**Using Image Registration to Find Errors in Circuit Boards**

Kevin Shevchuk

Dr. Dudley Girard, Mentor

Dr. Thomas Briggs, Mentor

**ABSTRACT**

There is a need to detect errors in the manufacture of circuit boards. Going from schematic to finished circuit board there are some significant problems caused by worn out or broken drill bits and defects in the copper or snags during the milling process. Because of a lack of automated error-checking solutions available, printed circuit boards must be checked by hand in a very inefficient process. The following details background research into developing a software application to analyze images of the completed circuit boards and a proposed solution to the problem to be developed in software. Development focuses on creating an application that is easy to use and provides meaningful feedback to the user to facilitate locating possible errors in a timelier manner.

1. **INTRODUCTION**

The image processing segment of this project involves five stages that must work together to produce a clear, easy to see and understand result. First is image registration. Second is removal of image noise. Third is detection of the edges of image features. Fourth is extraction of these image features into data for the fifth stage, comparison of image features to other images or in this case, to the original Gerber file the board was printed using. The result is a representation of the edges of the circuit board features displayed to the user. Circuit board features that are errors should be clearly highlighted for the user to see.

1. **IMAGE REGISTRATION**

Image registration is the process of aligning two images to the same orientation and scale so they can either be stitched together to create a larger image or used to assist in comparing features between two images. There are many factors that affect what registration method needs to be used for a particular application. Registration can be done in both two and three dimensional spaces, as well as deal with differences in perspective between images.” According to [38], “generally, the transformations can be divided into rigid and non-rigid transformations.” Rigid methods work with rotation and translation errors, while the non-rigid methods work with local differences in the images, i.e. warping [38]. The approaches to image registration are divided into feature-based or intensity based [38]. Feature based approaches use specific features like points, curves or other specific shapes and use a transformation based on that feature [38]. Intensity based approaches use the intensity of the images to compute the transformation [38].

* 1. **Control Point Registration**

Control Point Registration is a feature matching approach to image registration that uses feature points to extrapolate local data about the image. Control points can either be manually selected or automatically generated. For situations where automated control points are needed, corner detectors can work well. Corner detectors search for points where the edge gradient changes direction significantly and records those points for reference.

A common method involves using Random Sample Consensus (RANSAC) to derive features from points for best fit. The basic premise of RANSAC is to start at a small dataset and grow the set as more data as consistent data points are discovered [10]. [10] Describes in more detail the process of applying RANSAC for image processing.

The primary result of registration processing is the creation of transformation matrices that are applied to an image. In order to find the final transformation which would be used to transform the image in the desired manner, algorithms tend to approximate the transformation by checking the square error of the transformation [1]. Thus the control points are run through the algorithm till the square error is sufficiently low enough to be a close approximation to the transformation [1].

There are several problems with image registration that can affect the accuracy of the results and the computation time. As the complexity of the transformation that is required to register the images increases, so do the required number of points needed and the increased number of reliable matches to get a reliable transformation [35]. Because some estimation is needed to obtain a reasonable transformation, success is not guaranteed [35]. This is particularly a problem in very complex transforms where most algorithms would be unable to come up with a solution in a reasonable amount of time.

-

For this project, because the user is selecting the control points and only two dimensions need be dealt with and there should be virtually no warping or perspective change. Only scaling and rotation need to be done so rigid, feature based approaches are necessary.

1. **NOISE REMOVAL**

Noise removal is the first stage of any image processing application. Noise in the image can interfere with edge detection and extraction algorithms by producing false positives. The quality of an image declines due to the imperfections in image capturing sensors, analog to digital conversion and bit errors during transmission etc. [8, 17]. The most common types of noise encountered in images are Impulse (frequency) noise, Gaussian noise and a combination of both called mixed noise [26]. Gaussian noise makes the image slightly soft and blurry, where each pixel in an image is changed by a small amount [17]. There are two types of impulse noise. Salt and pepper noise has pixels taking either maximum (black) or minimum (white) gray level value creating spots on an image [8]. Random valued input noise has pixels taking random grayscale values across the whole spectrum. The three primary noise removal techniques covered here are Gaussian blurs, Median filters and Anisotropic (Perona-Malik) Diffusion.

* 1. **Gaussian Blur**

The Gaussian is a linear low-pass filter based on passing a Gaussian function over the entire image to smooth out noisy pixels. The Gaussian blur works by passing a Gaussian matrix with values weighted like Figure 1 over each pixel of the image. As the matrix passes over the image, the matrix value is multiplied by the value of the pixel underneath. These values are then added together to create a weighted average of the area covered by the matrix for each pixel. The resulting value becomes the new grayscale value for the pixel under the center square of the matrix [11]. Increasing the standard deviation of the distribution in the matrix creates a stronger blur effect [11].

There are two key problems with the Gaussian blur. First, “low pass filtering can capture global structure of the image but lose local control” [9]. Sharp edges are lost using a low-pass filter because the filter does not discriminate toward preserving those edges. This can partially be remedied by sharpening the image afterwards. Re-sharpening the image is undesirable because as [22] states; “Most of the filters in literature consist of a lot of user defined parameters or weights to the pixels which depends on the content of the image.” Since the performance of the enhancement depends on such user specified values, it leads to poor or over enhancement and artifacts on images.” [22] Notes that Genetic algorithms or particle swarm optimization can optimize user values. However this adds complexity to the problem that may be avoided using an edge aware detection algorithm or an algorithm with high performance on blurred images. The latter is not covered here.

Second, linear filters have poor performance in the presence of noise that is not frequency noise [26]. Salt and pepper noise will not be smoothed out completely by linear filters. This is because the averaging method of the Gaussian filter is influenced by the outlier values of salt and pepper noise changing them more than necessary.

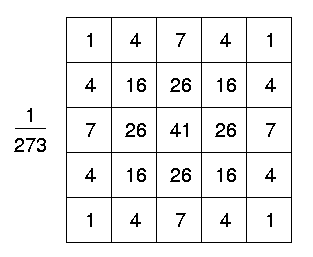


Figure 1 Discrete approximation to a Gaussian Function with a standard deviation of 1.0 expressed as a 5x5 matrix [9].

* 1. **Median Filter**

Median filters are a type of non-linear filter which relies on the median value of a matrix filter rather than the mean value. The procedure is to take the pixel values in the neighborhood of a pixel, find their median and use this value to replace the value of the center pixel [4]. Where Gaussian filters utilize a Gaussian distribution across the matrix, median filters use a binary matrix, one representing a pixel to be counted, zero representing a pixel to be ignored. Thus, asymmetric filters of differing shapes can be used to deal with edges and corners if required [4].

[19] Notes that “Non linear filters exhibit better performance as compared to linear filters when restoring images corrupted by impulse noise.” This is because median filters do not created additive effects to an image. Gaussian blurs modify all the pixel values of an image by a variance determined by the matrix values. Median filters however, according to [4] “Have local control and information but do not have any information about the global structure of the image.” Edges are preserved while blurring is minimal. Figure 2 shows the significantly reduced blurring effect of the median filter while still preserving much of the edges and reducing the noise. As Figure 2 shows, Median filtering does not remove all the noise, and in some cases, fine detail is lost, such as the eyelashes.



Figure 2: Detail of the "Lena" image showing the effects of Gaussian blurs and Median filters [12].

[4] Covers in detail the use of weighted median filters for greater control and flexibility in filter operation.

* 1. **Anisotropic Diffusion (Perona-Malik)**

Anisotropic Diffusion is a technique developed by Perona and Malik [28] that uses piece-wise smoothing and immediate localization to improve the quality of a noisy image [25] and is quite effective for smoothing images with edge preservation [16]. The anisotropic diffusion method models blurring using a non-linear partial differential equation called the diffusion equation which is also used to model heat transfer [28]. These images are used to piece a final image together using convolution filters that is the solution to the diffusion equation [28]. Different filters are used to copy different parts of different images in the image set. The resulting image will use stronger blurs in homogeneous regions and sharper portions near edges. This can be done iteratively to enhance the effect to the level desired as shown in Figure 3. Equation 1 is the equation for anisotropic diffusion as defined in [28].

