be optimized for highly reflective sections of the chip while the other can be optimized for low reflectivity areas. By having the two sets of LED arrays at different incident angles with respect to the tray, you increase the ease in which certain defects can be detected. The main mode of identifying scratches / cracks or foreign material deposits is by the shadow it creates due to the interaction between the illumination angle and its orientation, so by using two separate incident angles, you increase the number of defect orientations that can be easily seen. There have been previous linescan system designs in literature which demonstrate the utility of using different illumination angles to view different defect types [1].

6.1.3: Analysis of Reflectivities using two separate sources

A sample calculation demonstrating the necessity of a two wavelength, 4 LED array system, or a 2 LED array two-pass system design is given in this section. The material properties or compositions of the objects being imaged have not yet been given to us, so I will assume reflectivities of .1 and .9 for low and high reflectivity materials respectively. A line scan speed of 5000 frames per second will be assumed as well (5kHz). The incident illumination levels are
taken from the Cobra linescan illuminator spec sheets. The solid angle is assumed to be 1.1E-3, which is found by assuming a linescan area of 10cm (length) x 1mm (width), and a camera working distance of 30cm. The actual solid angle will depend on the lens given to us with the camera and the working distance. Table 4 below lists the results of a sample radiometric calculation using two wavelengths of LED arrays and a high and low reflectivity sample. The blue LED is used to illuminate the high reflectivity sample because the blue LED naturally has a lower peak intensity and a lower responsivity in the camera. The fourth column of the table shows the result if the Red LED was also used for higher reflectivity sample. The disparity in DN values from imaging the low and high reflectivity object is quiet high, and would most likely cause issues in detecting defects. The noise level of the camera is about 15DN, and the camera will most likely be used with some amount of gain, which increases its responsivity by several times for a 10dB gain. This would cause the high reflectivity object to saturate the detector when using Red illumination, while it will still be imaged correctly when using the Blue illumination. The main equations necessary to perform these calculations are the radiometric equations listed below. Equation 6.1 shows how to get the reflected radiance L from the incident irradiance on the object E and the reflectivity of the object \( \rho \). Equation 6.2 shows how to get the incident irradiance on the optics of the camera system from the solid angle of the system \( \Omega \) and the radiance L. The irradiance at the optics, E, is in units of W/m^2, which can be converted into J/m^2 by dividing by the frame rate of the camera, 2.5kHz. From this point, it is just a simple matter of converting the units and multiplying by the camera response to get the system DN value produced.

\[
L_{\text{reflected}} = \rho \pi E_{\text{incident}} \tag{6.1}
\]

\[
E_{\text{optics}} = \Omega_{\text{optics}} L_{\text{reflected}} \tag{6.2}
\]
<table>
<thead>
<tr>
<th>LED arrays used</th>
<th>Blue</th>
<th>Red</th>
<th>Red</th>
</tr>
</thead>
<tbody>
<tr>
<td>wavelength (nm)</td>
<td>470</td>
<td>630</td>
<td>630</td>
</tr>
<tr>
<td>spectral width (nm)</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Incident Irradiance at 0° (W/m^2)</td>
<td>750</td>
<td>1100</td>
<td>1100</td>
</tr>
<tr>
<td>Incident Irradiance at 45° (W/m^2)</td>
<td>530.3300859</td>
<td>777.8175</td>
<td>777.8175</td>
</tr>
<tr>
<td>reflectivity ρ</td>
<td>0.9</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>reflected radiance from object (W/m^2sr)</td>
<td>151.9283784</td>
<td>24.7587</td>
<td>222.8283</td>
</tr>
<tr>
<td>line scan rate / frame rate (kHz)</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>per-frame energy radiance into detector (J/m^2sr)</td>
<td>0.060771351</td>
<td>0.009903</td>
<td>0.089131</td>
</tr>
<tr>
<td>responsivity at wavelength (DN/(nJ/cm^2))</td>
<td>10.5</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Assumed solid angle of system (sr)</td>
<td>1.11E-03</td>
<td>1.11E-03</td>
<td>1.11E-03</td>
</tr>
<tr>
<td>per frame energy flux (nJ/cm^2)</td>
<td>6.745619999</td>
<td>1.099286</td>
<td>9.893576</td>
</tr>
<tr>
<td>scale factor for 2 LEDs of each used</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>DN at detector</td>
<td>141.65802</td>
<td>30.78001</td>
<td>277.0201</td>
</tr>
</tbody>
</table>

Table 4: Sample calculations for DN registered by the system for the case of two different LED arrays used for a high and a low reflectivity object. The last two columns show the case of Red illumination used for both high and low illumination.

6.2. Detection software system analysis

The method chosen for the defect detection was the second discussed in second 4 the contour analysis of a threshold image. One problem with the first method was the lack of ideal images and ideal test units. Due to company policies it is not possible to gain access to non-defective dies. The tests done on some defective images showed that the contour analysis of a threshold image is more robust on a variety of different images. The contour analysis of a threshold image preformed better on images with different levels of lighting because the algorithm was less susceptible to changes in lighting. This method also performed better with shifted images because an exact lineup is not necessary in the second method. When the image is lighter as long as it is not completely washed out the defect was more visible. The method was also tested on images that were darker as long as they were not completely void of light. The defects in the darker images were also seen more clearly with the second method. The following parts of this section go thought the way in which method two is implemented in subsections of defect detection.
For defect detection the system has to read the ID matrix this is accomplished using the image capture software used in the current system and no alterations to that application will be done. The next part of defect detection is finding the chip size this is done finding the contour whose properties match most closely to the average chip size. The final part of the defect detection is finding defects on the chip. This part also implements the contour analysis of a threshold image. The system will take the image of the chip and extract the contours using the method described. The image will be sectioned off to focus on the die for the defect detection. If many contours are found on the chip it means there are defects because a non defective chip will have either no contours or very minimal contours within the chip. This can be seen more clearly in Fig. 2.

![Image of chip with and without defect](image.png)

*Figure 2: Threshold of image with defect and the Threshold of image without defect.*

The threshold image without a defect has no contours in the center. The image with a defect has a contour that will be found on the chip. With this strategy it is easy to figure out which chips have defects and which do not. This method was chosen over the sum of difference method because after testing the threshold value proved to be more reliable. The sum of difference looked at the difference between a non-defective chip and compared it to the current chip from there it decides if there are defects in the chip. This method failed when one image was darker then the image they are comparing to and it failed when the image was shifted. This happened because these slight alterations resulted in many difference and evaluated to being defective even if they were not. Because the threshold method does not compare with other images to find if it is defective this is not an issue with that method.
6.3 Compression Software System Analysis

The compression software system is an implementation of compression concept 2, found in section 5.2.

The compression software was implemented in Matlab® using the Matlab® Image Processing Toolbox. The code for the software is available in the accompany CD. The software allows the user 3 different choices of compression with differences in quality loss, compression ratio, and standard view-ability. Table 5 lists the features of each compression mode.

<table>
<thead>
<tr>
<th>Compression Mode</th>
<th>Compression (Approximate)</th>
<th>Quality Loss</th>
<th>Requires Defect Viewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-JPEG</td>
<td>20x</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LL-JPEG</td>
<td>2x</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>L-Gold</td>
<td>100x</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LL-Gold</td>
<td>15x</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>H-JPEG</td>
<td>10x</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 5: Compression Modes

The compression software runs immediately after the defect detection algorithms finish. The coordinates on the image where the defects exist are passed to the compression software. In all compression cases, a text file with the defect categorization and defect location are saved with the compressed image in the same directory. This information is utilized differently depending on the compression mode chosen. All modes of compression are efficient with regards to time which is discussed in greater detail in section 10.3.

6.3.1 L-JPEG and LL-JPEG (Lossy and Lossless)

The L-JPEG and LL-JPEG compression modes are the simplest formats to produce. L-JPEG saves the image in lossy grayscale format with a small loss of quality. The JPEG image that is saved is completely viewable in any operating system that supports standard JPEG. The LL-JPEG saves the image in a completely lossless fashion with no loss of quality. Image processing and editing programs that support lossless JPEG can open this file, although many do not. It is advised to use the developed software to view the images. L-JPEG achieves roughly 20x compression over the bitmap captured image, while the LL-JPEG produces a lackluster, though lossless, 2x compression. Example images of L-JPEG and LL-JPEG can be seen in
6.3.2 L-Gold and LL-Gold (Lossy and Lossless)

The L-Gold and LL-Gold compression are not included in the final implementation, but can be easily added functionality in later versions. The idea behind the “Gold” mode is to use a single, non-defective, “gold standard” image of each type of chip. We were unable to receive an image of a non-defective chip to implement these modes. Essentially a non-defective pristine image of each type of chip will be saved as a master image. When chips of a certain type are imaged and tested for defects, the portions of the image considered defective are saved to disk in the directory with the corresponding gold image. L-Gold saves the defects in lossy JPEG, while LL-Gold saves the defects in lossless JPEG. When being viewed, the master image of the chip type is shown with the defective image positioned accordingly based on where it was found. The overhead of storing a single master image for each type of chip becomes negligible as the number of chips imaged becomes large. The 2D matrix “barcode” that identifies a chip is also saved for each chip imaged along with the defects. Note that this method of compression suffers little to no quality loss, but the information lost may be critical should the non-defective portions of the chips become important. L-Gold and LL-Gold modes are suggested for very high compression when it is certain that only the defect portions of the chips are important. Example images of the Gold modes can be seen in Appendix E.

