Automated Trace and Copper Plane Extraction of X-ray Tomography Imaged PCBs

Ulbert J. Botero, Fatemeh Ganji, Damon L. Woodard, Domenic Forte
jbot2016@ufl.edu, fganji@wpi.edu, {dwoodard, dforte}@ece.ufl.edu

***Abstract*—Reverse engineering (RE) of Printed circuit boards (PCBs) can be achieved through X-ray Computed Tomography (CT) in a non-destructive manner. For practical applications such as obsolescence replacement and hardware assurance, it is important to perform RE as automated and fast as possible. While our past work has focused on automated via detection, this paper introduces a framework for automated and unsupervised extraction of traces and copper conducting planes in an X-ray CT imaged PCB. We employ a series of algorithms from image processing, computer vision, and graph theory for a robust solution that produces a high fidelity result. We compare our results against off-the-shelf methodologies utilizing ridge detection and Canny edge detection in combination with connected components and show the necessity of utilizing a graph-theoretic approach. Our results are demonstrated on an in-house designed PCB for qualitative and quantitative analysis. On average across all layers, our proposed framework outperforms the relevant methods with IoU, SSIM, Correlation, and Dice image quality scores of 0.9166, 0.9954, 0.8952, and 0.9453, respectively.**

***Keywords*—printed circuit board; reverse engineering; computer vision; X-ray computed tomography; trace; ground plane.**

I. Introduction

At the heart of all electronic systems is the printed circuit board (PCB), which connects components together through a series of conductive interconnects, namely vias and traces. Vias are conductive electroplated holes that bore through multiple layers to provide connectivity between PCB layers. Traces provide conductive signals between components and vias within a PCB layer. Traces are unique to their discrete layers as opposed to vias which at minimum have adjacent layer connectivity. Another important PCB building block is the power/ground plane, a large area of copper foil on the board that is connected to the power supply/ground and serves as the source/return path for components on the PCB.

Each of these building blocks is vital to the faithful functionality of a design. If any were altered, especially in a PCB utilized in security-critical infrastructures, it could have drastic implications. One simply has to look at the field of hardware security, and more specifically, hardware Trojans for apt examples. The alteration of trace spacing and dimensions can cause malicious effects such as cross-talk [1]. Furthermore, an attacker can induce electromigration simply by thinning the traces of a design [2]. Such attacks are explored in research studies, but a possible real-world example was highlighted in October 2018. According to "The Big Hack" [3], unauthorized implants (disguised as conditioning components) were found on server motherboards used by high profile entities such as Apple, Amazon, and the US Government. As reported in [4], these hardware Trojans are especially difficult to detect using runtime Trojan detection techniques. The article also alluded to the possibility of embedding components in internal layers of the PCB. In any case, the only foolproof method of PCB trust and assurance is through comprehensive reverse engineering (RE). The most appropriate method of PCB RE utilizes X-ray Computed Tomography (CT) [5] because it non-destructively captures the internal connectivity of PCBs. However, RE of X-ray CT imaged PCBs remains a non-trivial process with substantial manual labor necessary to annotate the trace locations, widths, and spacings in the imaged data. Therefore, it is important to develop algorithms that automate the entire process as much as possible.

Automated trace and copper plane extraction is particularly challenging. Traditional ML techniques used, e.g., for classification or object detection, leverage domain knowledge about the task at hand. For example, the prototypical circular shape of vias assists with via detection by utilizing the popular Hough transform [6]. This cannot be assumed for trace and copper plane extraction since there is no set geometry for either of them. PCBs can use serpentine routing (see Figure 1(a)) to induce a delay [7], minimize skew by matching trace length [8], and reduce signal reflections due to discontinuities in trace width [9]. The copper plating of a layer embedded via software such as polygon pours [10] are also not constrained to any geometric criteria, see Figure 1(b). Earlier works explored trace detection by deep learning (DL), but have ignored conducting planes. In particular, Qiao et al. [11] used a combination of deep convolutional neural networks (DCNN) with graph-cut models to semantically segment traces in a variety of X-ray CT imaging environments. This approach is useful to address the lack of geometric constraints for shape-based modeling approaches; however, it is heavily reliant on (manually labeled) training data. Furthermore, DL approaches live and die by the volume of training data and how well it rep-

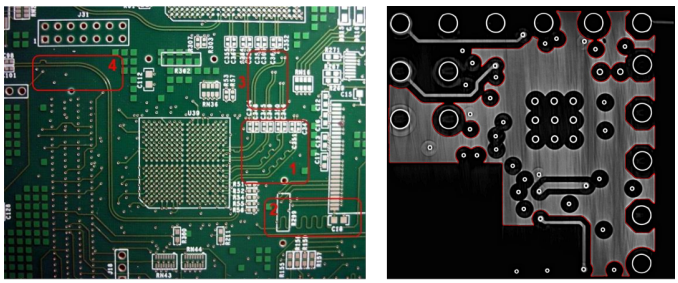

Fig. 1: (a) Examples of serpentine traces in a PCB design; (b) Random geometric shape of copper conducting plane.

resents the general population of samples. Thus, its scalability and generalizability across PCB designs are questionable.

This paper presents a method that utilizes a combination of image processing, computer vision, and graph-theoretic algorithms for the effective and accurate segmentation of traces, conductive copper plating, and defined copper pour cutouts of a PCB design imaged through X-ray CT. Our approach, which does not use DL, generalizes to any geometry yet still operates in a minimally supervised manner. The remainder of the paper is organized as follows: Section II provides a background on the different algorithms we employ. Section III discusses our methodology in depth and Section IV provides a qualitative and quantitative analysis of the results. Lastly, Section V concludes the paper.

