EE368: Reverse Engineering of Printed Circuit Boards

Ben Johnson - benj2@stanford.edu

I. Introduction

A. Purpose

For this project, a program was written which can automatically reverse-engineer a netlist of a one- or two-layer printed circuit board (PCBs) from photos of the board. Reverse-engineering a printed circuit board (PCB) is useful for purposes such as repairing equipment for which component-level documentation is not available, integrating a poorly-documented board into a system, and for identifying obsolete parts. This process can be performed manually by either checking for continuity with a meter or by tracing over the images by hand, but an automated process would be faster and more reliable.

The program is written in C++ using the OpenCV ([4]) library. It is an offline, non-interactive process, but various intermediate images are presented for testing and development purposes.

B. Scope and Assumptions

The program takes as input a photograph of each side of the board (or the copper-bearing side in the case of a one-layer board). The two photographs have approximately the same scale (e.g. in dots per inch), have been cropped to include only the board, and show the board in approximately the same orientation except that the board was flipped horizontally to take the bottom photograph.

The program produces as output a netlist which describes the connections between components.

The board is unpopulated so that no components obscure any traces. It has no inner layers so that all features can be seen from the outside of the board. The board does not have soldermask or silkscreen. Only copper is visible in the photos of the board.

A library of component footprint templates is available which represents all components that may be installed on the board. The templates have the same scale as the PCB images.

C. PCB Structure

PCB's consist of an insulating substrate, often a fiberglass composite such as FR-4, with at least one layer of etched copper attached to it. Copper may be present on one or both sides of the substrate. Inner layers can be fabricated by bonding multiple substrates together, but since imaging of inner layers is difficult and requires either delaminating or X-raying the board ([**?**]), boards with inner layers are outside the scope of this project.

On top of the copper there may be a soldermask, which is an insulating layer that defines pads where components attach, protects traces from inadvertent contact with solder, and provides a high-contrast background for silkscreen markings. Soldermask is optional and is often not present on low-density, low-cost boards. Soldermask is most commonly green, but can be made in many colors. This project assumes that no soldermask is present.

Silkscreened markings are often used to indicate component location, orientation, and identity. Silkscreen is most commonly applied on top of soldermask. Since silkscreen can obscure copper features, this project assumes that no silkscreen is present.

The presence of soldermask could be helpful for this project, since it would allow pads to be found more easily and would allow pixels to be classified as substrate, copper, or pad. This would require a somewhat different processing flow. Boards with soldermask are outside the scope of this project because boards with soldermask typically have silkscreen, which can make detection of copper problematic.

Surface-mount components may be present on either side of a two-layer board. Through-hole components are detected only on the top layer, although the actual assembly of the board may differ.

For two-layer boards, all holes are assumed to be plated so that they connect the top and bottom layers. Through-hole pads are holes which are part of components, while vias are holes which stand alone. This program does not distinguish between through-hole pads and vias, and considers both to be drills.

II. Process

All steps below except II-E are performed separately for each layer. The last step merges the results from the two layers into a single netlist.

A. Image Preparation

The images should be cropped manually so that only the board is visible. The photos should be taken with the board on a dark background so that any background visible through holes does not look like copper. An example cropped image is shown in Figure 1. This image has not been modified except for cropping.

Later processing steps start from a binarized image. To produce this image, the color input image is converted to grayscale and then thresholded.

To convert to grayscale, each pixel's color value is projected onto a line in RGB space. The position along this line becomes the pixel's new gray level. The direction of the line is found by applying the K-means algorithm to the RGB pixel values to classify them into two groups corresponding to board and

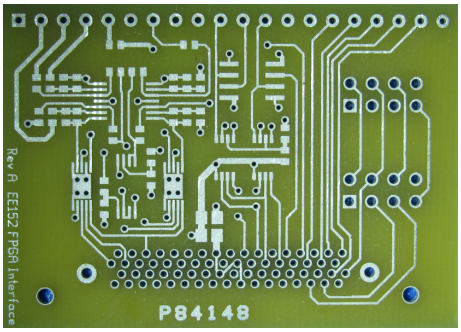

Figure 1. Color Input Image.

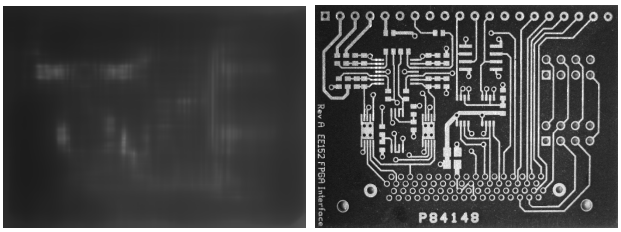

Figure 2. Lighting estimate (left) and unlit grayscale image (right).

copper colors. Since there may be lighting variation, the classification of pixels is not used to threshold the image. The centers of the clusters found by this classification are the two points which define the line onto which pixel values are projected to find grayscale values. This results in a high-contrast grayscale image regardless of the actual colors present in the image. For example, a board built with tin-plated copper on an FR-4 substrate would appear silver on yellow, while unplated copper on phenolic would be closer to pink on dark brown. The sign of the grayscale direction is chosen so that darker pixels in RGB space map to lower grayscale values.