6.3.3 H-JPEG (Hybrid)

The H-JPEG (“Hybrid-JPEG”) compression mode makes use of the assumption that the defective portion of the image is the most important, while the non-defective regions may still contain relevant information. The portion of the chip that has been identified as defective is saved in lossless grayscale JPEG. The rest of the image is saved in the lossy JPEG format, reducing image quality but still retaining characteristics to the specific sample (whereas L-Gold and LL-Gold would use a master image instead of the original). On average this achieves the goal of 10x compression, though it is dependent on the size of the defects. Should a very large region of the chip be considered defective, the compression will be closer to 2x, while a very small defect could yield up to 20x compression. This compression mode is therefore suggested...
to be used most frequently as it combines the best tradeoffs between relevant quality and overall compression. Example images of the H-JPEG mode can be seen in Appendix E.
7. Proposed Design

Our proposed design comes mainly from optical concept two. Since such a large part of our project is the optimization of our design, we decided to choose a design that provided the most flexibility. Through research, analysis, and conversations with industry experts, we felt that in order to make a significant improvement over the current system, we would need to use two different wavelengths. We were provided with four different wavelengths of LED arrays, so we went with a concept that would allow the use of any combination of these sources that we find useful. Our chosen design is a multiple wavelength system with the option of two imaging passes, each with their own illumination configurations. Extensive testing and optimization of all the wavelengths and orientations (angles and distances of sources) will provide us with the final chosen configuration. Our goal is to find through testing, a single configuration that will image all types of samples and defects with a single illumination configuration. If this is possible, then there will not be a need for a dual-pass or dual-camera system. Our design would also be considered a success if we found through testing that multiple passes and or multiple wavelengths would be needed to image different types of defects. In the event that there is no single illumination configuration that works for all defects and chip types, this proposed design gives us the flexibility we need to deliver testing results for every wavelength and configuration tested.
8. System Build

8.1. Hardware Mounting:

With a design chosen for its flexibility, we needed to fabricate mounting hardware that would allow for a wide range of illumination configurations. We were given a Flexlink conveyor system, and some spare mounting hardware from the original system designed by Intel. We needed to make modifications and additions to the provided equipment that would allow for a variety of repeatable and measureable configurations. We designed and fabricated mounts for the LEDs that could be adjusted independent of each other and the camera, and we made a camera mounting plate to fit the 8K high speed camera that has a heat-sink where the previous mount would have attached.

![Flexlink conveyor](image)

**Figure 3:** Flexlink conveyor. Aluminum frame with T-slots allows mounting of components using metric bolts and square nuts provided by Flexlink.
Camera mounting plate. **A**: The mounting plate provided by Intel, designed for the lower resolution camera that does not have the extra heat-sink on the bottom of the camera body. **B**: A drawing of the new camera plate which we machined from a 3/8”x4”x6” plate of aluminum. As shown in the picture, the 3” diameter camera tube on the 8K high speed camera fits through the hole in the plate, and attaches with four M3 bolts.
We fabricated an aluminum mount to attach the LED arrays to the conveyor frame. We started with a ½”x3”x18” aluminum piece, and made modifications to it using a mill and a drill press. We fabricated two of these components; one for each LED array. The completed mounts allowed us to move the LEDs individually, and separate from the camera, and provided four degrees of freedom. 

A: Two slots were milled in each piece, these slots allowed for a stable and repeatable union of the mount and the frame of the conveyor using four M8 bolts. Fine height adjustments could be made throughout the length of the slots.

B: To extend the height range of the LEDs, we drilled holes to allow for three different position of the rotation mount.

C: A rotational mount was given to us by Intel. This mount allows for a wide range of angular and has holes positioned so the older style Cobra LEDs can bolt directly to it.

D: In order to adapt the rotational mount to fit the wider hole pattern on the new model Cobra Slim LEDs, we fabricated an adapter made from angle aluminum.

8.2. Hardware Calibration

There were three important parameters that we needed to calibrate in order to obtain an image that was dimensionally accurate. First we needed to orient the camera so the CCD array was perpendicular to the motion of the conveyor; if this was not done correctly then the image would be skewed. We calibrated this by using focus mode in the provided camera software, and making small camera rotations until an image of a ruler was not skewed. Secondly we needed to set the frame acquisition start time. We placed an optical sensor beneath the path of the tray on
the conveyor that would trigger the camera to start acquiring frames when the tray approached the camera. We had to adjust the position of the sensor to optimize the camera start time so that the whole tray would be imaged, without any unnecessary frames. Lastly we had to adjust the conveyor speed to match the frame speed of the camera; if this was not correctly set, then the image would be stretched or shrunken in the direction of the conveyor motion. We used a trial and error process to do this. We first took an image of a tray of square chips, and adjusted the speed until the image of the chip looked fairly square. Next we measured the image by counting the pixels in the x and y directions. We would know that are speed was correct if the two values matched, since the speed adjustment only affects the length in the direction of the conveyor motion. We made fine adjustments to the speed until the dimensions in each direction matched. After this, our system was ready for the testing of different lighting configurations.

8.3. Software Algorithms

The software for the image capturing and identification matrix reading has been provided by Intel. No alterations were made to this application or the running of the application. The defect detection and compression will be written by the team. To implement the methods discussed for detection and compression Matlab® will be used. Matlab® is the chosen application because it provides an image processing toolbox which has many of the functions discussed. One of these includes finding edges which is the same as finding the contours of an image. Matlab® graphical tools also provided a great means of creating a Graphics User Interface (GUI) for the project.

For the detection the image toolbox Matlab® provides is used heavily. Use of the Graphics User Interface (GUI) discussed in the following section an image can be selected to have a defect analysis done on. After the image is read in and image erosion is preformed using the image erosion function provide by the image processing toolbox. The erosion partitions the image into very small uniform sectioning. This erases the small details in the image and creates a simple version of the chip outline. The threshold level, creating a threshold image and finding the edges are all preformed by implement functions as well. These edges are then analyzed to determine the location of the die, the chip outline, and the two labels in each image. Finally a cropping can be done on the image to remove the die and analyze that part of the image separately to find defects. Full code for the Matlab® implementation of the defect detection can be found in Appendix C.

8.4. GUI (Graphics User Interface) Guide

The GUI included with the software package was designed to be functional yet easy to use. A picture of the GUI is shown in Fig. 6. The GUI allows images of defective chips to be analyzed and saved in specific compression formats. The GUI also reconstructs images saved non-standard formats, such as the H-JPEG discussed in section 6.
Loading an image into the GUI is simple and straightforward. For loading standard image types, like bitmap or jpeg, select “L-JPEG” as the compression method. Type the file name of the image that you would like to load and view. Click the “Load Image” button and the chip image will be brought up in another window. From here the image can be navigated by using the zoom in, zoom out, and move buttons, along with all of the other standard features Matlab® image viewer provides.

After an image is successfully loaded, the defect analysis can take place. Click the “Run Defect Analysis” button to begin the defect detection. When the detection algorithm finishes, the chip image will be displayed with defective portions of the chip highlighted in green. Note that this does not alter the image, it is only to show the user what was found as defective.

Once the defect analysis is complete, the chip image is ready to be stored. Select either H-JPEG, LL-JPEG, or L-JPEG as the compression method. Specify the filename you would like the compressed image to have in the ‘Output Filename” box. Click the “Saved Compress” button to complete the process, successfully saving an image in the specified format and name. Note also that the quality of H-JPEG and L-JPEG can be adjusted – the higher the number, the higher the image quality but the lower the compression.

![Graphical User Interface](image)

**Figure 6:** The Graphical User Interface for processing captured chip images.
9. Illumination Optimization

A large portion of our design work is in optimizing the illumination conditions of the system to increase the quality of images captured. This increase in quality is measured by the amount and types of defects that can be detected in the images, and by the contrast between different portions of the images. The amount and types of defects can be visually seen in the images, while the contrast can be measured between different portions of the images using image processing software such as Matlab®.

The optimization plan is dependent on the number and types of LEDs we were provided with the system to test. The LEDs provided to us by Intel include two 5” red Cobra® LEDs, and one each of 5” blue, ultraviolet (UV), and infrared (IR) Cobra Slim® LEDs. All the LEDs used are manufactured by Stocker-Yale. The LEDs used in the optimization and some of their important specifications are shown in Table 6.