II. Background

A. Otsu's Thresholding

Otsu's thresholding performs segmentation using a single intensity threshold to separate the image's pixels into two classes [12], [13]. This threshold is automatically computed by minimizing intra-class variance while at the same time maximizing inter-class variance. The algorithm performs well when the statistical distribution of the image exhibits bimodal behavior (two peaks). If the distributions are unimodal or the variances of the object and background are large compared to the relative mean difference, the algorithm fails.

B. Ridge Detection

Ridges in computer vision refer to a smooth function of two variables that contains a set of curves whose points are made to be below the local maxima of the function in at least one dimension in one or more ways [14], [15]. For a function of D variables, the ridges extend to a set of curves whose points are local maxima in $D-1$ dimensions. While often confused for edge detection, the main difference between traditional edge detection and ridge detection is that ridge detection is primarily focused on capturing the major axis of symmetry of an elongated object. In contrast, the primary focus of edge detection is simply to capture the boundary of the object.

C. Super-Pixel Algorithm

Superpixels are groupings of pixels based on perceptual similarity or some other heuristic-based method of grouping [16]. Superpixels are useful since they allow for the representation of regions of an image with more information than one would typically achieve utilizing single pixels. Furthermore, they provide a method of expressing dense image data in a convenient, compact, and computationally efficient manner. The two main approaches to creating and aggregating pixels into superpixels are graph-based and clustering-based approaches.

D. Connected Components

In graph theory, a connected component is a component of an undirected graph whose induced subgraph contains pairs of vertices that are connected to each other by paths but not connected to any other additional vertices in the rest of the graph. This definition is commonly extended into the image processing and computer vision fields for labeling. Each subset of connected components is uniquely labeled according to a defined heuristic, most commonly pixel intensity value [17].

E. Graph Cycles

A cycle in a graph is a non-empty trail whose path contains non-repeating nodes other than the beginning and end node [18]. Cycles can also refer to an element of the cycle space of a graph where there is a cycle for each ring of nodes or coefficient field in a graph. Furthermore, a cycle basis of an undirected graph is the set of simple cycles that forms a basis of the cycle space of the graph. This basis ($\mathbf{B}$) is a minimal set of cycles ($\mathbf{C}$) that allows any even degree subgraph/cycle within the original graph to be expressed uniquely as a symmetric difference of the cycles in the basis [19].

III. Automated Trace and Copper Conducting Plane Extraction Methodology

In order to detect the appropriate copper planes and traces from each layer of a design, it is imperative to first consolidate the various slices into their respective fused layer images. Therefore, we leverage our earlier work dedicated to slice to layer identification [20] to determine what slices belong to what layer in a design, fuse them together, and then proceed with the following steps for automated trace and copper conducting plane extraction, outlined in Figure 2.

A. Pre-Processing

1) Adaptive Otsu Thresholding: To achieve an automated solution that generalizes across a variety of systems, one needs an automated method of calculating these thresholds as opposed to (manual) trial and error. For this reason, we propose utilizing Otsu's Thresholding since it is a method that utilizes the intra and inter-class statistics between foreground and background to automatically calculate the threshold in question [12]. Nevertheless, for our particular application, there are specific nuances for effective pre-processing. In particular, typical noise sources from the X-ray CT process include high impedance materials and blurring artifacts [21] which have drastic impacts on said distributions used throughout Otsu's Thresholding. There are often instances of very subtle noise injected into the resultant X-rays from the X-ray CT process to a more minute degree. This noise is nearly invisible to the naked eye but can skew the distributions, particularly around weak edges and boundaries, to distort the calculated threshold. Empirical analysis across various slices, layers, and data samples has shown that these pixels typically lie slightly above the usual floor of pixel intensities in an image. Therefore, we employ an initial subtle noise threshold to remove these subtle noise pixels without drastically distorting the rest of the image.

In addition, when discussing the manufacturing of traces and copper conducting planes in a PCB design, it is typically a matter of coordinate information. In particular, depending on the requirements, the copper in a PCB design is plated to a substrate, and disconnected regions, outlined in software as defined polygon pour cutouts [10], are etched away to expose the design of the board [22]. Therefore, to ensure reproducibility, the pre-processing output must resemble pre-manufacturing information. To achieve this, we work under

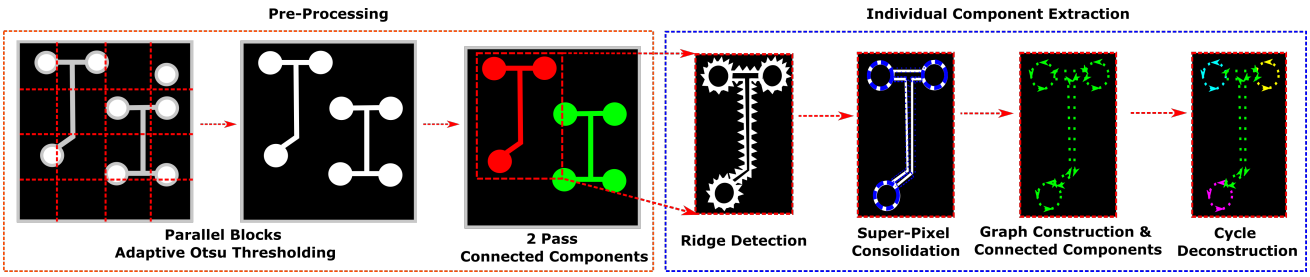

Fig. 2: Graph cycle based trace extraction block diagram.

the assumption that we have via location information across every layer a priori. Whether through automated detection [6], a subject matter expert manually extracted them or having the original design files. With this information, we implement a simple connectivity analysis to determine if the via in question is connected to the conducting structure in question (plane or trace). If so, the algorithm in-paints the via into the structure's texture to further improve the performance of our adaptive Otsu Thresholding process. On the other hand, if the via is not connected, we simply remove the via from the image under evaluation.

