Beat Tracking a Breadboard — Hole and Wire Identification on a Breadboard

Reem Alkhamis & Nicolás Naar[1]

5/13/2020

***Abstract*—**This report details the implementation of a beat tracking algorithm to predict the hole locations of a solderless breadboard, and the use of these predictions to identify straight red wires. The implemented beat tracking algorithm identified the majority of the breadboard holes proving that this is a viable method of hole identification, but the component identification using the predicted holes was not reliable.

I. Introduction

Solderless breadboards are frequently used for rapid prototyping of low frequency analog circuits. Once a circuit design is completed on a breadboard, one often wants to document the circuit for future reference. At this point it would be useful to be able to take an image of the circuit and have an image processing algorithm create a netlist of components and nodes instead of having to generate it by hand.

The first step to creating this application is having an image processing algorithm capable of locating the breadboard itself and the hole locations on the board.

As with any complex system, the scope of the algorithm must be clarified. The algorithm described in this paper was written to locate the holes for images of a standard 830 tie point breadboard taken from top view on a monochromatic dark background with minimal texture. The breadboard images were all resized to 23% of their original size (750x1000px) to reduce the run time. Additionally, this algorithm assumes that the only components on the board are straight red wires.

Section II details the system design for this algorithm and Section III lists the results of the implementation and discusses the interpretation of these results within the proposed scope.

II. System Design

The overall system algorithm takes in an input image of a breadboard populated with red wires and a black background. The system outputs the grayscale image of the board overlayed with dots that show the predicted hole locations, as well as boxes surrounding the predicted components.

This algorithm can be broken down into three main building blocks: image correction, hole identification, and component identification as seen in Fig. 1. The image correction block receives the original breadboard image and outputs a cropped image of the breadboard without the background while also removing optical distortions. This output is then passed into the hole identification block which identifies the pixel locations of the onset of breadboard holes. Hole onsets are defined as the location where white pixels shift to black pixels (and is further discussed in section 2B1). These locations are then passed to the component identification block which analyzes each hole to check if a component exists nearby. The following subsections describe each of the three building blocks in detail.

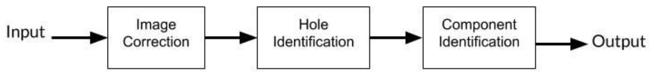

Figure 1. System Block Diagram

a) Image Correction Algorithm

The image correction algorithm corrects for optical distortion of the input image while also cropping the image to remove the dark background. The image correction algorithm can be broken down into three subsystems: breadboard outline detection, corner detection, and keystone correction and cropping as shown in Fig. 2.

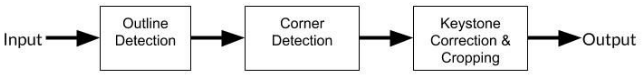

Figure 2. Block diagram of the image correction subsystem.

The outline detection subsystem applies Canny edge detection on the original image. The result is then used to create a list of continuous paths by iteratively checking if surrounding pixels were also detected as an edge. From the list of continuous paths, the largest path is assumed to be the outline of the breadboard because no individual component should have a path larger than the breadboard. The outline of the breadboard is the input to the corner detection subsystem. The corner detection subsystem uses the Harris corner detection algorithm to find the corners of the breadboard outline. These corner locations are then passed to the keystone correction and cropping subsystem which maps the grayscale image outlined by the four corners to a rectangle with pixel dimensions proportional to the known breadboard dimensions (265x815px). This process both crops the image and corrects for optical distortion. The described output of the corner detection block is the input to the hole identification block in Fig. 1.

b) Hole Identification Algorithm

The hole identification algorithm receives an input image from the image correction block and uses beat tracking to generate a list of predicted hole onset locations. The use of beat tracking on the holes of a breadboard is a focus of the overall algorithm described in this paper. For this reason, this section will first discuss the implementation of the beat

[1]Students of Engineering at Harvey Mudd College.

tracking algorithm for a one dimensional array and then describe how it is used to determine the hole locations.

1) *Beat Tracking Algorithm Using Energy Novelty*

The beat tracking algorithm is composed of two components, energy based onset detection and dynamic programming.

Onset detection looks for positive increases in energy of the original signal x(n). The energy of signal x(n) is calculated with Eq. 1, where w(m) is a window function, M is half the window length, and δ is the hop size in samples.

$$E_w^x(n) = \sum_{m=-M}^{M} |x(n*\delta+m)*w(m)|^2 \tag{1}$$

From the result of Eq. 1, the change of energy, ΔE, is calculated and rectified to only show positive increases in energy using Eq. 2.

$$\Delta E(n) = |E_w^x(n+1) - E_w^x(n)|_{\geq 0} \tag{2}$$

Using the rectified $\Delta E(n)$, the beat sequence score at each beat is calculated using Eq. 3. Eq. 3A generates a beat sequence score that rewards strong onsets but penalizes based on a chosen hyperparameter λ and the penalty function. The penalty function shown in Eq. 3B increases the penalty based on the difference between δ and $\hat{\delta}$, where $\hat{\delta}$ is the estimated beat frame distance. It is important to note that for an image of an object with constant physical dimensions, $\hat{\delta}$ can be calculated with simple geometry.