<table>
<thead>
<tr>
<th>LED #</th>
<th>LED Name</th>
<th>Peak Wavelength (nm)</th>
<th>Max Irradiance (W/m²)</th>
<th>Working Distance (mm)</th>
<th>Controllable Intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red</td>
<td>630</td>
<td>1100</td>
<td>40</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Red</td>
<td>630</td>
<td>1100</td>
<td>40</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Slim UV</td>
<td>395</td>
<td>400</td>
<td>40</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Slim Blue</td>
<td>470</td>
<td>1500</td>
<td>40</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Slim IR</td>
<td>740</td>
<td>2200</td>
<td>40</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Note the Slim LEDs irradiance is taken as double the respective original Cobra® irradiance as it is quoted in the Slim tech sheet.*

Table 6: LEDs used in optimization process.

The system parameters that can be varied in the illumination system include the number and wavelength of the LEDs used, the working distance and angle of each LED, the intensity of the Cobra Slim LEDs, and the aperture setting of the camera. Changing the working distance of the LEDs varies both the intensity of the LEDs and the line width of the focused illumination line incident on the tray. In our tests we want to keep the working distance constant at a value that approximately produces the most focused line, which is somewhere around 70mm (close to 3”) for the original Cobra LEDs. By keeping the working distance constant, we have one less variable to change, which makes our optimization space a little more manageable. Any variations in the working distance in our test data is due to the limited range of our mounts. Sometimes the working distance requirement had to be loosened a bit in order to test extreme angles. All this
information is noted in the test files. The intensity of the LEDs can still be varied via control cable for the Slim models, and the constant intensity of the original Cobra® red LEDs can be compensated for by adjusting the aperture. Our system is ideally setup for 2 LEDs to be used in the illumination system, so most tests will be done with a 2 LED setup. The illumination angle of incidence can be adjusted on our mounts through a range of about 20 to 73 degrees. It is impossible to get a perfect range of 0 to 90 degrees, as on either extreme the LED mounts would collide with either the conveyor belt and tray, or the path of the camera. The variables for each test setup will then be the wavelength of the sources, the angle of each source, and the intensity of the source (for the Slim models) or aperture stop setting for combinations with the original Cobra® models.

The amount of parameters that can be varied in the illumination system coupled with the time it takes to capture and view each image is daunting, so we chose to initially keep our optimization space limited, and then expand it in stages. Stage 1 of the optimization plan involved testing each of the 4 unique LEDs individually on the right side of the camera system. In this stage, the working distance of the LED was held at a constant 3” from the focus line on the tray, and the angle of incidence of the LED was adjusted through its full range in steps of approximately 5 degrees. Again, the angle of illumination is difficult to change in exact increments due to the mounts, so the step size is only approximate. Also, if a certain low or high extreme angle was showing a notable decrease in quality, then there was no reason to continue decreasing or increasing the angle further. The intensity of the LED will be predetermined through some initial trial runs so that the area of interest in the image is as bright as possible while still remaining unsaturated. This stage of the optimization consists of about 6-8 images captured per LED, so about 20 images total. The results of this optimization were used to give us a good idea of which angle performs well for viewing different defect types with different wavelengths. It also showed us which wavelengths are better at detecting different defect types. As a general trend, we found that the higher angles tested produced clearer images. The results of this first stage of the optimization were that the best angle for blue was 60 degree, red was 47.5 degrees, and IR was 55 degree. For blue, this was the highest angle tested, and for IR and red, this was the second highest angle tested.

Stage 2 of the optimization was to then optimize the system with sets of 2 LEDs. The pairs of LEDs used in stage 2 were determined based on the individual performance of those LEDs in stage one, and based on what we had available. The pair of 2 red LEDs was tested in this stage of the optimization, as that is what is currently used in the system at Intel. The results from the 2 red LED tests were used as a benchmark to compare our future results with. The point of stage 2 is to expand the optimization space to include combinations of 2 LEDs, which is what our final design relies on. In this stage, the working distance of the 2 LEDs is again held constant. The angle of incidence of both LEDs were varied throughout there whole ranges of motion. The ‘best’ angles obtained for stage one were used as a starting point to vary the angles around. Many pairs of high angles were used, as high angles were generally better than lower angles. We had
originally hoped to vary both LEDs through their full ranges of angles, again in steps of 5 degrees. This proved to be impossible with the amount of time we had, so we instead choose around 12-15 images for each pair of illumination wavelengths, and varied the left and right angle throughout their ranges giving precedence to the ‘best’ angles first. The aperture setting was varied between different images in order to keep the image as bright as possible without saturating the fiducial markings on the chip. For tests with a Cobra Slim® and Cobra® LED, we used the intensity control of the Cobra Slim® to reduce its intensity to about half, to keep it comparable to the red Cobra® LED. In this stage of the optimization we captured about 12-15 images per setting, and we tested 4 main settings. This resulted in a total of about 56 images captured. This may not seem like a lot, but each image could take over 15 minutes to set up correctly and change the aperture setting if necessary. The system also had to be refocused between some tests, because keeping best focus was very important. Even with our efforts to refocus the system, some images captured appear to be slightly out of focus. This will reduce the number of defects detected on that image slightly, but they should still be usable as comparisons to the other images.

The four main setups used in stage 2 tests were a red-red combinations, blue-red, IR-red, and IR-blue. The combination of IR-UV was also briefly tested, but poor results led to that test being stopped prematurely. The UV light was found to have such a low irradiance output compared to the others that it could not illuminate the images enough to be seen by the camera. This is also due to the cameras responsivity being extremely low in the UV. The tray of chips used in the tests was arranged to have the bottom row of 8 chips contains the range of defects we were trying to detect. One such chip was singled out as having all defects present on it. We used this chip as our ‘test’ chip, which served as the basis for comparison between the different illumination setups. Images of this single chip were viewed for each angular setting for a given wavelength pair. The best image of the bunch was selected from visual inspection of the key defect features. The settings for this image were the ‘best’ settings for that wavelength pair. The results of the optimization process, along with images taken and analyzed using the defect detection software are given in the Design Results section.

During the optimization process, it was noticed that higher intensity often corresponded to more visible defects on the die portion of the chip. This came at the expense of washing out other portions of the chip. Since our design concept was a multi pass system, it is within the scope of our design to have one pass using settings that completely wash out the chip except for the die, and use the other pass to capture the substrate part of the chip in greater detail. To see if this was a viable option, we decided to test some ‘over saturation’ images with a few combinations of LEDs. These over saturation tests were not as in depth as the normal tests, and only used a few pair of high angles, since high angles generally led to better view ability of the die. The over saturation tests were done for IR and blue, IR and red, and IR and UV. One such over saturation picture is shown in Fig. 7. In each of these over saturation tests, the aperture setting was reduced, increasing the aperture size until the 2D data matrix could just barely be
accurately read in to identify each chip designation. You can see in the image that nearly every part of the image that is not flat black shows up as white.

Figure 7: Image showing over saturation using IR-red combination of wavelengths

Overall, the optimization process took about 3 weeks from start to finish, beginning on April 4th, 2009, which was the day all equipment was setup. In total, around 100 good images were taken with different settings. Hundreds more poor quality images were taken and discarded while trying to achieve usable images. Each of these images consists of at least 8 good chip images, and each of these images must be manually checked for quality and defects visible. The settings for each good image captured during the optimization process are detailed in Appendix F.
10. System Testing

The Illumination Settings and the Detection Software is dependent on each other, without a uniform light intensity, the image captured by the line scan camera will not be good enough for the detection software to process. Since a different combination of light intensity and the angle of the LED array were adjusted during the setup, the illumination is good enough for the detection software. As for the compression, the result is very impressive and had better compression ratio than was required.

10.1. Illumination Settings Tests

The tests of the ‘best’ illumination settings found, and of the camera and illumination system in general can both be completed together. As a result of the illumination optimization, we have a set of settings for which LEDs to use, their angular position, their working distance, and the aperture setting. The test to see if each illumination setting is viable is that there is no over saturation of the fiducial markings (the dots and triangle on the corners of the chips). We made sure of this during the optimization process, so this test was passed while optimizing the system. Also, the defects that these settings are optimized for must be clearly visible by examining the image by hand, and should also be picked up by the detection software (which is dependent on how well the detection software works at the time of testing). The image must also not be under or over saturated in the region of interest with defects on the image. The image must not be obstructed by the LEDs position in any way. The uniformity requirement of the image has been thrown out because the problem has already been fixed in Intel’s system, by using longer LED arrays that we were not provided. All of these tests except for the defect visibility test were passed automatically during the optimization, as settings which did not pass these tests were thrown out from the optimization.

To pass the ‘detectability’ test, we visually examined a single test chip containing all relevant defects under the ‘best’ illumination settings of each wavelength pair. The test here was a comparison test between the different illumination pairs. Since we did not have a picture of each of the chips taken by Intel’s system, we could only compare our images to themselves and to how the physical chip looks when visually inspected. We found that while there were some variations in which defects the wavelength pairs detected better or worse, they all seemed to perform fairly similarly. The defect detection software, which detects defects on the die of the chip, was able to work and detect defects on all chip images taken, not just those that corresponded to the ‘best’ settings. This means that the optimized illumination settings passed the defect detection test, as well as all the other tests outlined in the prior paragraph. The specific strengths and weaknesses for each wavelength combination is outlined in the Design Results section later on.
10.2. Detection Software Tests

The functional requirements for detection were to find the size of the chip, the identity matrix and locate defects in the chip. The identity matrix is read by the current system so no testing is required for that part. The software checks for defects within the die and also finds the outline of the chip and locates the labels on the chip. Figure 8 shows the output of the detection software.