The conduction coefficient is a function imposed upon the gradient that governs how much blurring is to occur at a point [28]. If the coefficient is constant the diffusion would be isotropic and the blurring would be constant across the entire image [28].

The term c(x, y, t) ∆I is the gradient of the brightness function which is used as an estimator ||*E*|| of edges and non-edges [28]. This produces a one dimensional graph of the magnitude of the gradient as the estimator passes across the rows of the image [28]. Where the estimator returns large values, the conduction coefficient should be small and where the estimator returns small values, the coefficient should be large. Large coefficients indicate that blurring should be strong at that point. Hence, the coefficient is said to be inversely proportional to the gradient.

**Equation 1:** The definition for anisotropic diffusion [28]. I is a family of gray-scale images [28]. ∇ is a differential operator that when applied to a scalar field returns a vector field where each vector defines the magnitude and direction of greatest change in the scalar field (also known as its gradient) [6]. ∆ is the Laplacian, a second-order operator that is equivalent to the divergence of the gradient of a scalar field [6]. c(x,y,t) is the conduction coefficient, and div(…) is the divergence operator which returns a number quantifying the inwardness or outwardness of the flow of a vector field in the neighborhood of a point [6].

Equation 2: Perona and Maliks’ two functions for generating scale space [28]. The first favors high contrast edges over low contrast ones. The second favors wide regions over smaller ones. The estimator ||*E*|| is equivalent to ||c(x, y, t) ∆I|| [28]. *k* is a constant [28].

The divergence operator measures the change in the flux at a particular point [6, 28]. The gradient at that point is convolved with the conduction coefficient to produce the net flux [28]. This flux determines how much the intensity of the pixel at that particular point should change on successive images in the iteration of the algorithm.

One of the drawbacks to anisotropic diffusion is that it has a high computation cost, especially for large data sets however this can be remedied partially because it is highly data-parallel and suitable for massively parallel architectures such as General Purpose Graphical Processing Units (GPGPUs) [25].

-

Of the three noise removal methods discussed above, each has benefits and drawbacks. Gaussian blurs ignore edges and need sharpening afterwards. Median filters do not blur the edges but fine textual detail can be lost. Anisotropic Diffusion produces exceptional noise removal while retaining very sharp edges and much fine detail but has expensive computational costs, and will benefit greatly from parallelizing the process.



**Figure 3:** Anisotropic diffusion results for t = (0, 1, 4, 10, 20, 50, 100, 200) showing the successive passes and edge preservation capabilities [41].

1. **EDGE DETECTION**

Edge Detection is the second stage in image processing. “The edge detection process serves to simplify the analysis of the images by drastically reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries” [5]. [5] Also lays out three criteria that good edge detectors should have. First, they should have a low error rate [5]. Second, “edge points should be well localized; that is, the distance between the points marked by the detector and the center of the true edge should be minimized” [5]. Finally, there shouldn’t be multiple responses to a single edge [5]. The noise removal method chosen should have eliminated most of the noise of the image to mitigate false positives and minimize errors. The two edge detection methods that are discussed here are the Canny Edge Detector and the Gabor Filter.

* 1. **Canny Edge Detector**

The Canny filter was first developed in 1986 by John Canny to satisfy his three requirements of a good edge detector as listed above [5]. The primary edge detection component of Canny is its ability to find edges using the gradient strength and direction. The image is first run over with a Gaussian filter [28, 37]. A two dimensional first derivative operator is applied to the smoothed image to highlight regions of the image with high first spatial derivatives [28, 37].

**A B C**

**1 2 1 2 1 0 3 3 1**

**M1= 0 0 0 + 1 0 -1 = 1 0 -1**

**-1 -2 -1 0 -1 -2 -1 -3 -3**

**-1 -2 -1 -2 -1 0 -3 -3 -1**

**M2= 0 0 0 + -1 0 1 = -1 0 1**

**1 2 1 0 1 2 1 3 3**

**1 0 -1 0 -1 -2 1 -1 -3**

**M3= 2 0 -2 + 1 0 -1 = 3 0 -3**

**1 0 -1 2 1 0 3 1 -1**

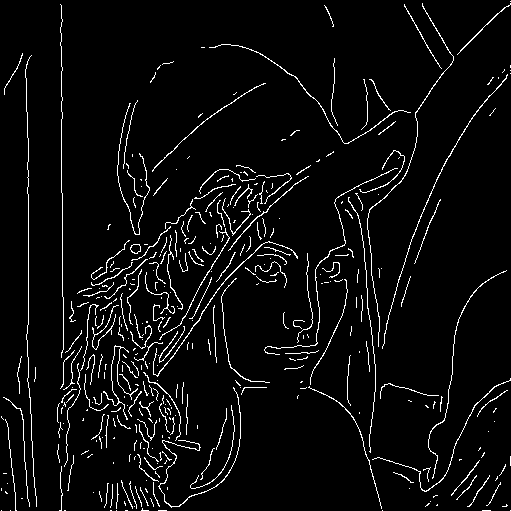
**-1 0 1 0 1 2 -1 1 3**

**M4= -2 0 2 + -1 0 1 = -3 0 3**

**-1 0 1 -2 -1 0 -3 -1 1**

**Figure 4:** The Sobel detectors suggested by [39]. Each one of the 12 3 by 3 matrices checks the gradient in a different direction in passing over the image [39].

The first derivative operator is set up as a 3x3 matrix of values like those shown in Figure 4, and passed over the image [40]. Varying filters can be used for convolution depending on the implementation. Column A in Figure 4 are filters to check gradient angles on the cardinal directions, column B filters check gradients on diagonals, and column C creates a hybrid of both. Only horizontal and vertical filters are necessary according to [16].

1. 
2. 
3. 

**Figure 5:** Canny filter applied to the Lena image. (a) The original image. (b) The image after detection. (c) The thinned image after segmentation. Standard deviation is set to 3 [28].

According to [21] “Small-scaled filters are sensitive to edge signals but also prone to noise where large-scaled filters are robust to noise but could filter out the fine details.” Depending on the noise removal method, Canny may perform poorly based on the degree of blurring if a Gaussian blur that is too strong is applied [21]. Figure 5a shows little distortion from extra noise; however some false edges appear in Figure 5c. Canny is however, widely modified and combined with other algorithms and techniques [21, 32, 39] to reduce error rate and improve performance.

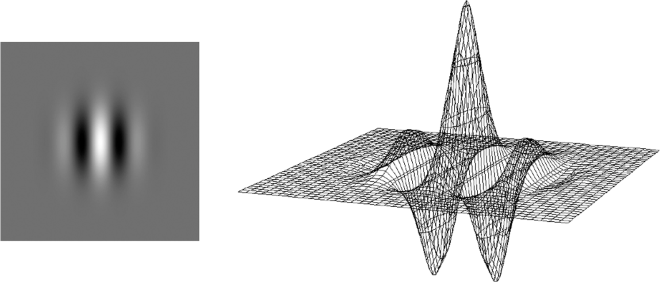
* 1. **Gabor Filter**

The Gabor filter is a convolution filter composing a Gabor function defined in Equation 3 [23] and shown as a heat-map and three dimensional rendering in Figure 6. It tries to investigate intermediate representations that are combinations of time/space and frequency information [14]. It consists of a Gaussian function multiplied by a sinusoidal carrier wave [23].



**Equation 3:** *(a)* the definition of a two dimensional Gabor function where s(x, y) is a complex sinusoid denoted by *(b)* and w(x, y) is a Gaussian function [23].

