Breadboard to Schematic

Catherine Olsson
MIT
catherio@mit.edu

Michele Pratusevich
MIT
mprat@mit.edu

Abstract

TODO: Write the abstract.

1. Introduction

Breadboards are an important tool for do-it-yourself hardware designers to quickly test circuit systems. They are easy to assemble, easy to change, and easy to test, so are the tool of choice for the first prototype of many hobbyists and students. A programmatically-drawn, or more likely hand-drawn, schematic diagram is used to prototype the circuit board, but if the circuit is going to be used for high-speed, low-noise, or multiple-production applications, a printed circuit board (PCB) is much more desirable than a breadboard. Our approach to solve this problem was a vision-based tool designed to go from a picture of a breadboard circuit to a file written in a PCB-ready format given by Eagle. Through a variety of approaches, we found the task too large and have implemented pieces of the full pipeline. Them main goal of the project was to transform the information about the circuit that we had in image form (pixel representation) to a different more general representation (component representation).

2. Related Work

2.1. Properties of Breadboards

When solving vision-based problems, it is critical to know your system properties to better develop algorithms customized to the strengths and weaknesses of a particular application. Photos of breadboards have a number of interesting properties that can be useful in segmentation, identification, and restructuring tasks. A number of visually-interesting properties are listed here:

- A single component on a breadboard, such as a wire, LED, or chip is often a single color.
- The colors of the various components are often easily visually distinguishable.

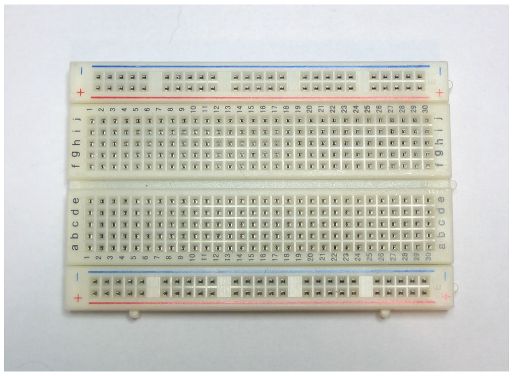


Figure 1: A breadboard oriented like those oriented in images that we can process

- The breadboard is often square with the image.
- The breadboard, when not covered with components, has a black checkered pattern of holes in it.
- The breadboard color is relatively constant over the entire breadboard, except for the holes.
- Breadboards follow a specific connectivity pattern. In Figure 1, each column (not including the ttop two and bottom two rows) can be considered a single connection. The middle gap in the square holes is a break in the connection for each column, so there are five total holes connected to one connection in each of the columns. The top two and bottom rwo rows are connected horizontally into a single connection. Typically, one denotes a voltage of $+5$ volts and one denotes `GND`, or 0 volts.

2.2. Schematic-drawing Software

The Eagle program, developed by CADSoft, is a free schematic and PCB-drawing software that translates schematics into a PCB-ready format. The Eagle file format is a type of XML, which makes it easy to programmatically update from an image. Eagle PCB and schematic diagrams are based off of the same language, so drawing a schematic in Eagle is sufficient to computing the full PCB.

3. User Interface of our Tool

TODO. Explain what the tool looks like from the point of view of the user. There's a GUI. They run it on their image, see the image, pick the primary colors. It thinks for a while and then shows them its segmentation. The user picks the components they want to include in the final layout and it lays them out for them.

4. The Approach

The problem of translating information from a picture of a circuit to a PCB-friendly file format has two main steps. First, the components must be identified in the image, and second, the components must be placed in a virtual grid representation independent of their relation in the image.

For our implementation we chose to use python and it's Tkinter, PIL, SciPy, and Numpy packages for ease of implementation, user interface, and speed.

4.1. Segmentation into Circuit Components

4.1.1 Color Following

As a first step to tackling the component segmentation problem, we took advantage of the natural segmentation in a breadboard picture according to color. Components on a breadboard are different colors that are localized in one location, so to take advantage of this, we developed an algorithm that "grows" a component based on a given color pixel. A seed color is given to the algorithm and added to an active set of pixels. For each neighbor of each pixel in the active set, the decision is made whether that pixel is in the component if the pixel does not change the running RGB color-average of all the pixels current in the component. If a pixel is added to a component, it is then also added to the active set and the algorithm continues.