![Figure 8: Output image of defect detection](image)

To test the accuracy of the software the results of 30 different images will be analyzed. The different checks to test accuracy will finding the number of defect in the die and compare to see how many defects were not located. This will give the accuracy level of finding all defects within the die of the chip image. The accuracy of the sectioning can also be tested. This will be done in the same analysis of 35 images. For each image the sectioning in of the outline and the labels will be examined to find the pixel error the detection software had for that particular image. Finally the robustness of the detection program will be tested by running the detection program on a minimum of 70 images and 5 different lighting types to check for proper functionality with different types of inputs.

To retain compatibility with the hardware all the defect detection software has to do is run on the current PC. This means that no significant delay is detected and when the software runs on the hardware. For the reading of the identification matrix a check just has to be done after the processing to make sure the current software provided is accurate with the new camera. Because the size will be provided by the user when the chip type is selected no testing has to be done on the chip size and it can be assumed that the user will select the appropriate type and the size specification for that type is correct.
10.3 Compression Software Testing

The compression software was tested by simply running the software using different compression modes as recording the file sizes ratios. The tests were run on the sample image seen in Appendix E. The tests considered the upper-left 1000x1000 pixels to be defective, which was considered a generous over-estimate of how much of the image would be defective in practice. The results of the tests are contained in Table 7. Note that the Gold compression modes require an overhead of about 2500KB per type of chip. For example, more accurate portrayal of L-Gold might be $\frac{5811000\text{KB}}{47000\text{KB} + 2500\text{KB}}$ for a sample of 1000 chips of the same type. This makes the L-Gold compression ratio about $118x$ rather than $123.64x$, though asymptotically it tends towards $123.64$ as the number of chips scanned approaches large numbers.

<table>
<thead>
<tr>
<th>Compression Mode</th>
<th>Original File Size</th>
<th>New File Size</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-JPEG</td>
<td>5811KB</td>
<td>257KB</td>
<td>22.61x</td>
</tr>
<tr>
<td>LL-JPEG</td>
<td>5811KB</td>
<td>2425KB</td>
<td>2.40x</td>
</tr>
<tr>
<td>L-Gold</td>
<td>5811KB</td>
<td>47KB</td>
<td>123.64x</td>
</tr>
<tr>
<td>LL-Gold</td>
<td>5811KB</td>
<td>422KB</td>
<td>13.77x</td>
</tr>
<tr>
<td>H-JPEG</td>
<td>5811KB</td>
<td>644KB</td>
<td>9.02x</td>
</tr>
</tbody>
</table>

Table 7: Results of testing a 5811KB captured image with different compression modes.
11. Design Results

11.1 Optimized Illumination Conditions

Several key results were discovered when examining the images captured under the optimized illumination settings. These results are outlined below, along with a table showing the best illumination conditions for each wavelength pair, and flagged areas of each chip’s die using the defect detection software. A single chip, serial number 2T716031B 0212, was used primarily when analyzing the data. This chip contained multiple cracks and scratches on the die as well as lots of foreign material. It also contained one large stain on the substrate, and lots of foreign material on the substrate. It was chosen as the test image because it contained every defect type we were interested in. All contrasts given in the results below are Michelson contrasts taken in 11x11 pixel averages around the areas of interest. Michelson contrast is the difference between the contrasts over the sum. Table 8 compares the results from the defect detection software on the test chip under the best illumination settings for each wavelength pair. The number of detected contours on the die refers to the number of edges flagged. This number is not a reliable metric to go by, since a chip that has less defects visible could have a higher number of edges present. This would be the case if a long scratch was not visible in some areas of the chip, so it would be flagged as several separate smaller edges, inflating the number of detected contours. A more reliable metric is the number of green pixels on the detected image. These green pixels outline each defect, so a chip with more defects present will have more green pixels.

<table>
<thead>
<tr>
<th>Wavelength Pair</th>
<th>Detected contours on die</th>
<th>Detected green pixels on die</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red-Red</td>
<td>2084</td>
<td>29699</td>
</tr>
<tr>
<td>Blue-Red</td>
<td>1278</td>
<td>11806</td>
</tr>
<tr>
<td>IR-Blue</td>
<td>976</td>
<td>12341</td>
</tr>
<tr>
<td>IR-Red</td>
<td>1096</td>
<td>20141</td>
</tr>
</tbody>
</table>

Table 8: Defect detection software results on the test chip using the best settings for each wavelength pair.

Red-red illumination was tested with 11 images with angles ranging from 45 to 70 degrees. The working distance was fixed at 3” except for angles above 60 degrees, which required a larger working distance. The aperture stop was set at 3.5 for all the images except the 45-45 degree image, which used a setting of 2.5. The best illumination angles for this configuration were found to be 70-60 degrees for the left and right LEDs, with the working distance at 3.75” right side LED. Fig. 9 shows the test chip under these illumination conditions, with certain areas flagged and enlarged. The green covered die image shows the defect detected die for this chip.
The upper right flagged area shows a stain on the die, which is not really visible at all with these illumination settings. The lower right flagged area shows the boundary between epoxy and substrate. While the boundary itself is somewhat clear, the contrast between the epoxy area and the substrate around it is very low (1.6%). The upper left flagged area shows a spec of dust on the substrate. This configuration can see this foreign material very well, with a contrast of 23%. The defect detected die image shows lots of green flagged contours which correspond to cracks, scratches, and foreign material on the die. This configuration had the best detection of die defects, as evidenced in Table 8.

Blue-red illumination was tested with 12 images with angles ranging from 37 to 72 degrees. The working distance was fixed at 3” except for angles above 50 degrees with the right red LED, which required a larger working distance. The aperture stop ranged from 4 at the lower angles to 4.75 at higher angles. The best illumination angles for this configuration were found to be 50-50 degrees for the left and right LEDs, with the working distance at 3” for each and an aperture stop setting of 4. Fig. 10 shows the test chip under these illumination conditions, with certain areas flagged and enlarged. The green covered die image shows the defect detected die for this chip. The stain in the upper right flagged area is only clearly visible with this configuration. The boundary of the epoxy area is again somewhat clear, though the contrast between the epoxy area and the substrate around it is still rather low (11%). The speck of dust in the upper left flagged area is very difficult to see in this configuration, with a contrast of only 6.25%. This configuration was not as good at imaging die defects, as evidenced in Table 8. The other main
difference in this configuration is the contrast difference in the leads of the substrate when using blue. While this does accentuate the electrical leads in the chip, there increased brightness masks the brightness that would show up from foreign material like dust in those areas.

![Image](image_url)

**Figure 10:** Blue-Red illumination with 50-50 degree angles and 3” working distances.

IR-red illumination was tested with 11 images with angles ranging from 39 to 72 degrees. The working distance was fixed at 3” for the left LED and ranged from 3” to 5” on the right red LED. The aperture stop ranged from 4 at the lower angles to 4.5 at higher angles. The best illumination angles for this configuration were found to be 72-61 degrees for the left and right LEDs, with the working distance at 3” for the left and 5” for the right, and an aperture stop setting of 4.5. Fig. 11 shows the test chip under these illumination conditions, with certain areas flagged and enlarged. The green covered die image shows the defect detected die for this chip. The stain in the upper right flagged area is again not visible at all in this configuration. The boundary of the epoxy area is again very clear, and the contrast between the epoxy area and the substrate around it is much higher at 31%. The speck of dust in the upper left flagged area is still rather difficult to see in this configuration, with a contrast of 19%. Although the contrast is high, the speck detected is smaller than on the red-red configuration. This configuration was second best at imaging die defects, as evidenced in Table 8.
IR-blue illumination was tested with 15 images with angles ranging from 29 to 73 degrees. The working distance ranged from 2.5” to 4.5” throughout the tests. The aperture stop ranged from 4 at the lower angles to 5 at higher angles. The best illumination angles for this configuration were found to be 73-54 degrees for the left and right LEDs, with the working distance at 3.25” for each and an aperture stop setting of 5. Fig. 12 shows the test chip under these illumination conditions, with certain areas flagged and enlarged. The green covered die image shows the defect detected die for this chip. The stain in the upper right flagged area is again not clearly visible. The boundary of the epoxy area is the most prominent in this setup, and the contrast between the epoxy area and the substrate around it is the highest at 31.1%. The speck of dust in the upper left flagged area is again difficult to see in this configuration, with a contrast of only 14.6%. Though the contrast is higher, using the blue LED in the illumination makes all the leads appear as white so a detection algorithm would have a harder time differentiating the foreign material from the leads. This configuration was not one of the better ones at imaging defects, as evidenced in Table 8. This configuration shares the quality of having bright leads on the substrate from the blue LED, and having the epoxy appear very clearly from the IR LED.
In conclusion, the best setup would include multiple wavelengths. Table 9 details the best settings and relative strengths of each illumination pair tested. The angle of orientation did not make as large a difference as aperture setting or wavelength did. In general, the larger angles were better for detecting defects. Having too shallow of an angle would create saturated portions of the image around the areas of different height, like the die. This could hinder the software detection algorithm, so it is suggested to avoid extremely shallow angles. The red-red setup was clearly the best setup for viewing die defects, but it cannot see all stains on the surface of the chip. Blue mixed with anything else produced very clear images of the substrate, since the substrate leads were much more reflective in the blue/green wavelength region. While this does produce an interesting image, it would need to be combined with other wavelengths to be able to find defects on the substrate image. Blue also was the only wavelength that could see a stain on the surface of the substrate. It is interesting to note here than all of the wavelengths could see this stain when only one LED was used, so when the intensity was very low. The problem with keeping the intensity low (or the aperture setting high) is that defects on the die like cracks and scratches are not as visible. The blue-red combo was the only combination to see the stain while also having the intensity high enough to detect many die defects. Infrared allowed the best viewing of the epoxy. While detecting errors in the epoxy was not within the scope of our detection software, it could be useful for Intel to be able to differentiate where the epoxy is on a chip because too much or too little epoxy is considered a defect of the chip.