A Gabor filter is defined as g(x, y, θ, ϕ) centered at the origin with θ the spatial frequency and ϕ the orientation [31]. When convoluted with the image I(x, y), g(x, y, θ, ϕ) is the response to the filter at the coordinates x, y [31]. A horizontal filter will produce high responses on horizontal regions, a vertical filter on vertical regions, etc [31]. The grayscale edge detected image is the x, y projection of the responses [31]. It is possible to get unidirectional detection by only applying one filter and plotting the results which could have some interesting applications.



**Figure 6:** Heat-map and 3D rendering of a Gabor filter [33].

The major drawback to Gabor Filters is that they are very computationally expensive, especially when using a large number of filters in a set [24]. This is mitigated by the fact that they are both data and task parallel because of the numerous filters used in the generation. By using N scales and M orientations, the number of passes the computations necessary increases N∙M times [31]. Thus, Gabor filters can gain large performance boosts through parallel processing. The extra overhead to parallelizing this process may not be desirable for some tasks.

-

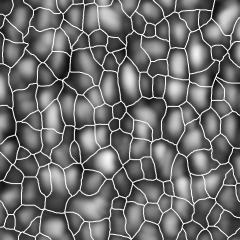
Some of the above edge detection methods have their own feature extraction methods that they are typically used with. Canny goes with non-maximum suppression [5], Gabor filters have their own method but work just as well using non-maximum suppression [24, 31]. Both of the above methods take different approaches to highlighting the edges of the image features, but both strive for the same result, producing a binary map of the image with the fewest errors and high detection [5]. Being able to clearly outline the features simplifies the next phase of image processing.

1. **FEATURE EXTRACTION**

Feature extraction involves obtaining a mapping of the exact edges of the features of an image as a binary representation like Figure 5c. This binary representation allows for the now clearly defined edges to be compared against other images using various methods outlined in the next section. The two approaches explored here for feature extraction are watersheds and non-maximum suppression.

* 1. **Watersheds**

Watersheds work by setting the image up as a topographical map, lighter grayscale values being higher than low values [7]. The gradient image generated during the edge detection is then “flooded” with virtual water [7]. As the water rises up the gradient, it starts with the darkest parts first and works its way to the topographic ridges [7]. Single pixel lines will form where two water flows meet [7]. These high points where flows meet become part of the binary image [7, 37].



**Figure 7:** Watershed Segmentation of a blurred image [15]. The white lines represent the segmentation boundaries of the gradients [15]. Note that this map used inverted topography during flooding.

One of the benefits of the watershed approach is that it has a very high detection rate compared with the non-maximum suppression as shown by a comparison of Figure 5c to Figure 7. It will always find the maximum values of an edge in the image [2]. One of the drawbacks however to watersheds is their tendency to over-segment the image if there is a lot of fine detail or too much noise [2]. There are two approaches to fix this problem. Proper noise reduction should smooth gradients enough to reduce the number of segmentations, or use a marker based watershed algorithm [37].

Markers are user designated or computer designated points to expand from when growing the region during the flooding process [37]. Because of the way the flooding works, if a single point is used, the system will overflow some topography in an arbitrary order causing one region to spill into others creating more segmentation. Markers allow numerous points to grow more evenly, preventing flow-over and allowing one region to neatly meet up with the surrounding regions [37]. The flooding is accomplished using a search on a weighted graph of nearby pixels [7].

* 1. **Non-maximum Suppression**

Non-maximum suppression is a technique used extensively in the Canny edge detector to minimize the edges down to the single pixel width and create a binary image [5, 39]. The edges produced by the edge detector are run through an algorithm which tracks along the tops of the ridges and zeroes all pixels that are not actually on the top of the ridge [39]. This “thinning” produces a thin line as an output as shown in Figure 5c.

“Linear features are defined as a sequence of points where the image has a maximum in the direction with the largest variance, gradient, or surface curvature” [34]. [5, 34] use directional filters to detect similar maxima and mark them as zero if they are not. The directional filters usually cover 0, 45, 90 and 135 degrees, though others such as 22.5, 67.5, 112.5, and 157.5 degrees can also be used [34]. Canny suggests using six directions in his trials [5].These linear windows are passed separately across the image, and then pieced back together to create the final image using the union formula in Equation 4 [34].

**Equation 4:** The formula for unifying the suppressed features. LDi is the result at direction Di, ND is the number of directions used and L is the combined result [34].

Canny uses a threshold for directional computation calculated based on the amount of noise in the image [5]. However, a user designated number allows for the user to tune the sensitivity to his/her desired outcome. Too low of a threshold and the segmentation will create lines where they are not needed. Too high and only the sharpest features will be segmented. Figure 5c is not the perfect threshold because some single pixels that are not edges still appeared on the image map. It is difficult to find a threshold that will not generate extra noise, extraneous points, or leave holes.

-

Neither of the above approaches for feature extraction are perfect. Canny reacts to excess noise and poor threshold selection. Watersheds create excess segmentation if markers are not used. There are extensions and variations on each of the above approaches to minimize some of these errors, which are not discussed here.

1. **FEATURE COMPARISON**

The final stage of this process is the comparison of the extracted features to another image. According to [36] “Existing object detection methods can be categorized into learning-based approaches and template-based approaches.” Learning-based approaches are outside the scope of this project and as such, only template matching methods will be explored here. “In the template-based approach, objects are described explicitly by templates and the goal of object detection becomes to find the best matching template given an input image” [36]. The methods discussed here are Manhattan distance, and Chamfer matching.

* 1. **Manhattan Distance (L1 Distance)**

Manhattan distance or taxicab distance is used for the similarity comparison between the point P1 with the coordinates (x1, y1) and the point P2 at the point (x2, y2) between two separate images [3]. It is a form of template matching using vectors to represent the features of the images. For comparing larger feature sets, it can be generalized to the equation in Equation 5.

**Equation 5:** Definition of Manhattan distance between the two feature vectors, x and y [18].

Manhattan hashing, is an extension of this approach that uses a hash of the vector data to speed up the comparisons by dividing the vector up into sections, computing a hash value, and then comparing the hash values of the parts to the target template to produce results [18]. [18] Discusses using binary hashing, breaking the vector into four parts and computing a binary representation of the vector. Rather than comparing the binary images as a whole to each other (requiring a large number of computations per pixel), only computed vectors from the source and destination image are compared to each other, reducing the number of computations[3]. The values produced by the comparison are an estimate of similarity [18]. On a binary image, the more correct comparisons, the more likely the vectors match.

* 1. **Chamfer Matching**

Chamfer matching uses the binary representation of the image as an edge image, comparing points directly between the two images as in Equation 6 below:

**Equation 6:** The formula for Chamfer distance. X and Y are the two images and ||a-b|| denotes the Euclidean distance between two pixels, a and b in the image [30].

One of the upsides to using Chamfer matching is that it provides a smooth measure of the fitness and can tolerate some small rotations, misalignments, occlusions and deformations. However it becomes much less reliable in the presence of background clutter [20]. This is not a main problem because of the geometric simplicity of the circuit board. The benefit of working with circuit boards is the clearly defined edges that should have no background noise. Should background clutter become an issue, there are modifications explained in [20] that help mitigate most of these problems.

-

The problem with feature comparison is finding a matching measure and algorithm to evaluate the template with the correspondent part of the image [29]. There are a number of different searching algorithms, but the primary goal is to find a matching criterion with high reliability [29]. “The matching criterion has to cope successfully with any type of distortions – translation, rotation, scaling, nonlinear changes in the intensity, etc” [29]. Most of these issues will be mitigated before the noise removal stage; however, those issues could still be present so it should be important to choose a method that can compensate for any problems.

“All the algorithms for template localization, based on consecutive comparison with the template (so called brute-force algorithms) have got great computational complexity, thus they work slowly and it makes them unworkable for real-time applications” according to [29]. This is not a major issue for this project because it is not done as part of a real time application, and accuracy of the outcome is more important than execution time. [29] Lists some other methods for speeding up the template matching algorithm.