The main problem with this algorithm is that it is highly dependent on the seed pixel to the image. If a pixel is chosen that is too dark for example, the running average that the algorithm will calculate will be off from the actual running average of the color of the component, so the algorithm will return incorrect results.

4.1.2 Segmentation by Nearest Color

catherio will write this

4.2. Occlusion Handling

A problem with breadboard circuits is often that components (most often wires) are occluded from each other. This is a critical issue that in any finalized version of a product like this needs to be addressed. Because we ran into other issues with our approach, we were not able to effectively solve the occlusion problem.

4.3. Virtual Grid Representation

The next component of the project involved translating the component data into a schematic. Knowing all the pixel locations that make up a component, we calculate the end pixel positions of each component rounded to the nearest 10. Because components are aligned to the hole-grid that is present on the breadboard, the calculated endpoints of the components can be used to place components on a grid in the schematic. The rounded values for the x and y coordinates are used to place the components into their XML representations into the schematic.

After all components are identified and represented in their XML formats, a new file is constructed with the proper sschematic-XML information.

5. Results

5.1. The Full Spectrum

5.2. Comparing Color-following and Segmentation by Nearest Color

The color-following and nearest-color algorithms solve somewhat different problems. The color-following algorithm assumes a seed pixel (whether given programmatically or by a user) and constructs a component from that seed pixel, while the nearest-color algorithm separates the image into sections based on colors found in the entire image. In both bases, if the same component is found, it should be found correctly. Given that the exact parameters for both algorithms were not perfectly tuned, these results are a comparison of the current implementations of the algorithms as we have them.

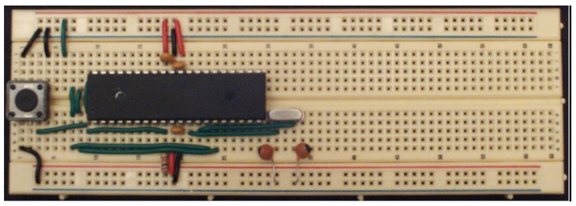
Figure 2a has the original image that we are analyzing. Figure 2b shows one wire detected by the color-following algorithm. Note how, given a particularly good seed color, it was able to only detect one of the two wires that are nearest each other. In Figure 2c, note the same wire is detected as a single wire with the wire immediately below it.

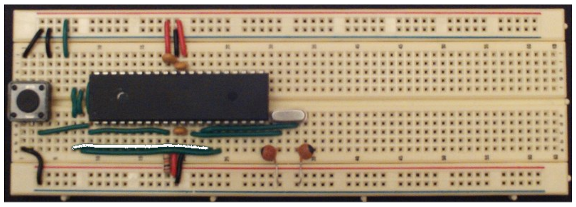
5.3. Limitations

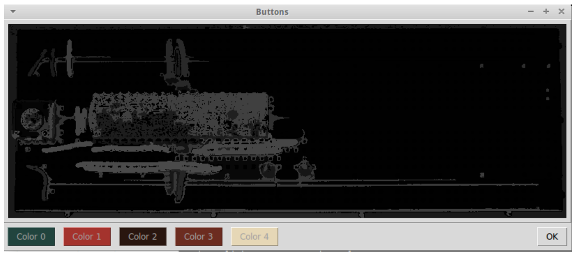
One limitation of our product as we've written it is that it can only handle breadboard circuits that are oriented horizontally as shown above in Figure 1. This is because connectivity properties of the breadboard are geometry-dependent, and our program automatically assumes a particular orientation for the breadboard to infer connectivity.

6. Future Work

TODO: automatically identify components


(a) The original image


(b) The white wire detected by the color-following algorithm



(c) Note the same wire incorrectly detected by the closet-color algorithm

7. Conclusion

TODO