Figure 12: IR-Blue illumination with 73-54 degree angles and 3.25” working distances.
<table>
<thead>
<tr>
<th>Left Wavelength</th>
<th>Right Wavelength</th>
<th>Left Angle (°)</th>
<th>Right Angle (°)</th>
<th>Left WD (°)</th>
<th>Right WD (°)</th>
<th>Aperture Setting</th>
<th>Strengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>Red</td>
<td>70</td>
<td>60</td>
<td>3</td>
<td>3.75</td>
<td>3.5</td>
<td>die defects, material on substrate</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>50</td>
<td>50</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>stains on substrate, substrate leads</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>72</td>
<td>61</td>
<td>3</td>
<td>5</td>
<td>4.5</td>
<td>die defects, epoxy</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>73</td>
<td>54</td>
<td>3.25</td>
<td>3.25</td>
<td>5</td>
<td>epoxy, substrate viewing</td>
</tr>
</tbody>
</table>

Table 9: Best illumination settings and relative strengths of each wavelength pair tested.

It would be interesting to continue tests with white light, to see if some setup of two white LEDs, or a white LED mixed with any of the colored ones, could result in an image in which all the benefits of the individual LEDs tested were realized. It seems likely though that since blue-red could see the stain while IR-blue could not, having a white LED might just nullify the benefits of having individual colors on their own. A good imaging approach would be to have two or three cameras in line, with different wavelengths and settings for each camera. This was the original design we suggested, and the results of the image optimization show this would increase the quality of images produced by the system. Having a system with red-red, blue-blue, and IR-IR setups inline with each other would allow the system to capture three images of the same chips, and process those three images separately to find certain defect types. The system would only really have to process the die of the red-red setup image, the substrate of all three, and the epoxy area of the IR-IR image. Having a setup like this would allow the system to be able to detect the most range of defects easily, including stains on the substrate, die defects, and the epoxy area.

11.2 Detection Test Results
An analysis was performed on 35 different images. The results can be seen in table 10
<table>
<thead>
<tr>
<th>Image</th>
<th>Image Outline Error</th>
<th>Label 1 Error</th>
<th>Label 2 Error</th>
<th>Defects Found</th>
<th>Defects Missed</th>
<th>%Detected</th>
<th>Lighting Type</th>
<th>Detection Software Functioned Properly</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>97</td>
<td>2</td>
<td>97.98%</td>
<td>IR-Blue</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>9</td>
<td>3</td>
<td>106</td>
<td>5</td>
<td>95.50%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>5</td>
<td>4</td>
<td>110</td>
<td>1</td>
<td>99.10%</td>
<td>IR-Blue</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>112</td>
<td>6</td>
<td>94.92%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>0</td>
<td>3</td>
<td>94</td>
<td>2</td>
<td>97.92%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>60</td>
<td>3</td>
<td>95.24%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>6</td>
<td>5</td>
<td>76</td>
<td>1</td>
<td>98.70%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>87</td>
<td>5</td>
<td>94.57%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>132</td>
<td>1</td>
<td>99.25%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>10</td>
<td>13</td>
<td>97</td>
<td>1</td>
<td>98.98%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>23</td>
<td>13</td>
<td>160</td>
<td>7</td>
<td>95.81%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>9</td>
<td>2</td>
<td>143</td>
<td>9</td>
<td>94.08%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>137</td>
<td>5</td>
<td>96.48%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>153</td>
<td>4</td>
<td>97.45%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>172</td>
<td>2</td>
<td>98.85%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>17</td>
<td>6</td>
<td>10</td>
<td>5</td>
<td>125</td>
<td>3</td>
<td>97.66%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>18</td>
<td>10</td>
<td>9</td>
<td>4</td>
<td>149</td>
<td>4</td>
<td>97.39%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>12</td>
<td>11</td>
<td>76</td>
<td>2</td>
<td>97.44%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>20</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>55</td>
<td>1</td>
<td>98.21%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>49</td>
<td>2</td>
<td>96.08%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>36</td>
<td>2</td>
<td>94.74%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>23</td>
<td>5</td>
<td>10</td>
<td>7</td>
<td>109</td>
<td>0</td>
<td>100.00%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>24</td>
<td>15</td>
<td>6</td>
<td>10</td>
<td>73</td>
<td>3</td>
<td>96.05%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>92</td>
<td>4</td>
<td>95.83%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>6</td>
<td>10</td>
<td>55</td>
<td>3</td>
<td>94.83%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>105</td>
<td>7</td>
<td>93.75%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>129</td>
<td>6</td>
<td>95.56%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>29</td>
<td>3</td>
<td>9</td>
<td>7</td>
<td>132</td>
<td>1</td>
<td>99.25%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>85</td>
<td>10</td>
<td>89.47%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>8</td>
<td>9</td>
<td>40</td>
<td>5</td>
<td>88.89%</td>
<td>Blue-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>11</td>
<td>9</td>
<td>136</td>
<td>3</td>
<td>97.84%</td>
<td>Red-Red</td>
<td>Yes</td>
</tr>
<tr>
<td>Averages</td>
<td>5.65625</td>
<td>6.40625</td>
<td>7.9375</td>
<td>102.6452</td>
<td>3.54839</td>
<td>96.66%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 10:** Detection test results

The first column is the trial number, the second through fourth are the pixel error for the outlining of the labels and the chip. The fifth column number of defects found in the image followed by the number of defect missed in the image. The percent detection accuracy is found...
in column 7 followed by the lighting used on the image and whether the software worked properly for this test case.

From the results the software found 96.7% of the defects in the images tested. The average defects per images found was 102.6 and the average defects missed is 3.55. The outline detection and label average error was a maximum of 7.9. This means that there is an average pixel error of 7.9 pixels. This is a very low error when the size of the image averages at 2258x2246 pixels.

The test performed on the robustness was to take 70 images and run the software detection on the images. The type is logged and so was whether it functioned like it was supposed to. Table 11 shows the number of each type of lighting tested. The detection software functioned properly for all images tested.

<table>
<thead>
<tr>
<th>Lighting Type</th>
<th>Number Images Tested</th>
<th>Number Images where software functioned properly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue-Red</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>IR-Blue</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>IR-Red</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>IR-UV</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Red-Red</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>Single Red</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>71</td>
<td>71</td>
</tr>
</tbody>
</table>

*Table 11: Results Table for detection software*
12. Conclusions and Design Recommendations

The project was considered a success and several conclusions regarding design have been made. Regarding hardware, several setups of different colored LEDs were found to be optimal for analyzing several types of defects. Using a pair of 2 red LEDs, die cracks, scratches, and foreign material on the substrate was easy to see. A blue LED paired with a red produced the best images for identifying stains and viewing the substrate embedded circuitry. An infrared and blue LED combination gave clear outlines of epoxy, while IR and red also gave a good view of the epoxy and foreign material. Therefore, it is recommended that a system like this be implemented using three combinations in manufacturing. The red-red combination, IR-blue combination, and the blue-red combination would produce 3 images can easily identify a wide range of defect classifications.