1. **PRIMARY OBJECTIVE**

To develop an application to perform noise removal, edge detection, and feature extraction on a digital image of a circuit board and compare the extracted features to the original schematic to detect errors in manufacture.

* + The level of effort shall not exceed 1 person-month over 12 weeks.

1. **SOLUTION DESCRIPTION**

Because the usefulness of this application hinges on output to the user, much of the development will be devoted to creating an easy to use and informative user interface. The calculations will be done using the built in image processing functions in MATLAB. These functions are already correct and proven to work, leaving more development time for creating the user interface. The research done for this project detailed above was largely used to determine exactly which image processing routines would be most effective in providing the most useful feedback to the user by reducing the probability that the applications processing pipeline would cause errors on output.

The schematic of the board is a Gerber file. The Gerber file standard is an ASCII file that provides instructions to the Computer Numerical Control (CNC) machine which follows them when milling the circuit board. The file is characterized by sets of coordinates and machine instructions specifying what action to perform at a certain coordinate or a path to perform said action on. For this project, it is necessary to transform the Gerber file into a format that is usable, likely a vector file such as a Scalable Vector Graphics (SVG) file which can then scale well with the image registration without loss of detail. However, MATLAB itself does not support vector graphics, and so the vector would have to be rasterized before the comparison would take place.

Images of the printed circuit boards will be taken with a digital scanner at 1200dpi resolution. Because detail up to one-tenth of a millimeter needs to be preserved, the image needs to be taken at a high resolution. 1200dpi provides forty-two pixels per millimeter which may or may not be sufficient enough resolution to retain fine image features. One-tenth of a millimeter would be preserved in 4.7 pixels. Tests of the application would determine whether the scanning resolution would need to be increased to 2400dpi or higher to preserve fine detail. The drawback to higher resolutions is increased scanning time in scanning the printed circuit board. The main benefit to using a scanner is that extremely high resolutions can be achieved without worrying about distortions or warping that would be caused by perspective as would appear when using a digital camera on a mount.

Because of the fact that a scanner is being used to obtain the source image, the image registration procedure should not need to account for warping because of perspective. The input images should be as two-dimensional as possible to reduce the complexity and error probability. Because automated approaches to image registration have issues with proper matching, manual matching will be done based on a set of user specified control points in the images.

From sample trials of various image processing routines it was determined that anisotropic diffusion produced the most useful output. It successfully reduced noise better than median filters in the image while preserving edges to a high degree. After about fifteen iterations of the anisotropic diffusion algorithm, the less well-defined edges began to blur at the ends. It was determined that between ten and fifteen iterations were suitable for adequate noise removal.

Canny filter proved to be no better or worse off than Gabor filters at detecting edges. While both methods produced an edge map neither appeared to have over or under-sampled the image by a significant degree. While sample trials were performed using a simple Gaussian blur, both implementations produced a similar number of false positives for edges, a factor that will be significantly decreased with the use of anisotropic diffusion for noise removal. Canny filters are less computationally expensive than Gabor filters and for the sake of performance, it was decided that the Canny filter would be used for edge detection.

Non-maximum suppression is a frequent part of most Canny implementations so it was selected over watersheds. MATLAB also uses non-maximum suppression in its default implementation of the Canny filter.

It was decided that the comparison phase would be downplayed due to the image registration process scaling and aligning the images to the same resolution. As such it has been reduced to a 1:1 comparison of pixels between the two images. No real scaling or alignment issues should need to be taken into account unless the image registration process has some serious issues.

All of these operations will be MATLAB functions called from inside the application, thus, MATLAB will need to be present on the workstation in which this application will be used.

1. **PRODUCT DESCRIPTION**

The application itself will either be entirely programmed using MATLAB or will have some components written in Java, because of this the actual implementation of this product may differ slightly from the proposed implementation below. Java and MATLAB work well together, but MATLAB won’t necessarily provide all the needed resources until a more thorough review of MATLABS GUI and image processing functions can be done.

* 1. **High Level Product Description**

When the application is launched a GUI will be presented to the user (Figure 8) which prompts for the filename for the Gerber file to be loaded into the software and the source image taken from the scanner. If the image hasn’t been scanned yet, the user can use the scan option to scan directly into the software. The two images will be presented to the user in two separate display areas and the user is prompted to select control points that the images share in common. The user will be specifically asked to use drill holes because they have extremely high contrast compared to the surrounding board features.

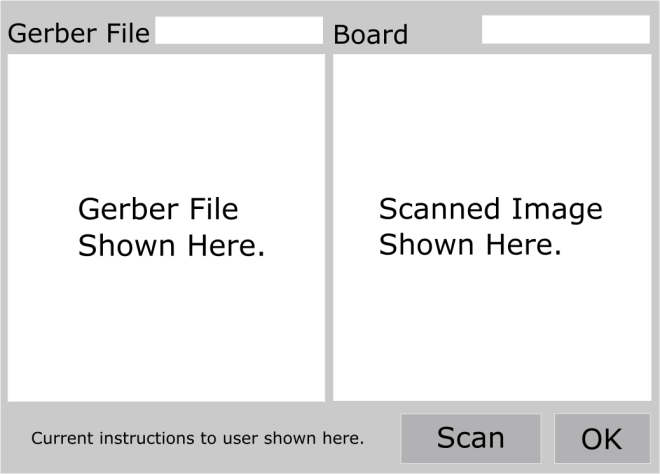


Figure 8: A Mockup of the Image Registration GUI. Users enter filenames in the context boxes at the top.

Once the user has selected the points, hitting the run button then executes the image processing routines in the background. When done, the results are made into transparent, colored layers and shown on the resulting GUI, Figure 9. The base image is shown in the background as grayscale and the other layers are each toggle-able depending on what the user wants to see. When overlaying two layers, the user should be looking for spots where only one of the layers colors is showing. These areas indicate where there is a high possibility of an error. Figure 10 shows mockups of these overlays.

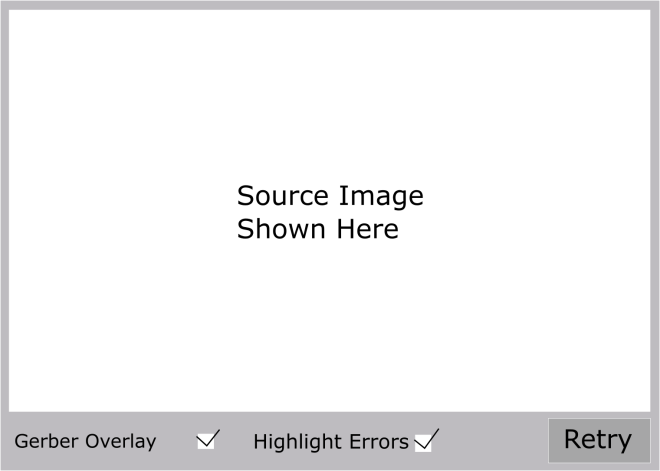


Figure 9: A mockup of the output GUI. The transparent overlays can be toggled on and off using the checkboxes on the bottom.

* 1. **Product Justification**

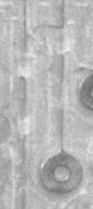
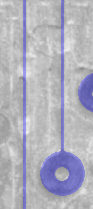
There are many applications for image processing that are used for specialized purposes. While there are publically available applications, most of them are generalized and require time to learn to use and some of them cost money. The primary justification for this project is to provide software to perform the task of comparing the Gerber file to the produced board, to determine whether errors exist in a cost-effective manner, with little learning time.

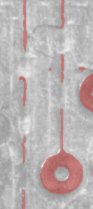
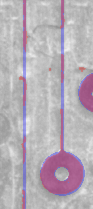
* 1. **User Description**

The primary users for this application are any students in Shippensburg University’s Computer Engineering major who are using the departments CNC machine to print circuit boards for development projects. This application is applicable to any situation where comparing a Gerber file to the result of that Gerber file is needed, perhaps having a much wider range of use.