Lastly, we break up the overall layer images into a series of blocks to perform Otsu's thresholding on local image statistics as opposed to global image statistics. A key nuance of this approach is to first detect whether the region in question satisfies a bimodal distribution of pixel intensities since this is the optimal working environment for Ostu's Thresholding [13]. Otherwise, the results are sub-par in regions of texture that belong to the foreground or copper planes. Therefore, for each local block, we analyze the histogram of the intensities. If there are two or more distinct modes in the respective block, we apply Otsu Thresholding to calculate the value used for thresholding. Otherwise, we binarize the block according to the minimum non-zero intensity of the image block.

2) Two Pass Connected Components: After producing a high fidelity binarized image, we then perform connected components on the image in 2 successive passes [17]. The first pass is to detect the simplest structures in the image. Afterwards, we perform ridge detection on the individual sets of connected components [14], [15]. Ridge detection effectively eliminates the insides of a filled structure while emphasizing the boundaries to form a closed contour. The second pass is then done on the ridge detected image to detect any internal structures that are connected that are meant to be defined separately. For example, the defined polygon pour cutouts of the copper plane that encircle the traces/vias of a layer due to them being disconnected. In both passes, the algorithm screens connected components for likely instances of noise by removing those containing fewer pixels than the smallest known via. This is due to the fact that any trace connected to a via would have trace pixels in addition to the via pixels.

B. Individual Component Extraction

When utilizing connected components in the image space, there are instances with structures being connected erroneously. Therefore, we conduct analysis and characterizations of cycles in graph space to separate incorrectly connected components.

1) Super-Pixel Consolidation: Directly converting an image into graph space can result in extremely dense graphs which present speed and computational efficiency issues [23]. Here only the foreground pixels of the binarized image are grouped into super-pixels.

A byproduct of ridge detection on a binarized image is the production of "spurs" on the image structure. These spurs typically radiate in the direction of the largest consolidation of foreground pixels and lead to unintentionally connected components. Therefore, we use a series of thresholds to remove them from the set of produced super-pixels. Specifically, we first compute the average intensity of each super-pixel for every super-pixel in the ridge detection version of the image to produce the set $\mu_{\mathbf{intSP}} = (\mu_{\mathbf{intSP_1}}, \mu_{\mathbf{intSP_2}}, ..., \mu_{\mathbf{intSP_n}})$ for N super-pixels. Followed by computing the density of turned on pixels to total pixels in a super pixel for a similar set $\mathbf{dens_{intSP}} = (\mathbf{dens_{intSP_1}}, \mathbf{dens_{intSP_2}}, ..., \mathbf{dens_{intSP_n}})$ for N super-pixels where

$$dens_{intSP_n} = \frac{\#NonzeroPixels_{SP_n}}{\#TotalPixels_{SP_n}}, \tag{1}$$

but now for the binarized version of the image to account for the spurs affinity to be in the direction of the largest density of turned on pixels prior to ridge detection.

Next, both distributions of scores undergo Otsu's method [13] to determine the optimal threshold of the bimodal distributions. With these thresholds determined, we threshold the super-pixels with the following rules: (1) any super-pixels less than the threshold computed for the average intensity are to be removed, and (2) any of the remaining super-pixels whose density is greater than the otsu method threshold for density are removed as well. The reason behind this is that the higher density indicates a region where the spurs are directed towards and, thus, can be removed.

Afterward, we have a reduced set of super-pixels that consists of the inner boundaries of the ridge detected image and ignores the previously produced spurs. Yet, super-pixels that are nearby spatially can still be grouped into even larger super-pixels for a much reduced overall set of super-pixels. To do this, we employ mean-shift clustering [24] the (x, y) coordinates of every pixel in the super-pixel as the feature vectors. A kernel size of 10 is used, so only those nearby are clustered together and minimize bridging of those farther apart.

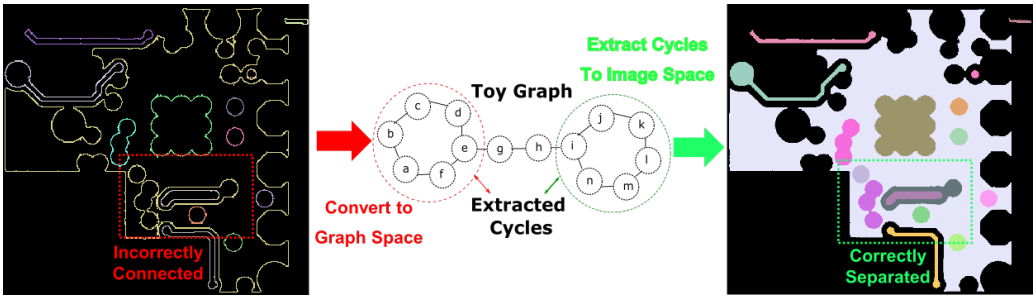

Fig. 3: Example of cycle deconstruction to separate incorrectly connected components.