Regarding software, it seems that the best methods for finding defects on a computer chip’s die and substrate is using a contour analysis. Since the chip should inherently not have contours other than where the die and substrate meet (epoxy), this is clearly a great method to find defects. Defects that appear as stains, cracks, scratches, and foreign material all produce contours and are therefore easily identifiable. Image compression proved to be a challenging problem due to the nature of compressing an image – data will be almost always inevitably lost, especially at the ratios expected from this project. Therefore several methods of compression were devised. It is recommended that the standard use of the compression software be in the hybrid-JPEG format. This format offers high quality archival of the defect relative to other, less interesting, portions of the chip. However, it was decided that the quality of the photos be adjustable to help with identifying where defects are coming from in chip production. More specifically, if many chips were coming off as defective it may be useful to view a higher quality image to investigate the causes, and so the software is equipped to do so when necessary.
13. Acknowledgements

Our team would like to thank the following people for their various contributions to our project:

Michael Mahler – Intel
Jim Trego – Intel
Anthony King – University of Arizona
Dr. Jerzy Rosenblit – University of Arizona
Bryan Geunther – Veeco Metrology
Cassi Cucuel – Veeco Metrology
Frank Radocha – Flexlink

14. References


Windows is a registered trademark of Microsoft Corporation in the United States and other countries.
Acclaro® is a registered trademark of Axiomatic Design Solutions, Inc.
## Appendix A: Contributions

<table>
<thead>
<tr>
<th>Name</th>
<th>Contributions to final design report</th>
</tr>
</thead>
</table>
| Travis Rippstein  | • Introduction  
 |                   | • Top Level system design  
 |                   | • Software design (compression)  
 |                   | • Analysis (compression)  
 |                   | • Spec Review (compression)  
 |                   | • Conclusions and Design Recommendations  
 |                   | • Compression Software Tests  
 |                   | • Appendix A/C  
| Maribel Hudson    | • System Requirements  
 |                   | • Software Design (detection / categorization)  
 |                   | • Analysis (detection / categorization)  
 |                   | • Spec Review (detection / categorization)  
 |                   | • Detection Software test  
 |                   | • Appendix Organization  
| Frederick Chyan   | • Introduction  
 |                   | • System Requirements  
 |                   | • Document Formatting  
 |                   | • Appedix  
 |                   | • References  
 |                   | • Final System Tests  
| Mark Wiley        | • Analysis of optics and compilation  
 |                   | • Optical Design Concepts  
 |                   | • System Build (Hardware Mounting)  
 |                   | • System Build (Hardware Calibration)  
 |                   | • Acknowledgements  
 |                   | • Proposed Design  
| Jonathan Nation   | • System Requirements  
 |                   | • Optical Design Concepts  
 |                   | • Title Page Abstracts, and Table of Contents  
 |                   | • Illumination Optimization  
 |                   | • Illumination Settings Tests  
 |                   | • Illumination Settings Results  
 |                   | • Hardware system analysis  

Table 12: Individual Contributions
Appendix B: Hardware Models

The specifications and documentation for the Dalsa Piranha 3 camera system are given below. Figure 13 presents a set of technical drawings for the camera system. Figure 14 is a graph of the responsivity of the camera system through spectrum. Figure 15 shows the technical drawings for the Cobra linescan LED illuminators. Figure 16 gives a very important graph of the relative intensity vs. working distance of the linescan LED arrays. This graph will be used extensively when optimizing the LED placement. Figure 17 and 18 show different views of a solid model of the camera and illumination system design that will be constructed.

Figure 13: Technical Drawings for the Dalsa Piranha 3 camera system
**Figure 14:** Responsivity through spectrum at 0dB gain for Dalsa Piranha 3 camera system.

**Figure 15:** Technical drawings for the Cobra linescan LED illumination.

The COBRA Linescan device can be mounted using M4x0.7 metric threaded holes present on the heat-dissipating fins. The positions of these holes are shown in the drawings. There are holes on both sides of the unit. All dimensions are given in millimeters.
**Figure 16:** Relative Intensity and line thickness vs. the working distance of the Cobra linescan LED arrays.

**Figure 17:** Solid Model of the system design chosen. There are two sets of two linescan illuminators and two camera systems, pointed at a conveyor belt where the tray of chip moves along.
**Figure 18**: Solid Model top view of the system design. Cameras are centered over the focus line of the illumination system.
Appendix C: Software Code

C.1 Compression Algorithms Pseudo Code

X and Y variables are passed from Detection methods

Method CropDefect(X1, Y1, X2, Y2, totalSizeOfScan){
    NewImage <= CreateNewImage()
    Loop for X1 to X2
        Loop for Y1 to Y2
            NewImage.pixel( i  ,  j ) = This image (x1, y1)
            j++
        Y1++ Endloop
    i++
    X1++ Endloop
    If( x1 * y1 > 0.10 * totalSizeOfScan)
      Compress(NewImage)
    // Store information in the JPEG text header so the viewer knows where to put it
      NewImage.TextJPEGHeader = “X1, Y1”
    }
Compress(Image){
    For All Pixel Macroblocks
        // If there is an outlying value of a pixel in the block, don’t compress – it’s probably the defect and we need to keep that information
        Compare pixel values for outliers
        If(outlier = true)
          End loop
        // Otherwise, write the average value of those pixels into the entire block
        Else, Average Pixel Values for Image
          Image.CurrentMacroBlock <= AveragePixel Value
    }
C2. Matlab Code for GUI Compression and Detection

```matlab
function varargout = addGUI(varargin)

% TRAVIS RIPPSTEIN AND MARIBEL HUDSON
% CAPSTONE PROJECT ENGR 498
% DEFECT DETECTION CATEGORIZATION COMPRESSION (DDCC) MODULE

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
 gui_State = struct( 'gui_Name', mfilename, ...
     'gui_Singleton', gui_Singleton, ...
     'gui_OpeningFcn', @addGUI_OpeningFcn, ...
     'gui_OutputFcn', @addGUI_OutputFcn, ...
     'gui_LayoutFcn', [], ..., ...
     'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% Initialize GLOBAL VARIABLES - TJR

global imageLoaded
global x1
global x2
global y1
global y2
global cMode
global quality

% --- Executes just before addGUI is made visible.
function addGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to addGUI (see VARARGIN)
```
global x1
global x2
global y1
global y2
global cMode
global quality

x1 = 1;
x2 = 1000;
y1 = 1;
y2 = 1000;
cMode = 'H-JPEG';
quality = 75;

% Choose default command line output for addGUI
handles.output = hObject;

% Radio buttons thing - TJR
set(handles.cModes,'SelectionChangeFcn',@cModes_SelectionChangeFcn);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes addGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% load the background imageLoaded into Matlab
% if imageLoaded is not in the same directory as the GUI files, you must use the
% full path name of the iamge file
backgroundImage = importdata('Intel_Logo.jpeg');
% select the axes
axes(handles.axes1);
% place imageLoaded onto the axes
image(backgroundImage);
% remove the axis tick marks
axis off

backgroundImage = importdata('UA_Logo.jpeg');
% select the axes
axes(handles.axes2);
% place imageLoaded onto the axes
image(backgroundImage);
% remove the axis tick marks
axis off
% --- Outputs from this function are returned to the command line.
function varargout = addGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function input1_editText_Callback(hObject, eventdata, handles)
% hObject    handle to input1_editText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of input1_editText as text
%        str2double(get(hObject,'String')) returns contents of input1_editText as a double

% store the contents of input1_editText as a string. if the string
% is not a number then input will be empty
% input = str2num(get(hObject,'String'));
%
% checks to see if input is empty. if so, default input1_editText to zero
% if (isempty(input))
%   set(hObject,'String','0')
% end
% guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function input1_editText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to input1_editText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function input2_editText_Callback(hObject, eventdata, handles)
% hObject    handle to input2_editText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of input2_editText as text
% str2double(get(hObject,'String')) returns contents of input2_editText as a double

% store the contents of input1_editText as a string. if the string
% is not a number then input will be empty

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function input2_editText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to input2_editText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in add_pushbutton.
function add_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to add_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% a = get(handles.input1_editText,'String');
% b = get(handles.input2_editText,'String');
% % a and b are variables of Strings type, and need to be converted
% % to variables of Number type before they can be added together
% %
% total = str2num(a) + str2num(b);
% % c = num2str(total);
% % need to convert the answer back into String type to display it
% set(handles.answer_staticText,'String',c);
% guidata(hObject, handles);

global imageLoaded
global cMode

fileName = get(handles.input1_editText,'String');

if(strcmp(cMode,'H-JPEG'))
chipImage = imread([fileName '_CHIP.jpeg']);
defectImage = imread([fileName '_DEFECT.jpeg']);
L = textread([fileName '_LOC.txt']);

imageLoaded = chipImage;
imageLoaded(L(1):L(3),L(2):L(4)) = defectImage;

% Put red lines outlining the defect portion here if we want to...

else
    imageLoaded = imread(fileName);
end

scrsz = get(0,'ScreenSize');
figure('Name', fileName, 'OuterPosition',scrsz)
imshow(imageLoaded)

set(handles.loadText, 'String', [fileName ' Loaded']);
set(handles.loadText, 'BackgroundColor', 'green');

% --- Executes on button press in saver_button.
function saver_button_Callback(hObject, eventdata, handles)
    global imageLoaded
global cMode
global quality
global x1
global x2
global y1
global y2

    writeName = get(handles.input2_editText,'String');
    quality = get(handles.qualityText, 'String');
    quality = str2num(quality);

    if(strcmp(cMode,'L-JPEG'))
        imwrite(imageLoaded,[writeName '.jpeg'], 'Quality', quality);
    end

    set(handles.saveText, 'String', ['Saved ' writeName]);
    set(handles.saveText, 'BackgroundColor', 'green');
end

if(strcmp(cMode,'LL-JPEG'))
imwrite(imageLoaded,[writeName '.jpeg'], 'Mode', 'Lossless');
set(handles.saveText, 'String', ['Saved ' writeName]);
set(handles.saveText, 'BackgroundColor', 'green');
end

if(strcmp(cMode,'H-JPEG'))