* 1. **User Documentation**

User documentation will be provided with the application in the form of a PDF file explaining various details about the software, including basic use of the software, explanation of its features, explanation of the output to the user and possible interactions with that output that are available to the user.

A  B

C  D

**Figure 10:** Mockups of the resulting transparent overlays from the software. A) The base source image showing several severed copper traces caused by the machine. B) The Gerber file overlaid over the source image. C) The edge map overlaid over the source image. D) Both transparent layers overlaid showing where the edge map does not match the Gerber file indicating a possible error.

1. **PRODUCT DEMONSTRATION**

The software is demonstrated to the audience by letting it run on a set of pre-determined inputs, explaining the process at each step and explaining the features of the UI at each interactive stage. The output of a pair of inputs that are forcibly incorrect will be shown, as well as the output of a pair of correct inputs (ones without any errors). Any additional cases which produced errors, such as bad contrast inputs, or scans with too much noise would also be demonstrated to explain the outstanding issues of the software.

1. **SOLUTION**

The application turned out quite differently from the planned solution above. It contains three main parts, the Launch GUI, the Control Point Selection GUI, and the Image Registration GUI.

When the application is launched, the Launch GUI appears where the user can load in the Gerber file, the drill file and the scanned image. For reasons explained in the control point selection GUI, the drill file is a necessary addition. The Gerber files are parsed and rendered directly from the loaded files and displayed on the windows for the user to verify that they look good. Several things happen during this process.

**11.1 Gerber Parsing and Rendering**

First, the file is read sequentially beginning with parsing the header to extract any useful rendering information such as the unit type, and zeros padding.

After the header comes the aperture definition. This contains all of the basic shapes that make up the file. A typical aperture definition contains a shape type defined as a primitive. The most common shapes are circles and rectangles, but the Gerber standard supports oblong shapes and n-sided polygons. It also contains the coordinates of the center of the circle and its diameter. In the case of a rectangle, the diameters of both dimensions are separate values.

As a Gerber file is a set of instructions meant to be followed one after another, one has to deal with two possible coordinate systems. The Gerber standard supports both absolute and incremental coordinates for files. To simplify things and speed up rendering, all coordinates are forced to absolute for the applications internal representation of the data. This saves having to calculate these values during render time, especially if multiple files need to be rendered on the same image.

The Gerber standard also has the ability to draw lines between two points. By default this uses the last supplied aperture definition as the width of the line. This also means that this can be done with no aperture definition selected. This allows for drawing complex polygons. Because no definition is set, the width of the trace is effectively zero, causing the software to treat it as an outline. This is useful because this allows the manufacturer to speed up manufacturing time by not performing a rub-out (removing all non-trace copper from the surface). This has the drawbacks of increasing the likelihood of a short on the board if there was a significant error in milling. This process would be accomplished in this software by drawing the outline and then performing a flood-fill algorithm on the space. This functionality was left out of the application due to time constraints but it is important to note that this application only supports a barebones set of Gerber instructions. Therefore, not all Gerber files render completely in the first iteration of the software.

The Gerber standard also has support for custom defined aperture macros which are combinations of aperture definitions. Support for this was also left out of the software as it is largely unused by the department.

The Gerber file is rendered by following the internal representation instruction by instruction on a pre-sized image. As the file is processed, a function keeps track of the bounds of the Gerber file to create the best image size. This is done for performance reasons as performance degrades quickly as the images get larger.

Two other operations are performed on the base data set before rendering begins. First, the image must be mirrored across the Y-axis due to the different origin locations between Gerber files and MATLAB. The Gerber file is rendered with the origin in the lower left corner, while MATLAB’s origin is in the upper left. Secondly, the image is grounded back to the origin. This is because Gerber files are not ground to the origin by default. This involves finding the coordinate closest to the origin and subtracting that distance from all coordinates of the image.

The drill file is also rendered using the same rendering process as the Gerber file; however the drill files contains some differences in how the file is laid out, so a different parser was needed to process this. The drill file is of no use if rendered without the Gerber file below it. So it is always rendered on top of the base image. This involves drawing black circles of proper diameter centered where each drill hole is.

**11.2 Processing the Scanned Image**

The scanned in image is a full RGB image of the circuit board scanned at a designated DPI. For purposes of testing, the images are scanned at 1200 DPI though the application supports higher resolutions. Added to the GUI (See Appendix Figure 12) is a dropdown menu containing several preset DPI values and the user selects the one to match the scanned image.

During testing it was discovered that most of the image processing methods described in the literature background did not work well for this particular task. None of the edge detection methods provided a sufficient level of edge preservation or detection to be useful for this application. A solution was devised however. Rather than work in gray scale it was decided to use the relations between the values for the RGB color channels for specific pixels to determine what is copper and what is not. During initial testing of this copper detection method it was discovered that it detected approx. 99% of the copper on the board. It had to have additions put in to detect darker shades of copper and reflections caused by the scanners light. This detection method also produced some false positives but in low enough quantities that a median filter passed over the image could suppress them.

Contrary to usual image processing methods, it was decided to perform noise removal after edge detection. This decision was made because of a need for accuracy. By only using a single pass of a 3x3 median filter after detection to clean up the detector, there is minimal risk of weakening the detector by performing heavy amounts of noise removal on the image before the edge detector can run. The primary drawback to this is there is nothing to protect the edge detector from noise except for its own detection limitations.

During this phase a bigger problem was also discovered. The scanner used to create these images would sometimes pick up the copper on the walls of a drill hole, particularly on the extremities of the board. This has major implications as to the effectiveness of the control point registration as some drill holes do not get detected correctly.

**11.3 The Control Point GUI.**

Control point registration was added as a solution to possible warping of the scanned image. Initially this was not thought to be a problem until during registration testing using manual methods, it was discovered that the scanned image was stretched in both dimensions.

The most important consideration during this phase was providing a consistent method of registration that could be repeatable with minimal error. It was decided that the drill holes would be the most useful common points of reference between the two images, hence why the drill file had to be added.

The user must select three reference points on each image to proceed in the application. The user can’t be trusted to always select the same normalized reference points each time so a method to correct the points to increase the effectiveness of aligning the images had to be developed. As the drill file is processed, all drill-hole center coordinates are stored in an array. If the user clicks on the Gerber file, the application finds the closest center point to that click coordinate and substitutes that for the clicks registration coordinates. This ensures that every point used in the registration is the exact center of that drill hole.

On the source image there is no guarantee that a drill hole actually resembles a hole. So a custom algorithm (See Appendix Figure 13) was created to find the center of the blob that may make up the hole on the image.

Function centroid\_blob(x\_coor, y\_coor)

Limit = 50

ex\_factor = 3

While (perimeter points not all 1){

Image = ex\_factor by ex\_factor section of source centered around click;

Check edge points;

If (4 or less are 1)

Ex\_factor + 3;

If (5 or 6 are 1)

Ex\_factor + 2;

If (6 or 7 are 1)

Ex\_factor + 1;

If (8)

Find\_centroid(Image)

If exfact > Limit

Break; //Expansion limit exceeded.

}

Function find\_centroid(matrix)

[m, n] = size(matrix);

s = Sum of x coordinate values for black pixels per row.

s = s/count of black pixels in row.

A = average(sum(s for each row)/m)

s = Sum of x coordinate values for black pixels per column.

s = s/count of black pixels in column.

B = average(sum(s for each rows)/n)

Return(A, B)

**Figure 11:** The algorithms for finding the center of the drill holes on the scanned image. It finds the average x coordinate of each row and column that contains black pixels which is the center of the blob.

The sets of coordinates returned from each algorithm are passed to a transform operation which uses them to reshape the scanned image to match the same coordinate space as the Gerber file. Note that incorrect or wild selection of points leads to rather undesired results. Thus the user is prompted to select drill holes only as points. Because the edge detector is not perfect, it is up to the user to decide which holes would have a high chance of finding a close real center.