Illumination may vary across the image. Note that in Figure 1, the upper left corner is darker than the lower right corner. To correct for this, the lighting is estimated by applying a median filter with a large window (1/10 the image width). The resulting lighting image is subtracted from the grayscale image to produce an unlit image which has less variation than the original image, and which is more suited to a global threshold (Figure 2).

Finally, Otsu's method ([5]) is used to find a global threshold for the image to classify each pixel as PCB substrate (black) or copper (white). Holes in the board will appear primarily black because of the dark background (Figure 3). Opening and closing morphological operations are performed to reduce noise in the thresholded image.

B. Finding Nets

Each connected region in the thresholded image is considered a net. For two-layer boards, nets are found on each layer independently and merged in a later step. The OpenCV function `cv::findContours` is used to find the boundaries of connected regions and any holes within them. Each hole is likely to correspond to a drilled hole in the PCB, so these holes are stored for later use when merging two-layer boards.

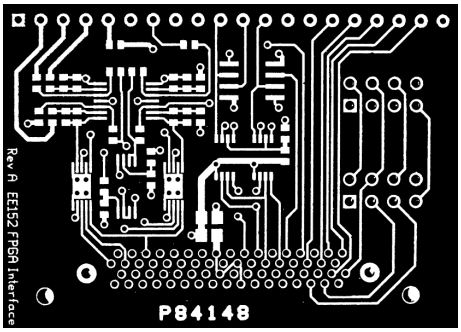

Figure 3. Thresholded image.

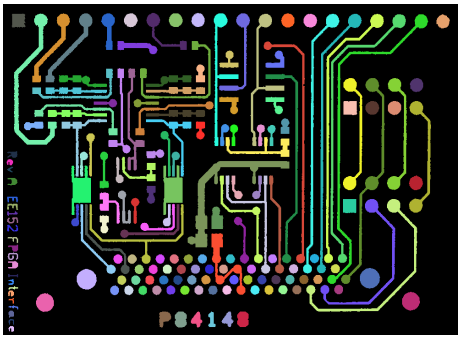

Figure 4. Nets.

Nets are sequentially numbered so that they can be identified in the output netlist. Net numbers are unique across all layers.

C. Finding Components

Components are found by comparing each footprint in a library of component templates to the thresholded image. Each template is a grayscale image with the same scale as the board image. A pixel in a template may be black (0), gray (1-254), or white (255). A black pixel indicates a location where no copper is expected, a white pixel indicates a location where copper is expected, and a gray pixel is ignored. A minimum fraction of the white and a minimum fraction of the black pixels must both be found with the template in a particular location for the component to be detected in that location. Considering the white and black pixels separately prevents a large number of pixels of one color from reducing the influence of the other color.

Since the template is compared directly against the thresholded image, the template must be in the same orientation as the image. To allow for multiple component orientations, a duplicate, rotated by 90°, of each template is made when the templated loaded. Most templates are insensitive to rotations of 180°, so no special handling is required in those cases, but for asymmetrical components an additional rotated copy would need to be created.

To reduce spurious component detections, it is helpful to add black pixels to a template to exclude certain situations.

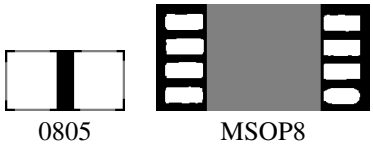

Figure 5. Typical component templates.

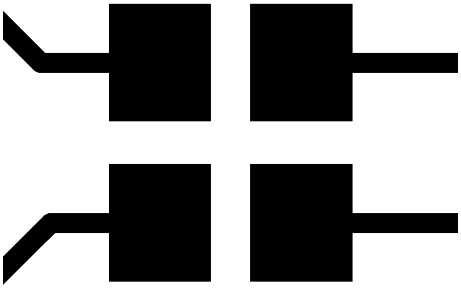

Figure 6. Example of ambiguous layout.

For example, there is room for a trace to be routed between the pads of an 0805 capacitor, but few boards are built like this. The template in the test data has black, rather than gray, pixels in this area to reduce the number of 0805 footprints detected in ambiguous situations.

Figure 5 shows some typical component templates. Note the large gray area under the MSOP8 template: this allows copper traces to pass under the component without affecting the match.

White regions in a template correpond to a pad, which is an electrical connection to a component. The center of each pad is stored and will be used to generate the netlist later.

Component locations may sometimes be ambiguous. Figure 6 shows a layout designed for two 0805 components, but it is not clear whether the components should be oriented horizontally or vertically. In this case, the program would detect all four possible components and it would be the user's responsibility to determine which two are correct and to delete the other two.

The location of pin 1 of a component is not in general visible in the copper. For through-hole components, it is common for pin 1 to be indicated by a square pad while other pins have round pads, but this convention is not always followed or clearly visible. For surface mount components there is typically no indication in copper of component orientation, so the pin numbers produced by this program will likely not match those used in the original schematic.

For each detected component, all pads from the template are copied to the component and translated into their final locations in the image.