%Needs to create the output text file as well

%Capture portion flagged as defective and write it

tempImage = imageLoaded;
defectivePortion = tempImage(x1:x2,y1:y2);
tempImage(x1:x2,y1:y2) = 0;
chipPortion = tempImage;
imwrite(defectivePortion,[writeName '_DEFECT.jpeg'], 'Quality', 100);
imwrite(chipPortion,[writeName '_CHIP.jpeg'], 'Quality', quality);

fileOut = fopen([writeName '_LOC.txt'],'w');
fprintf(fileOut,%d %d %d %d', x1,y1,x2,y2);
fclose(fileOut);

set(handles.saveText, 'String', ['Saved ' writeName '_DEFECT.jpeg']);
set(handles.saveText, 'BackgroundColor', 'green');

end

% --- Executes on button press in defectRun_button.
function defectRun_button_Callback(hObject, eventdata, handles)
% hObject    handle to defectRun_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global imageLoaded
global x1
global x2
global y1
global y2
fileName = get(handles.input1_editText,'String');
im=imageLoaded;
%imshow(im);
se = strel('rectangle',[15,15]);
level = graythresh(im);
IM2 = imerode(im,se);
BW = im2bw(IM2,level*.1);
dim = size(BW);
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)));
boundary = bwtraceboundary(BW,[row, col],'N');
counter=1;
figure('name',strcat('Defect Dection on ',fileName))
imshow(im)
hold on
boundaries = bwboundaries(BW);
contours=zeros([length(boundaries),9]);
for k=1:length(boundaries)
    b = boundaries{k};
    if(length(b)>300)
        %plot(b(:,2),b(:,1),'g','LineWidth',2);
        contours(counter,1)=k;
        [A,B,C,D,E,F,G,H]=info(b);
        contours(counter,2)=A;
        contours(counter,3)=B;
        contours(counter,4)=C;
        contours(counter,5)=D;
        contours(counter,6)=E;
        contours(counter,7)=F;
        contours(counter,8)=G;
        contours(counter,9)=H;
        counter=counter+1;
    end
end
imCenterY=size(im,1)/2;
imCenterX=size(im,2)/2;
plot(imCenterX,imCenterY,'b','LineWidth',40);
minDistance=100000;
distance=0;
The_center_contour=-1;
for i=1:length(contours)
    distance=abs(contours(i,6)-imCenterX)+abs(contours(i,7)-imCenterY);
    if(distance==0)
        break
    end
    if(distance<=minDistance&&contours(i,8)<400000&&contours(i,8)>30000)
        The_center_contour=i;
    end
end
minDistance = distance;
end

%%plot(contours(i,6),contours(i,7),'r','linewidth',20);
end
b = boundaries{contours(The_center_contour,1)};
x1=contours(The_center_contour,2);
y1=contours(The_center_contour,3);
y2=contours(The_center_contour,4);
y1=contours(The_center_contour,5);

%plot(b(:,2),b(:,1),'g','LineWidth',2);
outline=[contours(The_center_contour,6)+2-1041,contours(The_center_contour,6)+2+1041,contours(The_center_contour,7)-4-1027,contours(The_center_contour,7)-4+1027];

x_value_outline=[outline(1),outline(2),outline(2),outline(1),outline(1)];
y_value_outline=[outline(3),outline(3),outline(4),outline(4),outline(3)];
ident_mat_1=[outline(1)+28,outline(1)+28+245,outline(3)+554,outline(3)+554+939];
ident_mat_2=[outline(2)-33-245,outline(2)-33,outline(3)+563,outline(3)+563+939];

x_value_ident_1=[ident_mat_1(1),ident_mat_1(2),ident_mat_1(2),ident_mat_1(1),ident_mat_1(1)];
y_value_ident_1=[ident_mat_1(3),ident_mat_1(3),ident_mat_1(4),ident_mat_1(4),ident_mat_1(3)];

x_value_ident_2=[ident_mat_2(1),ident_mat_2(2),ident_mat_2(2),ident_mat_2(1),ident_mat_2(1)];
y_value_ident_2=[ident_mat_2(3),ident_mat_2(3),ident_mat_2(4),ident_mat_2(4),ident_mat_2(3)];

plot(x_value_outline,y_value_outline,'b','LineWidth',4);
plot(x_value_ident_1,y_value_ident_1,'b','LineWidth',4);
plot(x_value_ident_2,y_value_ident_2,'b','LineWidth',4);

% CenterImage=imcrop(im,
[contours(The_center_contour,2),contours(The_center_contour,4),contours(The_center_contour,3)-
contours(The_center_contour,2),contours(The_center_contour,5)-contours(The_center_contour,4))];
% figure, imshow(CenterImage)
Image_of_center=im;
Image_of_center_th=im;
Image_of_outside=im;
Image_of_outside_th=im;

for i=1:size(im,1)
    for j=1:size(im,2)
        
            if(j<=(contours(The_center_contour,2)) )
                j>=(contours(The_center_contour,3))&&i<=(contours(The_center_contour,4))
                    
                        Image_of_center(i,j)=0;
                        Image_of_center_th(i,j)=255;

                    end

            end
end

% figure, imshow(Image_of_center)
% figure, imshow(Image_of_outside)
levelc = graythresh(Image_of_center);
bwc=im2bw(Image_of_center_th,levelc*.3);
%figure, imshow(bwc);
dimc = size(bwc);
colc = round(dim(2)/2)-90;
rowc = min(find(bwc(:,colc))); boundaryc = bwtraceboundary(bwc,[rowc, colc], 'N'); BW_filledc = imfill(bwc, 'holes'); boundariesc = bwboundaries(bwc); %figure, imshow(im);
%hold on;
%contoursC=zeros([length(boundariesc),9]);
centercount=0;
for k=2:length(boundariesc)
b = boundariesc{k};
if (length(b)>60)
    contours(counter,1)=k;
    [A,B,C,D,E,F,G,H]=info(b);
    if(abs(D-contours(The_center_contour,6))<20&& abs(E-contours(The_center_contour,7))<20)
        centercount=centercount+1;
    else
        plot(b(:,2),b(:,1), 'g', 'LineWidth',2);
    end
else
    plot(b(:,2),b(:,1), 'g', 'LineWidth',2);
end
end

%DEFECT DETECTION ALGORITHM GOES HERE

function cModes_SelectionChangeFcn(hObject, eventdata)
global cMode

%retrieve GUI data, i.e. the handles structure
handles = guidata(hObject);

switch get(eventdata.NewValue, 'Tag')
    % Get Tag of selected object
    case 'radio_HJPEG'
        cMode = 'H-JPEG';
        % Code for when there is no match.
case 'radio_LLJPEG'
        cMode = 'LL-JPEG';
case 'radio_LJPEG'
        cMode = 'L-JPEG';
otherwise
    % Code for when there is no match.
function qualityText_Callback(hObject, eventdata, handles)
% hObject    handle to qualityText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of qualityText as text
%       str2double(get(hObject,'String')) returns contents of qualityText as a double

% --- Executes during object creation, after setting all properties.
function qualityText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to qualityText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over radio_LJPEG.
function radio_LJPEG_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to radio_LJPEG (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function [x1,x2,y1,y2,centerX,centerY, sudoarea ,angle] = info(aMatrix)
x2=-1;
x1=1000000;
y2=-1;
y1=1000000;
for j=1:length(aMatrix)
y=aMatrix(j,1);
x=aMatrix(j,2);
if(x>x2)
x2=x;
end
if(x<x1)
x1=x;
end
if(y>y2)
y2 = y;
end
if (y < y1)
y1 = y;
end
centerX = x1 + ceil((x2 - x1) / 2);
centerY = y1 + ceil((y2 - y1) / 2);
sudoarea = (y2 - y1) * (x2 - x1);
angle = tan((y2 - y1) / (x2 - x1)) * 360 / (2 * pi);
end
Appendix D: Images for Case Study

A case study to analyze the impact on the reduction / normalization of colors within JPEG was performed. Table 13 shows the results of the tests on various images using the JPEG baseline standard. The images that were used can be viewed in Appendix D. Essentially a color picture of Pentium 4 chip at 400 x 400 pixels was taken unaltered. Then by manually using an image editing program, the chip’s image was “dumbed down” where blocks of pixels were converted to the same color to reduce the file size. At small sizes the result is roughly 2x compression. However, larger sizes have more significant improvements when details are being removed. A size of 4000 x 4000 pixels yielded nearly 10x compression over the normal format. The sizes for the trays that are being imaged are 8000 x 20,000 pixels. Therefore the conclusion of this case study is that, for large images, reducing the number of colors within blocks of pixels can significantly help the JPEG algorithm compress while keeping a standard file format. Additionally, storing only the portion of the defect that is relevant will see improvements far greater than 10x improvements for disk space. Data collected and some sample images are included on the following page.
<table>
<thead>
<tr>
<th>Image</th>
<th>File Size @ 400 x 400 pixels</th>
<th>File Size @ 4000 x 4000 pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 (Color chip)</td>
<td>36.1KB</td>
<td>2.73MB</td>
</tr>
<tr>
<td>#2 (Altered chip)</td>
<td>14.2KB</td>
<td>.305MB</td>
</tr>
<tr>
<td>#3 (100% White)</td>
<td>3.05KB</td>
<td>244KB</td>
</tr>
</tbody>
</table>

**Table 13:** File Sizes and Pixel Area.

**Figure 19:** “Color chip” image used for image #1 in case study

**Figure 20:** “Altered chip” image used for image #2 in the case study
Appendix E: Compression Modes / Images

The following images can also be viewed on the CD accompanying this report. Images have been resized to fit in the document, please refer to the CD for the full-size images.