**11.4 Image Registration GUI**

Once the control point selection is completed the result is computed and fused into a single image and displayed to the user on the registration GUI (See Appendix Figure 14). Here the user can fine-tune the registration if the transform was not perfect but moving or rotating the source image. The Gerber file is stationary and cannot be moved in this phase, only the source image is movable.

The source image is displayed as green while the Gerber file is displayed as purple, points where they overlap show up as white. Excess amounts of green indicate that the source image has properties that the Gerber file doesn’t have or has features in the wrong place. Excess amounts of purple indicate that there is expected to be copper there but it is missing on the scanned image. The user should pay attention to any areas where large amounts of either purple or green is showing as these are indicators of possible problem areas.

1. **LIMITATIONS**

The two weakest points of this application are the quality of the scanner taking the image of the board, and the accuracy of the copper detection. Using a scanner with a tighter scanning focus would minimize the chances of scanning the walls of the drill holes, leading to cleaner holes on the extremities of the board. It would also address some of the warping issues seen from some scanning tests.

Adjustments to the copper detection could fine-tune it more to get better edges, but this problem remains that the holes have poor definition because of the fact that the full spectrum of copper color has be to be found. During a few of the tests, searching for darker shades of copper ended up making smaller drill holes disappear entirely from the detector. However removing the detection of darker shades would impact the detection on images in which the scanners illumination lit up the boards differently, causing all the copper to not show up.