2) Graph Cycle Based Deconstruction: Once the image is converted into a set of locally condensed super-pixels, we have the data primed for creating an efficient graph representation where each node in the graph corresponds to each super-pixel. Each node is connected to all locally adjacent super-pixels in image space, and each edge is weighted according to how many super-pixels are adjacent to the node in question in image space. Once the graph is fully constructed, we run connected components once again but now in graph space. In particular, we try to decompose the graph into a set of connected components by iteratively working through each node in the graph and finding the set of nodes they are connected to.

Once the connected nodes in graph space are known, we exploit the fact that a cycle in graph space is a closed loop of nodes where the only repeating nodes along the traversed path are the beginning and end nodes of the cycle [18]. We utilize this characterization because the only known geometric constraint of a trace, copper conducting plane, or defined cutout is that each will be a closed-loop in image space. Furthermore, this graph space cycle constraint allows us to separate incorrectly connected components in image space due to a few pixels of distortion. Instead, we can identify the closed-loop structures in graph space, separate them from one another, and then reintegrate them in image space for a more high fidelity reconstruction of the original design (see Figure 3).

We achieve graph cycle-based deconstruction of connected components by first breaking down the connected components into a basis of cycles [25]. This basis serves as the fundamental building blocks of the entire connected components. Therefore, any cycle that exists in the connected components can be recreated through some series of logical exclusive-or'ing of pairs of cycles in the basis [19]. We work under the assumption that the most likely correct structure in the overall connected component we are trying to extract is the overall largest cycle on the basis of cycles. This is because it is easy for noise distortions in pixel space to erroneously connect structures, but we can identify and separate structures to a much finer degree in graph space. Once we have extracted the largest cycle out of the set of connected components, we iterate and repeat this process of breaking down the remaining connected components into the basis of cycles again to extract the next largest cycle. This is repeated until convergence, where there are no longer any cycles within the remaining connected components. At the conclusion of cycle deconstruction, we once again screen the produced cycles for likely instances of noise by removing any that contain fewer nodes than the necessary number of nodes for the smallest known via under the same line of thought from earlier.

IV. Results and Evaluation

A. Experimental Setup

Our dataset is an unpopulated 6-layer PCB, referred to as RASC, with dimensions of 15mm by 15mm. The X-ray CT image acquisition parameters for the dataset were a source voltage of 75kV, source current of 573μA, pixel size of 11.30μm, a filter of 0.5mm Aluminum, exposure time equal to 38ms, and using a flat panel camera. The reconstruction parameters are a beam hardening coefficient of 43%, ring artifact of 18%, a gaussian smoothing kernel of sigma equal to 2, and a post alignment value of 7.7. After proper alignment, registration, and removal of all slices containing only air, the final size of the dataset resulted in a 3-D stack of 1357×1286×157 slices for the 6 layer board.

1) Hyper-Parameter Selection: Our algorithms are implemented with minimal reliance on parameter tuning; however, we will discuss which hyper-parameters are most critical and their values in this section. The first of those is the subtle noise threshold mentioned earlier in Section III-A. This value can change according to the resolution of the X-ray CT scan, but in general, it is good practice to utilize a value slightly above the floor of pixel intensity values but still below the minimum intensity of the foreground. In practice, this value was **10** for our RASC dataset and a reasonable assumption for most datasets. The next important parameter determines the size of the block for the parallel blocks used during Otsu's thresholding. For RASC, we divided the image into 64 non-overlapping blocks of equal size unless at the edges of the image, in which case the block size is up to the limits of the board's dimensions. Lastly, when utilizing the maskSLIC [23] algorithm, two key parameters are the number of segments and the compactness factor. We use a compactness factor of **1** and an adaptive number of segments $\#segments = \frac{\#NonZeroPixels}{3}$ to ensure at least 3 pixels in each superpixel.

B. Quantitative Evaluation

Here we provide quantitative support for the methods we have discussed above to achieve automated unsupervised extraction of traces and copper conductive planes in a PCB

Method	IoU	SSIM	Corr.	Dice
Global Otsu	.91609	.92848	.90776	.95378
Global Means	.91105	.92144	.90287	.95100
Global Triangle	.89916	.90593	.89038	.94429
Global Isodata	.91275	.92147	.90318	.95189
Parallel Blocks Otsu	.91721	.92815	.90800	.95391
Parallel Blocks Mean	.91075	.92058	.90214	.95088
Parallel Blocks Isodata	.91726	.92816	.90806	.95394

TABLE I: Evaluation of global vs. parallel blocks implementation of automatic threshold computing algorithms averaged across all layers.

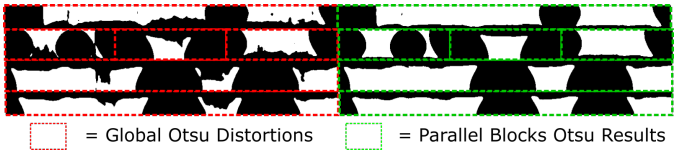

Fig. 4: Comparison of distortions due to global Otsu thresholding across a variety of layers compared to their respective parallel blocks Otsu implementation.

design. The critical stages of our framework are first pre-processing followed by the final result with the individual components in a design extracted. We utilize the following appropriate segmentation, and image quality assessment metrics: mean intersection-over-union (IoU) [26], mean DICE coefficient [27], Correlation [28] and SSIM [29] between the ground truth and predicted results. IoU measures the overlap between the ground truth image and our predicted result for each layer divided by the union of the two images. The formula for IoU is

$$IoU = \frac{M_p \cap M_{gt}}{M_p \cup M_{gt}} \tag{2}$$

where M is the mask of the predicted result (with subscript p) or ground truth (with subscript gt) per layer. Since this is a binary segmentation task, the IoU is computed for both the foreground and background separately and then averaged to provide a mean IoU score for evaluation.