D. Generating a Netlist

Once nets and components are found and component pad locations are determined, a netlist can be generated. For each pad on each component, the net containing the pad's center point is found and the component and pad are added to a list of connections on that net. Since a pad is a connected copper region, each pad is part of exactly one net. The netlist for a one-layer board can now be printed. Each line in the netlist takes the form:

net_name: component1−pad1 component2−pad2

A netlist of the same form is generated by many schematic capture tools. While the net, component, and pad names are different, the connectivity of this netlist should match that generated from the correct schematic for the board.

In the case of a two-layer board, the second layer is processed in the same way as the first and a merging process will generate the complete netlist.

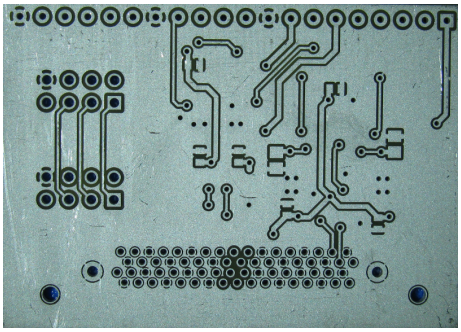

Figure 7. Bottom layer of test board.

E. Merging Layers

For a two-layer board, the netlist of each layer is found independently. The netlists are then merged by finding corresponding nets on each side of each drill. The merging process proceeds follows for each drill:

1) Find the top-layer net containing this drill.
2) Find the corresponding drill on the bottom layer.
3) Find the bottom-layer net containing that drill.
4) Move the set of pads connected to the bottom net to the top net's list of connections.

If no corresponding bottom-side drill can be found, the drill is assumed to be a false detection and is ignored. After all drills have been processed, the union of the two netlists is the complete netlist.

III. Results

The program was tested with a two layer board, of which the top layer is shown in Figure 1 and the bottom layer is shown in Figure 7.

Templates for all surface mount parts and the 5x2 pin headers were generated. The program was able to correctly detect all nets (Figure 4) and components on the top layer (except for the two connectors without templates). The detected components are shown in Figure 8. Three extra, overlapping 0805 components were detected where components are close together.

Surface-mount components are correctly detected on the bottom layer, but the merging process for two-layer boards

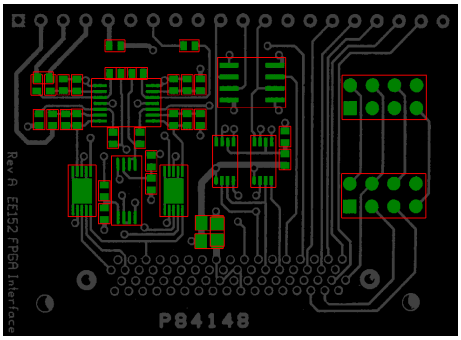

Figure 8. Component detection results.

does not work. The program does not successfully match bottom-layer drills to top-layer drills. This appears to be an implementation error.

The top-layer netlist has many nets with only one pad because many traces lead only to vias. The two-layer merged netlist has numerous errors.

IV. Future Work

Merging netlists from the two layers needs to be fixed.

The program would benefit greatly from a user interface. There are a few parameters such as minimum areas which should be easily modifiable by the user. There needs to be a way to graphically select spurious components for deletion.

The method of detecting components is fairly reliable, but slow and inflexible. While most components are rotated by multiples of $90°$, some board contain components at arbitrary orientations. A method of detecting components regardless of orientation would be valuable. The template matching approach is slow, and takes a few seconds per template per layer. A complex board would take several minutes to analyze. Heuristics could be used to find regions of interest in which to search for components, such as ignoring blank areas of the board.

Automatically generating a schematic from the netlist would be very helpful. In general, this is a hard problem, but an interactive interface could allow a user to label and reorganize components and nets and to select from a number of schematic layout strategies.

The library of component templates should be automatically scaled to match the input images based on a physical measurement made by the user. Automatically generating templates from files used in PCB layout software would allow a large number of components to be detected with little user intervention.

References

[1] Deno, S.; Landis, D.; Hulina, P.; Balasubramanian, S., "A rapid prototyping methodology for reverse engineering of legacy electronic systems," Rapid System Prototyping, 1999. pp.222,227, Jul 1999

[2] Mat, R.C.; Azmib, S.; Daudc, R.; Zulkiflid, A.N.; Ahmade, F.K., "Reverse engineering for obsolete single layer printed circuit board (PCB)," Computing & Informatics, 2006. ICOCI '06. pp.1,7, 6-8 June 2006

[3] Longbotham, H.G.; Ping Yan; Kothari, H.N.; Jun Zhou, "Nondestructive reverse engineering of trace maps in multilayered PCBs," AUTOTESTCON '95. Systems Readiness: Test Technology for the 21st Century. pp.390,397, 8-10 Aug. 1995

[4] Bradski, G., "The OpenCV Library," Dr. Dobb's Journal of Software Tools (2000). http://opencv.org/.

[5] Otsu, Nobuyuku. "A Threshold Selection Method from Gray-Level Histograms," Systems, Man and Cybernetics, IEEE Transactions on, vol.9, no.1, pp.62,66, Jan. 1979