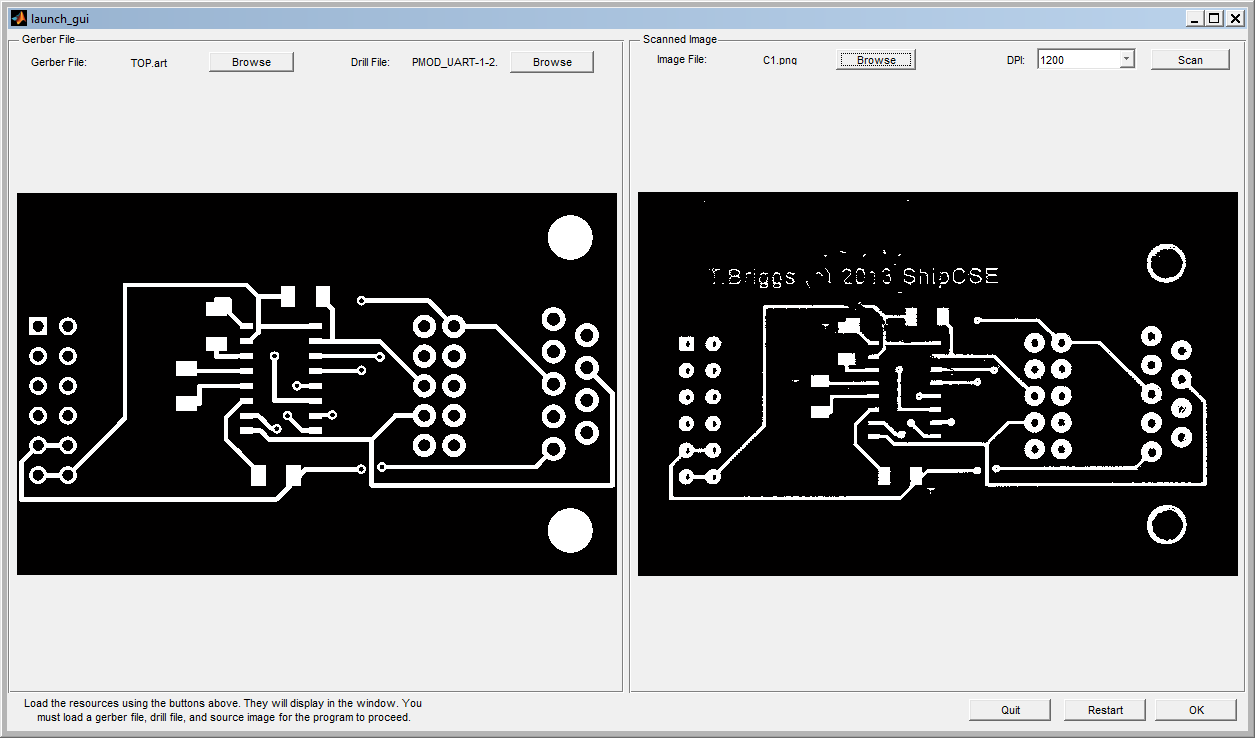
1. **FUTURE WORK**

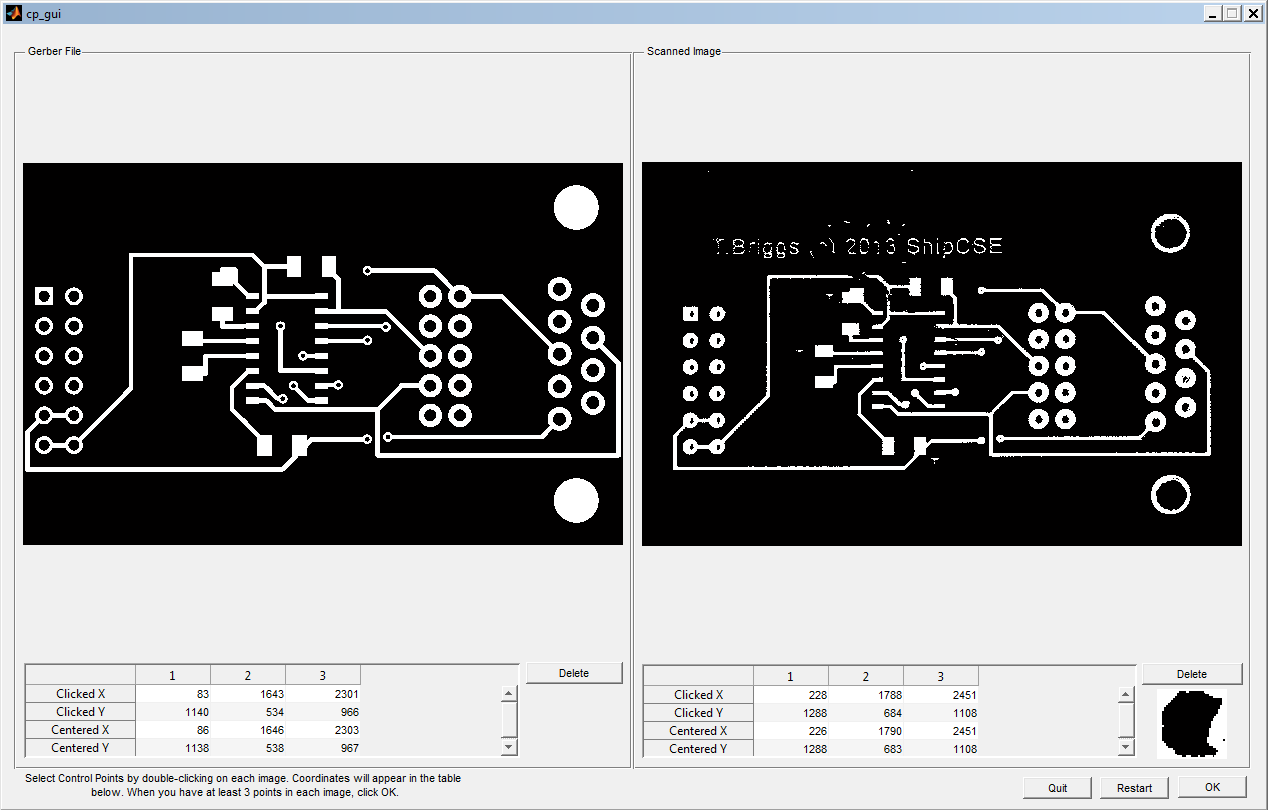
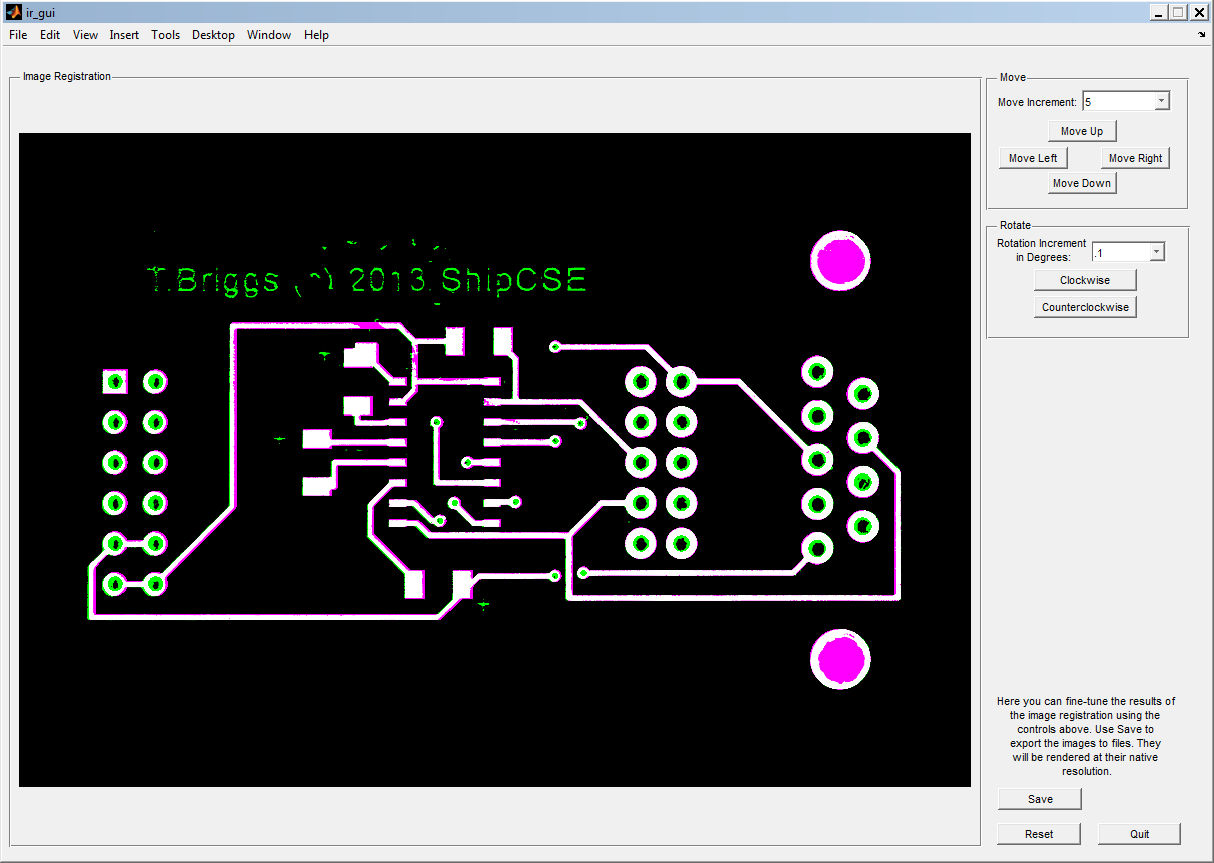
There are some improvements that could be made to this application to increase usefulness to the user. The main improvement is added full support for all relevant Gerber instructions. This will allow the application to be used on a very large subset of Gerber files. Also useful to this application would be the ability to load all the Gerber files relevant to the board and selecting which side or part to view. The application would then find the appropriate files, load and render them. This is an addition more from a usability standpoint. Finally, support for direct scanning from a USB scanner. This provides for full system integration, simplifying the interactions needed by the user.

1. **CONCLUSION**

Image processing is a field that is still in in infancy, but it is quickly evolving to become a tool to solve problems in many fields. While not necessarily useful for everything, the results of this application prove that it is a useful solution to a costly problem. While it still needs improvement and refinement, its use should result in faster error detection on circuit boards, and lower costs in production.