The DICE coefficient, also referred to as F1-Score, is computed similarly to the IoU, and we once again prefer the mean DICE score result to quantify the performance regarding foreground and background classes equally. DICE is computed as follows

$$DICE = \frac{2 * M_p \cap M_{gt}}{Total\#ofPixels} \tag{3}$$

DICE coefficient behaves similarly to the IoU, and both are positively correlated with one another. This means that they are often in accordance with one another when evaluating model performance. However, an important difference is that the IoU penalizes instances of misclassified pixels harsher than DICE. Lastly, we utilize the common SSIM and Correlation Coefficients to evaluate the perceptual structural similarity and pixel-level similarity of our predicted mask versus the ground truth mask.

1) Pre-Processing Evaluation: Since the effectiveness of any component extraction algorithm depends on the quality of our pre-processing, we first evaluate our pre-processing's quality using a series of image quality metrics for a series of popular thresholding algorithms. Thresholding is utilized to maximize the effectiveness of the ridge detection, and connected components algorithms, where noisy or highly textured image data can result in sub-optimal performance. As mentioned earlier in Section III-A1, it is important for the thresholding method to adapt its threshold based on the data that is being presented as opposed to any hard thresholds. The variance across designs and imaging conditions makes this a necessity. Therefore, when comparing our proposed approach utilizing Ostu's Method in parallel blocks for local statistical analysis to compute respective thresholds, we compare against other automated threshold computation methods. Namely, the Triangle [30], Mean [31], and Isodata [32], [12] methods in addition to the earlier mentioned Otsu's thresholding. We first perform each method on each layer in a global fashion such that the threshold computed is applied globally and then compare to a parallel-blocks implementation of the top three performing global methods.

The results utilizing the aforementioned metrics to evaluate the pre-processing are listed in Table I. For evaluating the pre-processing, the binarized ground truth labeled by an expert is the ideal result if all connected vias were in-painted to their respective traces or copper planes, and all non-connected vias were removed from the image as discussed earlier. The predicted thresholded image used in conjunction with the ground truth to compute these metrics is the result of our in-painting, based on a simple non-supervised connectivity analysis. The respective scores are computed for every layer and then averaged to provide a comprehensive evaluation across the board. Since each layer has its own nuances (weak edges, noise, spatial distribution/proximity of features, etc.), the average score provides a holistic assessment of the best overarching approach. We begin the evaluation by first applying globally to see how the methods react to the earlier mentioned nuances with a singular threshold value. We can see from the table that in terms of global performance, the top three performing were Otsu, Mean, and Isodata, with Otsu performing the best globally and second-best overall.

While the results were adequate for some layers, when analyzing the results qualitatively at a layer level, there were significant distortions at the boundaries of layers containing copper planes. These cause them to be irregularly shaped or lose connectivity in a practical sense (see Figure 4). Consequently, they can create a problem for subsequent stages, where it is necessary to produce CAD software interpretable polygonal shapes for polygon pours of copper planes. This is due to the statistics at boundaries or other weak edges in the layer having different statistics locally than in relation to the global statistics present throughout the image. Therefore, we employed a parallel blocks approach (see Section III-A1) to each method to analyze and perform the same thresholding but with respect to the local statistics instead. Here we see a drastic improvement for the Isodata and Otsu methods in relation to their global counterparts. With both methods being nearly indistinguishable visually from one another and very similar quantitatively. The only artifacts incorrectly thresholded in these parallel implementations are those that remain due to vias not being fully removed during in-painting/removal. When analyzing each layer qualitatively, the previously mentioned distortions are now resolved for a

Pre-Proc.	Method	IoU	SSIM	Corr.	Dice
Global Otsu	Canny Edge + CC.	.895	.9836	.8621	.9217
	Ridge + CC.	.9095	.9951	.8834	.9379
	Graph Cycle	.9057	.9911	.8813	.9364
Blocks Iso.	Canny Edge + CC.	.9011	.9844	.8720	.9263
	Ridge + CC.	.9000	.9947	.8648	.9279
	Graph Cycle	.9145	.9913	**.8969**	.9443
Blocks Otsu	Canny Edge + CC.	.9010	.9844	.8719	.9262
	Ridge + CC.	.9001	.9947	.8649	.9280
	Graph Cycle	**.9166**	**.9954**	.8952	**.9453**

TABLE II: Average component extraction evaluation across top performing thresholding approaches for all layers.

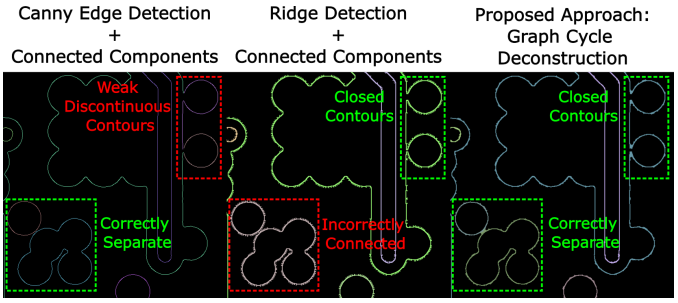

Fig. 5: Comparison of component extraction methods.

high fidelity binarization suitable for the subsequent individual component analysis and extraction.