![Image: The Original 581KB Image]

**Figure 21:** The Original 581KB Image
Figure 22: The L-JPEG Mode image, lossy 257KB
Figure 23: The LL-JPEG Mode image, lossless 2425KB
Figure 24: The L-Gold Mode “defective” portion, lossy 47KB
Figure 25: The LL-Gold Mode “defective” portion and the H-JPEG “defective” portion, lossless 422KB
Figure 26: The H-JPEG “non-defective” portion, lossy 222KB
Appendix F: List of Test Images Captured During Optimization

<table>
<thead>
<tr>
<th>Serial numbers of row 1 test chips</th>
<th>serial numbers of row 2 test chips</th>
</tr>
</thead>
<tbody>
<tr>
<td>2T602056C 0214</td>
<td>2T602056C 0121</td>
</tr>
<tr>
<td>2T716031B 0009</td>
<td>2T716031B 0015</td>
</tr>
<tr>
<td>2T716031B 0212</td>
<td>2T716031B 0306</td>
</tr>
<tr>
<td>2T716031B 0231</td>
<td>2T716031B 0309</td>
</tr>
<tr>
<td>2T716031B 0288</td>
<td>2T716031B 0405</td>
</tr>
<tr>
<td>2T716031B 0344</td>
<td>2T716031B 0407</td>
</tr>
<tr>
<td>35629857B 0085</td>
<td>2T716031B 0410</td>
</tr>
<tr>
<td>MB742053A02702</td>
<td>91803240A14177</td>
</tr>
</tbody>
</table>

Table 14: Serial numbers of the test chips used. Row 1 was the main row of test chips, with uniform lighting on this row. Row 2 was a secondary row that did not have uniform lighting, so it was not used for any comparisons.

<table>
<thead>
<tr>
<th>Wavelength</th>
<th>Angle (°)</th>
<th>Wavelength</th>
<th>Angle (°)</th>
<th>Wavelength</th>
<th>Angle (°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>18.5</td>
<td>IR</td>
<td>32.5</td>
<td>blue</td>
<td>32.5</td>
</tr>
<tr>
<td>Red</td>
<td>25</td>
<td>IR</td>
<td>40</td>
<td>blue</td>
<td>40</td>
</tr>
<tr>
<td>Red</td>
<td>30</td>
<td>IR</td>
<td>45</td>
<td>blue</td>
<td>45</td>
</tr>
<tr>
<td>Red</td>
<td>35</td>
<td>IR</td>
<td>50</td>
<td>blue</td>
<td>50</td>
</tr>
<tr>
<td>Red</td>
<td>39.5</td>
<td>IR</td>
<td>55</td>
<td>blue</td>
<td>55</td>
</tr>
<tr>
<td>Red</td>
<td>42</td>
<td>IR</td>
<td>60</td>
<td>blue</td>
<td>60</td>
</tr>
<tr>
<td>Red</td>
<td>47.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Red</td>
<td>53.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 15: Angles of the images captured for stage 1 of the optimization, using only a single LED on the right side, at a working distance of 3".

<table>
<thead>
<tr>
<th>Left Wavelength</th>
<th>Right Wavelength</th>
<th>Left Angle (°)</th>
<th>Right Angle (°)</th>
<th>Left WD (°)</th>
<th>Right WD (°)</th>
<th>Aperture Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>Red</td>
<td>45</td>
<td>45</td>
<td>3</td>
<td>3</td>
<td>2.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>48</td>
<td>49</td>
<td>3</td>
<td>3</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>55</td>
<td>55</td>
<td>3</td>
<td>3</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>59</td>
<td>50</td>
<td>3</td>
<td>3</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>67</td>
<td>50</td>
<td>3</td>
<td>3</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>70</td>
<td>35</td>
<td>3</td>
<td>4.5</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>70</td>
<td>45</td>
<td>3</td>
<td>3</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>70</td>
<td>50</td>
<td>3</td>
<td>3</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>70</td>
<td>53</td>
<td>3</td>
<td>3</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>70</td>
<td>60</td>
<td>3</td>
<td>3.75</td>
<td>3.5</td>
</tr>
<tr>
<td>Red</td>
<td>Red</td>
<td>70</td>
<td>65</td>
<td>3</td>
<td>4.75</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Table 16: Settings for the red-red wavelength pair illumination optimization tests.
<table>
<thead>
<tr>
<th>Left Wavelength</th>
<th>Right Wavelength</th>
<th>Left Angle (°)</th>
<th>Right Angle (°)</th>
<th>Left WD (&quot;)</th>
<th>Right WD (&quot;)</th>
<th>Aperture Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Red</td>
<td>39</td>
<td>37</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>44</td>
<td>37</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>45</td>
<td>45</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>50</td>
<td>50</td>
<td>3</td>
<td>3</td>
<td>4.25</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>54</td>
<td>37</td>
<td>3</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>60</td>
<td>37</td>
<td>3</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>60</td>
<td>60</td>
<td>3</td>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>72</td>
<td>37</td>
<td>3</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>72</td>
<td>46</td>
<td>3</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>72</td>
<td>50</td>
<td>3</td>
<td>3.5</td>
<td>4.5</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>72</td>
<td>55</td>
<td>3</td>
<td>4</td>
<td>4.5</td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td>72</td>
<td>60</td>
<td>3</td>
<td>5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Table 17: Settings for the blue-red wavelength pair illumination optimization tests.

<table>
<thead>
<tr>
<th>Left Wavelength</th>
<th>Right Wavelength</th>
<th>Left Angle (°)</th>
<th>Right Angle (°)</th>
<th>Left WD (&quot;)</th>
<th>Right WD (&quot;)</th>
<th>Aperture Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR</td>
<td>Red</td>
<td>43</td>
<td>39</td>
<td>3.25</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>43</td>
<td>45</td>
<td>3.25</td>
<td>3.25</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>49</td>
<td>45</td>
<td>3</td>
<td>3.25</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>54</td>
<td>45</td>
<td>3</td>
<td>3.25</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>60</td>
<td>45</td>
<td>3</td>
<td>3.25</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>60</td>
<td>51</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>72</td>
<td>39</td>
<td>3</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>72</td>
<td>44</td>
<td>3</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>72</td>
<td>51</td>
<td>3</td>
<td>3.5</td>
<td>4.5</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>72</td>
<td>56</td>
<td>3</td>
<td>4</td>
<td>4.5</td>
</tr>
<tr>
<td>IR</td>
<td>Red</td>
<td>72</td>
<td>61</td>
<td>3</td>
<td>5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Table 18: Settings for the IR-red wavelength pair illumination optimization tests.
<table>
<thead>
<tr>
<th>Left Wavelength</th>
<th>Right Wavelength</th>
<th>Left Angle (°)</th>
<th>Right Angle (°)</th>
<th>Left WD (&quot;)</th>
<th>Right WD (&quot;)</th>
<th>Aperture Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR</td>
<td>Blue</td>
<td>38</td>
<td>41</td>
<td>2.5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>38</td>
<td>29</td>
<td>4.5</td>
<td>3.5</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>44</td>
<td>49</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>44</td>
<td>54</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>47</td>
<td>29</td>
<td>3</td>
<td>3.5</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>52</td>
<td>54</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>53</td>
<td>60</td>
<td>3</td>
<td>3</td>
<td>5.25</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>68</td>
<td>58</td>
<td>3.25</td>
<td>3.25</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>68</td>
<td>63</td>
<td>3.25</td>
<td>3.25</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>73</td>
<td>26</td>
<td>3</td>
<td>4.75</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>73</td>
<td>27</td>
<td>3.25</td>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>73</td>
<td>34</td>
<td>3.25</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>73</td>
<td>41</td>
<td>3.25</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>73</td>
<td>54</td>
<td>3.25</td>
<td>3.25</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>Blue</td>
<td>73</td>
<td>63</td>
<td>3.25</td>
<td>3.25</td>
<td>5</td>
</tr>
<tr>
<td>IR</td>
<td>UV</td>
<td>64</td>
<td>55</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>IR</td>
<td>UV</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 19: Settings for the IR-blue and IR-UV wavelength pair illumination optimization tests.