1. **REFERENCES**
2. S. Amiri, S. Kasaei, “an Ultra Fast Method to Approximate Affine Transform using Control Points” [Online] http://ce.sharif.edu/~mo\_amiri/pub/FastAffine.pdf
3. W. Barrett, A. Cheney, “Object-based Image Editing” in Proc. of the 29th Annu. Conf. on Computer Graphics and Interactive Techniques, New York, NY, USA, 2002, pp. 777-784 http://dl.acm.org/citation.cfm?id=566570.566651
4. R. Barskar, G. Ahmed, “CBIR Using Fuzzy Edge Detection Mask” in Proc. of the 2001 Int. Conf. on Communication, Computing & Security, New York, NY, USA, 2011, pp 277-280 http://dl.acm.org/citation.cfm?id=1947940.1947999
5. D. Brownrigg, “A Weighted Median Filter” in Commun. of the ACM, New York, NY, USA, 1984, pp807-818 http://dl.acm.org/citation.cfm?id=358198.358222
6. J. Canny, “A Computational Approach to Edge Detection” in IEEE Trans. Pattern Anal. Mach. Intell. 1986, Vol 8, No. 6, pp. 679-698 http://origin-www.computer.org.navigator-ship.passhe.edu/csdl/trans/tp/1986/06/04767851.pdf
7. M. Corral, “Gradient, Divergence, Curl and Laplacian” in *Vector Calculus* GNU Free Documentation License, 2008, ch 4.6, pp. 177-185 http://www.mecmath.net/calc3book.pdf
8. J. Cousty et al. “Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle” in IEEE Trans. Pattern Anal. Mach. Intell. 2009 pp.1362-1374 http://origin-www.computer.org.navigator-ship.passhe.edu/csdl/trans/tp/2009/08/ttp2009081362.pdf
9. B. Deka, D. Baishnab, “Removal of Random-valued Impulse Noise Using Overcomplete DCT Dictionary” in Proc. of the CUBE Int. Inform. Technology Conf., New York, NY, USA, 2012, pp 42-46 http://dl.acm.org/citation.cfm?id=2381716.2381726
10. J. Dong et al. “Design Pattern Detection by Template Matching” in Proc. of the 2008 ACM Symp. on Applied Computing, New York, NY, USA, 2008, pp 765-769 http://dl.acm.org/citation.cfm?id=1363686.1363864
11. M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM, 24(6):381–395, 1981. http://dl.acm.org/citation.cfm?id=358669.358692
12. R. Fisher et al. (2003) “Gaussian Smoothing” [Online] http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
13. E. Goulliart, G. Varoquaux, (2003) “Image Manipulation and Processing Using Numpy and Scipy” [Online] http://www.tp.umu.se/~nylen/fnm/pylect/advanced/image\_processing
14. A. Hirani, T. Totsuka, “Combining Frequency And Spatial Domain Information for Fast Interactive Image Noise Removal” in Proc. of the 23rd Annu. Conf. on Comput. Graph. and Interactive Techniques, New York, NY, USA, 1996, pp. 269-276 http://dl.acm.org/citation.cfm?id=237170.237264
15. Y. Ji et al. “Efficient Edge Detection and Object Segmentation Using Gabor Filters” in Proc. of the 42nd Annu. Southeast Regional Conf., New York, NY, USA, 2004, pp. 454-459 http://dl.acm.org/citation.cfm?id=986537.986651
16. Sedlář Jiří et al. “Detection of Elliptical Particles in Atomic Force Microscopy Images”, ICASSP 2011: IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Praha, CZ, 2011 http://zoi.utia.cas.cz/afm-particles
17. H. Kekre, S. Gharge, “SAR Image Segmentation Using Co-Occurrence Matrix And Slope Magnitude” in Proc. of the Int. Conf. on Advances in Computing, Communication and Control, New York, NY, USA, 2009, pp.368-372 http://dl.acm.org/citation.cfm?id=1523103.1523177
18. N. Khan et al. “Fuzzy Based Diffusion Coefficient Function In Anisotropic Diffusion For Impulse Noise Removal” in Proc. of the 8th Indian Conf. on Comput. Vision, Graph. and Image Process., New York, NY, USA, 2012, Article 67 http://dl.acm.org/citation.cfm?id=2425333.2425400
19. W. Kong et al. “Manhattan Hashing for Large Scale Image Retrieval” in Proc. of the 35th Int. ACM SIGIR Conf. on Research and Develop. in Inform. Retrieval. New York, NY, USA, 2012, pp 45-54 http://dl.acm.org/citation.cfm?id=2348283.2348293
20. M. Kumar, P. Kumar, “Image Fusion Using Bi-Dimensional Empirical Mode Decomposition for Sale and Pepper Noise Removal In Digital Images” in Proc. of the 2nd Int. Conf. on Computational Sci., Eng. and Inf. Technology, New York, NY, USA, 2012, pp. 346-351 http://dl.acm.org/citation.cfm?id=2393216.2393275
21. M. Liu et al. “Fast Directional Chamfer Matching” [Online] http://www.umiacs.umd.edu/~mingyliu/papers/liu\_cvpr2010.pdf
22. T. Manish et al. “Edge Detection by Combined Canny Filter With Scale Multiplication & Ant Colony Optimization” Proc. of the 2nd Int. Conf. on Computation Sci., Eng. and Inform. Technology, New York, NY, USA, 2012, pp.497-500 http://dl.acm.org/citation.cfm?id=2393216.2393299
23. S. Mohammed et al., “Particle Swarm Based Unsharp Masking” in Proc. of the 7th Indian Conf. on Comput. Vision, Graph. and Image Process., New York, NY, USA, 2010, pp. 498-505 http://dl.acm.org/citation.cfm?id=1924559.1924626
24. J. Movellan (2008) “Tutorial on Gabor Filters” [Online] http://mplab.ucsd.edu/tutorials/gabor.pdf
25. M. Naskar, R. Parekh, “Texture Recognition in Images Using Wavelets” [Online] http://iem.edu.in/conference/IEMCON6JAN/Computational%20Intelligence%20S-6A/spsitm2011\_submission\_118.pdf
26. V. Pallipuram et al., “A Multi-Node GPGPU Implementation of Non-Linear Anisotropic Diffusion Filter” in 2012 Symp. on Application Accelerators in High Performance Computing, Argonne, IL, USA, 2012, pp. 11-18 http://www.computer.org.navigator-ship.passhe.edu/csdl/proceedings/saahpc/2012/4838/00/4838a011-abs.html
27. J. Patel et al. “Fuzzy Inference Based Edge Detection System Using Sobel and Laplacian of Gaussian Operators” Proc. of the Int. Conf. & Workshop on Emerging Trends in Technology, New York, NY, USA, 2011, pp.694-697 http://dl.acm.org/citation.cfm?id=1980022.1980171
28. P. Patel et al., “A New Adaptive Median Filtering Technique for Removal of Impulse Noise from Images” in Proc. of the 2011 Int. Conf. on Communication, Computing & Security, New York, NY, USA, 2011, pp. 462-467 http://dl.acm.org/citation.cfm?id=1947940.1948035
29. P. Perona, J. Malik, “Scale-Space and Edge Detection Using Anisotropic Diffusion” IEEE Trans. on Pattern Anal. Mach. Intell., 1990, Vol 12, No 7, http://authors.library.caltech.edu/6498/1/PERieeetpami90.pdf
30. Y. Petkova, N. Nuri, “M-estimators Over Separable Sets of Representative Points for Template Matching” in Int. Conf. on Comput. Syst. and Technologies, Sofia, Bulgaria, 2010. Pp. 210-216 http://delivery.acm.org/10.1145/1840000/1839417
31. M. Potamias, V. Athitsos, “Nearest Neighbor Search Methods for Handshape Recognition” in Proc. of the 1st Int. Conf. on Pervasive Technologies Related to Assistive Environments, New York, NY, USA, 2008, Article 30. http://dl.acm.org/citation.cfm?id=1389586.1389622
32. V. Prasad, J. Domke, (2005) “Gabor Filter Visualization” [Online], Tech. Rep., University of Maryland http://www.cs.umd.edu/class/spring2005/cmsc838s/assignment-projects/gabor-filter-visualization/report.pdf
33. S. Sastry, “Canny Edge Detector Demos” [Online] http://robotics.eecs.berkeley.edu/~sastry/ee20/cademo.html
34. S. Sin et al., “Fingerprint Verification With Scratch Feature For Personal Portable Devices” *Optical Engineering*. 51(2), 2012 http://opticalengineering.spiedigitallibrary.org/article.aspx?articleid=1183327
35. C. Sun, P. Vallotton, “Fast Linear Feature Detection Using Multiple Directional Non-Maximum Suppression” in Int. Conf. on Patten Recognition” 2006, pp. 288-291 http://origin-www.computer.org.navigator-ship.passhe.edu/csdl/proceedings/icpr/2006/2521/02/252120288.pdf
36. C. Taylor, A. Bhusnurmath, “Solving Image Registration Problems Using Interior Point Methods.” [Online] GRASP Laboratory, University of Pennsylvania http://www.cis.upenn.edu/~cjtaylor/PUBLICATIONS/pdfs/TaylorECCV08.pdf
37. N. Thanh et al., “An Improved Template Matching Method for Object Detection” ACCV Comput. Vision, 2010, Vol. 5996, pp 193-202, Abstract, http://link.springer.com/chapter/10.1007%2F978-3-642-12297-2\_19
38. H. Wang, C. Yuan, “License-plate Location Method Based on Integrated Features and Marker Controlled Watershed Algorithm” in Int. Conf. on Electrical and Control Eng., 2010, pp.1234-1237 http://origin-www.computer.org.navigator-ship.passhe.edu/csdl/proceedings/icece/2010/4031/00/4031b234.pdf
39. X. Wang et al., “Elastic Medical image Registration based on Image Intensity” in Proc. of the 2001Pan-Sydney area workshop on Visual Inform. Processing, 2001, Darlinghurst Australia, AUS, pp. 139-142
40. L. Weibin et al., “An new Canny-based Edge Detector for SAR Image” in 2008 Congr. on Image and Signal Processing, 2008, pp. 211-215 http://origin-www.computer.org.navigator-ship.passhe.edu/csdl/proceedings/cisp/2008/3119/03/3119c211.pdf
41. S. Xiao Fan et al., “Edge Based Region Growing: A New Image Segmentation Method.” In Proc. of the 2004 ACM SIGGRAPH Int. Conf. on Virtual Reality Continuum And its Applications on Industry, New York, NY, USA, 2004, pp. 302-305 http://dl.acm.org/citation.cfm?id=1044588.1044653
42. X. Yuan, “Image Processing, Diffusion” [Online] http://www.users.cs.umn.edu/~xyuan/imageproc/diffuse.htm

**Figure 12:** The Initial GUI the user is presented with when launching the application. The images above are generated on the fly based on the files the user selects for input.

**Figure 13:** The control point selection GUI the user is presented with once the files are loaded and rendered. In the tables below are the points selected by the user and the corrected points that are centralized by the software. The centralizing of the points reduces the chance of inaccuracy by attempting to determine the center of a feature on the scanned image and searches for an indexed drill hole on the Gerber file. Note the small image in the bottom right which is a preview of the selected feature on the scanned image.

**Figure 14:** The image registration GUI. Here the user can fine-tune the registration using the controls on the right. The scanned file is represented by the green while the Gerber file is purple.