2) Individual Component Extraction Evaluation: With data prepared for the second stage of analysis (Section III-B), the individual traces, copper planes, and defined cutouts for said planes can be extracted. Each of these components in a PCB design is essential in maintaining signal connectivity throughout each discrete layer or, in the case of defined cutouts, ensuring what is not supposed to be connected at a particular layer remains that way. Therefore, it is important that after all of the earlier discussed pre-processing, we can extract each of the necessary components with as high fidelity as possible. As mentioned earlier, we utilize a graph cycle deconstruction approach to separate out the individual components from the aforementioned binarized data. This holds since there are instances with components connected in pixel space that should be separated in the actual reproduced design. One such example is cutouts of a polygon pour's copper plane that are very spatially near one another but still defined separately. While their connectivity in the reproduced design may not functionally change the design in a reproduced sample, it is still a noticeable design change from the original. However, another instance could be two traces being close to one another and connected in pixel space when they should not be. Here they must be separated prior to manufacturing, otherwise inducing a short circuit in the newly reproduced design. Therefore, our approach for enforcing graph cycle deconstruction is applied to ensure components that are meant to be disconnected remain so in as much of an unsupervised fashion as possible in the absence of data volume for model/data-driven approaches.

To emphasize the ability and necessity for this graph-cycle-based deconstruction approach, we compare it with two other common approaches for detecting the desired components that rely on no domain knowledge: (1) Ridge Detection of the binarized image followed by Connected Components with no further post-processing (2) Canny Edge Detection of the binarized image followed by Connected Components. Ridge detection (as mentioned earlier in Section III-A2) is particularly effective at producing strong-edged closed contours that remove the interior or valleys of an image. This makes it ideal for performing connected components afterward to detect each desired trace, copper plane, and defined cutout contour, albeit at the likelihood of incorrectly connected components that should be separated due to spurious artifacts. Canny Edge Detection [33] of the binarized images will also produce the desired contours of each component that can also be extracted via connected components. These produced contours will be thinner and absent the spurious artifacts that incorrectly connected components in ridge detection but not guaranteed to be closed or with as strong edges as ridge detection. Not to mention Canny's need for a user-defined upper and lower threshold employed during hysteresis thresholding that is unlikely to generalize across a vast amount of boards and layers. Therefore, in our implementation of Canny Edge detection for comparison, we automatically calculate the upper and lower thresholds in the following manner unique to each layer:

$$V = \text{median}(Im_{binarized}) \tag{4}$$

$$Th_L = \max(0, (1-\sigma) * V) \tag{5}$$

$$Th_H = \min(255, (1+\sigma) * V) \tag{6}$$

where σ is a user-defined value that empirically was used as $\mathbf{0.33}$, and $\mathbf{Th_{H/L}}$ are the required high and low thresholds for Canny Edge detection, respectively. We then employ ridge detection with connected components, Canny edge detection with connected components, and graph cycle deconstruction methods, each with the top three threshold algorithms from Table I (Global Otsu, Parallel Blocks Otsu, Parallel Blocks Isodata) to find the most effective overall methodology.

To effectively compare the results across the methods, we utilize the same evaluation metrics as discussed to highlight the correct detection and the correct localization of traces, copper planes, and defined cutouts. Specifically, for each layer's ground truth component's we compare against each respective layer's predicted extracted components in a masked image of that component only. Here the highest-scoring pair is the correct pairing and score. Finally, the scores for each set of components are averaged to provide a respective accuracy score for the layer. The scores reported in Table II are the scores for each layer averaged to provide a holistic evaluation once again in the presence of nuances for each layer. This evaluation method properly penalizes incorrectly connected components (False Negatives/Positives) instead of an image with all components present and filled that would score correctly and incorrectly connected components equally.

Thus, when analyzing the scores in Table II, we can see the effects of our approach in comparison to the less nuanced approaches for the top three performing thresholding methods. We can see that for the top two performing threshold methods, the graph cycle deconstruction approach performed best in almost every category, especially in combination with the parallel block's implementation of Otsu's thresholding. Whose

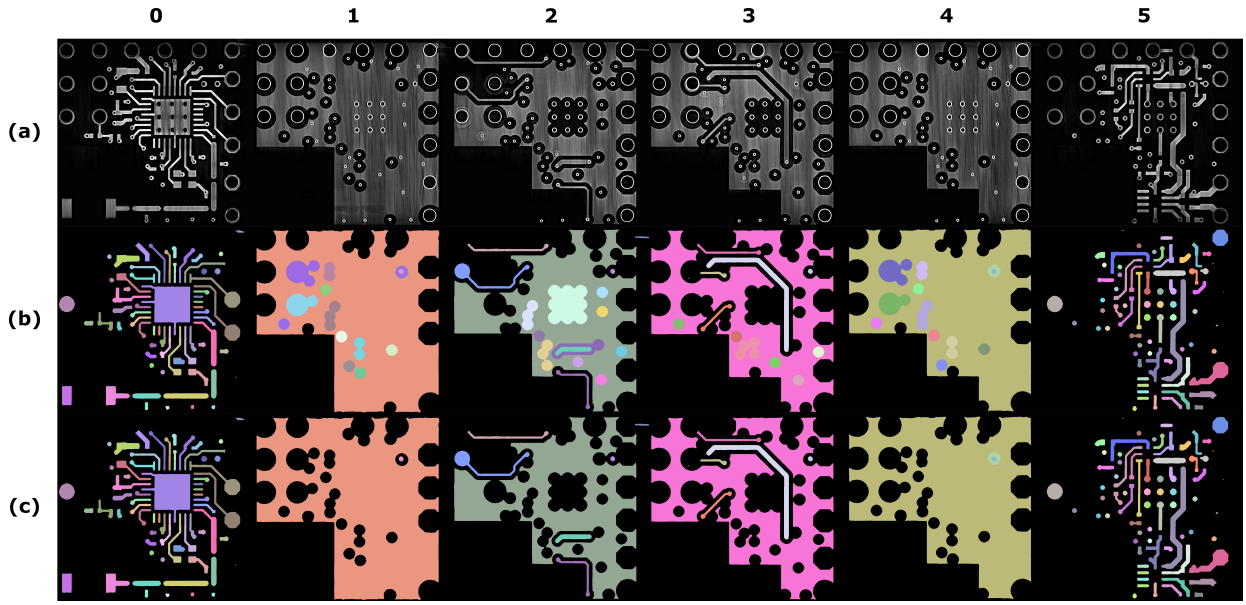

Fig. 6: (a) Original X-Ray CT design per layer; (b) Trace and copper plane detection results for RASC prior to etching; (c) Trace and copper plane detection results for RASC post-etching.

scores are a fair amount higher than the next closest method, parallel blocks Isodata with graph cycle deconstruction, outside of for correlation. It is important to mention that our discussed approach is the optimal performing in a holistic analysis, although there are, of course, instances where the other approaches may have outperformed ours in a particular layer. For example, Canny and ridge detection methods performed well for the first and last layers due to their relatively straightforward trace and pad structures in the absence of copper planes and defined cutouts. But these methods either produced discontinuous edges/contours (Canny) or connected defined cutouts incorrectly (ridge detection) when utilized for the other layers, Figure 5. Emphasizing the need for holistic analysis, we have discussed providing a method that performs well for a particular layer and all layers and their nuances.

C. Qualitative Evaluation

Figure 6 illustrates the results for all 6 layers and compares our optimal performing results to what we have expected from the original design. In the middle row, we provide a pre-etching version that contains traces, copper planes that would be defined via a software's polygon pour, and the defined cutouts of said copper pours to disconnect any regions not meant for connectivity in the layer. Each of the individual component extracted by our methodology is depicted in a unique color. In the bottom row, the post-etching version is shown that presents the likely post-manufacturing reproduction from our results and their similarity to the original design.

For layers 0 and 5 -of the 6 layers- RASC board, there are only traces and no copper conducting planes. When comparing the extracted result visually to our expected result from the original scans, we can see that all traces are extracted correctly. Albeit, there are some vias whose contours were extracted incorrectly as traces. However, this is not consequential since the trace information can later be consolidated with via detection results. These vias and other small artifacts are detected incorrectly due to artifacts during the pre-processing binarization and naïve in-painting when analyzing connected versus non-connected traces.

When looking at layers 1 through 4, the impacts of copper conductive planes and our efforts to correctly segment them out in relation to traces and the earlier mentioned artifacts can be observed. In particular, determining in-paint connected vias in the copper planes was crucial and effective to get the full copper plane structure that would later be carved/etched to make way for the other traces and vias on the layer that are disconnected from the plane. Additionally, we can see the effect of the entire graph cycle-based deconstruction process to separate out connected components in the image space due to the spurs from ridge detection. While the connectivity of etched/carved regions may not alter functionality since they are surrounding components disconnected from the copper planes, we want as high fidelity a reproduction as possible to mimic manufacturing specifications as close as possible.

V. Conclusion

This paper provided a methodology for automated, unsupervised extraction of both traces and copper conducting planes in a PCB design. The usefulness of cycles in graph space for separating incorrectly connected components to produce high fidelity results in image space was demonstrated. This new result and the earlier work in automated via detection can be integrated into one comprehensive workflow, essential for designing trusted PCBs. In this regard, a direction for future research is to translate our results from the pixel domain to a format compatible with the software used by most PCB manufacturers, e.g., Gerber format, i.e., vectorization.

Acknowledgment

This paper is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1315138 and DGE-1842473.

REFERENCES

[1] S. Ghosh, A. Basak, and S. Bhunia, "How secure are printed circuit boards Trojan attacks?" *IEEE Design and Test*, vol. 32, no. 2, pp. 7–16, 2015.

[2] M. McGuire, U. Ogras, and S. Ozev, "PCB Hardware Trojans: Attack Modes and Detection Strategies," in *Proceedings of the IEEE VLSI Test Symposium*, vol. 2019-April. IEEE Computer Society, 4 2019.

[3] J. B. Robertson and M. B. B. Riley, "The Big Hack: Statements From Amazon, Apple, Supermicro, and the Chinese Government," 2018. [Online]. Available: https://www.bloomberg.com/news/articles/2018-10-04/the-big-hack-amazon-apple-supermicro-and-beijing-respond

[4] S. K. I. S. Moore, "This Tech Would Have Spotted the Secret Chinese Chip in Seconds," *IEEE Spectrum*, 2018. [Online]. Available: https://spectrum.ieee.org/riskfactor/computing/hardware/this-tech-would-have-spotted-the-secret-chinese-chip-in-seconds

[5] N. Asadizanjani, M. Tehranipoor, and D. Forte, "PCB Reverse Engineering Using Nondestructive X-ray Tomography and Advanced Image Processing," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 7, no. 2, pp. 292–299, 2017.

[6] U. J. Botero, D. Koblah, D. E. Capecci, F. Ganji, N. Asadizanjani, D. L. Woodard, and D. Forte, "Automated Via Detection for PCB Reverse Engineering," in *ISTFA 2020: Papers Accepted for the Planned 46th International Symposium for Testing and Failure Analysis*, vol. 83348. ASM International, 12 2020, pp. 157–171.

[7] "Timing Relationship between Signals." [Online]. Available: http://referencedesigner.com/books/si/adding-delay-intentionally.php

[8] "Board Design Resource Center." [Online]. Available: https://www.intel.com/content/www/us/en/programmable/support/support-resources/support-centers/board-design-guidelines.html

[9] "Serpentine Routing Tips to Snake in Your Tuned Traces — Advanced PCB Design Blog — Cadence." [Online]. Available: https://resources.pcb.cadence.com/blog/2019-serpentine-routing-tips-to-snake-in-your-tuned-traces

[10] P. C. M. Loughhead, "Polygon Pour." [Online]. Available: https://documentation.circuitmaker.com/display/CMAK/PCB_Obj-PolygonPour((Polygon+Pour))_CM#Polygon+Pour-Summary

[11] K. Qiao, L. Zeng, J. Chen, J. Hai, and B. Yan, "Wire segmentation for printed circuit board using deep convolutional neural network and graph cut model," *IET Image Processing*, vol. 12, no. 5, pp. 793–800, 2018.

[12] B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, p. 146, 1 2004. [Online]. Available: https://www.spiedigitallibrary.org/terms-of-use

[13] N. Otsu, "THRESHOLD SELECTION METHOD FROM GRAY-LEVEL HISTOGRAMS." *IEEE Trans Syst Man Cybern*, vol. SMC-9, no. 1, pp. 62–66, 1979.

[14] J. Damon, "Properties of ridges and cores for two-dimensional images," *Journal of Mathematical Imaging and Vision*, vol. 10, no. 2, pp. 163–174, 1999. [Online]. Available: https://link.springer.com/article/10.1023/A:1008379107611

[15] T. Lindeberg, "Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction," *Journal of Mathematical Imaging and Vision*, vol. 3, no. 4, pp. 349–376, 11 1993.

[16] M. S. Nixon and A. S. Aguado, "Region-based analysis," in *Feature Extraction and Image Processing for Computer Vision*. Elsevier, 1 2020, pp. 399–432.

[17] M. B. Dillencourt, H. Samet, and M. Tamminen, "A General Approach to Connected-Component Labeling for Arbitrary Image Representations," *Journal of the ACM (JACM)*, vol. 39, no. 2, pp. 253–280, 1 1992. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/128749.128750

[18] "Graph Theory and Its Applications, Second Edition - Jonathan L. Gross, Jay Yellen - Google Books." [Online]. Available: https://books.google.com/books?id=-7Q_POGh-2cC&pg=PA197#v=onepage&q&f=false

[19] "Graph Theory: Springer Graduate Text GTM 173 - Reinhard Diestel - Google Books." [Online]. Available: https://books.google.com/books?id=eZi8AAAAQBAJ&pg=PA23

[20] U. J. Botero, N. Asadizanjani, D. L. Woodard, and D. Forte, "A Framework for Automated Alignment and Layer Identification of X-Ray Tomography Imaged PCBs," in *GOMACTech*, San Francisco, 2020.

[21] U. J. Botero, R. Wilson, H. Lu, M. T. Rahman, M. A. Mallaiyan, F. Ganji, N. Asadizanjani, M. M. Tehranipoor, D. L. Woodard, and D. Forte, "Hardware Trust and Assurance through Reverse Engineering: A Survey and Outlook from Image Analysis and Machine Learning Perspectives," *arXiv*, 2 2020. [Online]. Available: http://arxiv.org/abs/2002.04210

[22] S. A. Technology, "PCB Design Manufacturing Process." [Online]. Available: https://www.sierraassembly.com/blog/stripping-and-etching-process-of-pcb/

[23] B. Irving, "maskSLIC: Regional Superpixel Generation with Application to Local Pathology Characterisation in Medical Images," 6 2016. [Online]. Available: http://arxiv.org/abs/1606.09518

[24] K. G. Derpanis, "Mean Shift Clustering," Tech. Rep., 2005.

[25] K. Paton, "An algorithm for finding a fundamental set of cycles of a graph," *Communications of the ACM*, vol. 12, no. 9, pp. 514–518, 9 1969. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/363219.363232

[26] D. M. W, "EVALUATION: FROM PRECISION, RECALL AND F-MEASURE TO ROC, INFORMEDNESS, MARKEDNESS & CORRELATION," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011. [Online]. Available: http://dspace.flinders.edu.au/dspace/http://www.bioinfo.in/contents.php?id=51

[27] F. Ge, S. Wang, and T. Liu, "New benchmark for image segmentation evaluation," 2007.

[28] J. Li and W. Dai, "Image quality assessment based on the correlation coefficient and the 2-D discrete wavelet transform," in *Proceedings of the 2009 IEEE International Conference on Automation and Logistics, ICAL 2009*, 2009, pp. 789–793.

[29] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multi-scale structural similarity for image quality assessment," in *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, vol. 2, 2003, pp. 1398–1402.

[30] G. W. Zack, W. E. Rogers, and S. A. Latt, "Automatic measurement of sister chromatid exchange frequency," *Journal of Histochemistry and Cytochemistry*, vol. 25, no. 7, pp. 741–753, 1 1977. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/25.7.70454

[31] C. Glasbey, "An Analysis of Histogram-Based Thresholding Algorithms," *CVGIP: Graphical Models and Image Processing*, vol. 55, no. 6, pp. 532–537, 11 1993.

[32] T. W. Ridler and S. Calvard, "PICTURE THRESHOLDING USING AN ITERATIVE SLECTION METHOD." *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-8, no. 8, pp. 630–632, 1978.

[33] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.