$$S(B) = \sum_{l=1}^{L} \Delta(b_l) + \lambda \sum_{l=2}^{L} P_{\hat{\delta}}(b_l - b_{l-1}) \tag{3A}$$

$$P_{\hat{\delta}}(\delta) = -log_2(\delta/\hat{\delta})^2 \tag{3B}$$

The beat sequence score is then converted into a dynamic programming equation as shown in Eq. 4, where D(n) is the maximum cumulative score of a sequence of beats ending on frame n. The previous beat of the optimal path that terminates on frame end is stored in an array P(n).

$$D(n) = \begin{cases} \Delta(n) \\ D(0) + \Delta(n) + \lambda P_{\hat{\delta}}(n) \\ D(1) + \Delta(n) + \lambda P_{\hat{\delta}}(n-1) \\ D(2) + \Delta(n) + \lambda P_{\hat{\delta}}(n-2) \\ \dots \\ D(n-1) + \Delta(n) + \lambda P_{\hat{\delta}}(1) \end{cases} \tag{4}$$

From the D(n) array, the maximum value is located and should correspond to the last beat. From this beat, the path with the highest score is backtraced to find the optimal path. This path is a list of frame indices pertaining to the beat sequence.

2) *Hole Identification Using Beat Tracking*

To identify holes, the beat tracking algorithm described above is used to identify columns and rows with beats. The intersections of the rows and columns with beats are computed and the output pixel pairs are overlaid on the grayscale breadboard image from the image correction block. The block diagram for the hole identification algorithm is shown in Fig. 3.

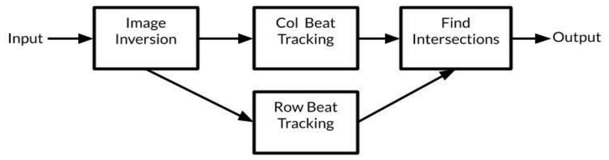

Figure 3. Hole identification block diagram

The image inversion subsystem takes an input grayscale image of a breadboard, meaning that each pixel has a value between 0 and 255 (inclusive), and computes a pixel-wise inversion using Eq. 5.

$$P_{inv}(x,y) = 255 - P_{gray}(x,y) \tag{5}$$

The output of the image inversion is then passed to the row and column beat tracking algorithms. For column beat tracking, the inverted image is summed across all rows and normalized to create a one dimensional array. This array is then passed to the beat tracking energy based novelty algorithm that was described in the previous section. For this specific application the window function was a Hanning window of length eight and the hop size was set to four.

For the row beat tracking algorithm, the board is split into four sections, as seen in Fig. 4, in attempts to avoid detecting power line markings and the center divide as rows with high energy. Each of these sections is summed across its columns and the four resultant one dimensional arrays are passed through the beat tracking using the same energy based novelty algorithm. The four outputs of the row beat tracking are then added sequentially to generate a single output.

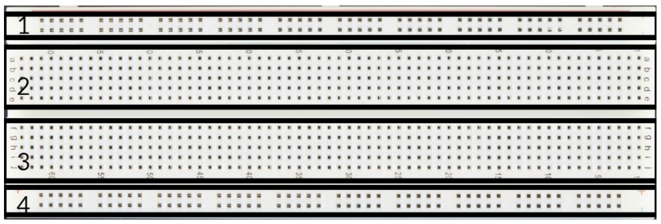

Figure 4. Board split into four sections for row beat tracking

The outputs of the column and row beat tracking are filtered to remove duplicate beats. Duplicate beats are beats detected on two subsequent frames. This is a result of the windowing function partially catching a hole on frame *i* and then catching a large amount of the hole on frame *i+1* so both frames appear to have large increases in energy, but it should only be considered one beat.

The find intersections subsystem receives the row and column beats and generates every combination of row and column which represents the predicted hole onset locations.

From Fig. 4, it can be seen that sections one and four have less holes per row than sections two and three. To account for this, a hard mask is applied to remove these extraneous holes.

c) Component Identification

The component identification algorithm uses the fact that straight wires must occur near the predicted hole locations. Furthermore, on an inverted grayscale image, red wires appear to have a lot of concentrated energy. Due to this, component identification can be simplified to checking near the predicted holes for high energy content. To search for high energy regions, a gaussian rectangular window was element-wise multiplied by the pixels surrounding each hole. This window captured the energy to the bottom right of a predicted hole onset location, which is where any components must be located.

From the data of energy surrounding each hole, energies that were three standard deviations higher than the mean were predicted as component locations. This assumption is derived from the fact that if there were no components, the hole energy data will have a normal distribution. Statistically, this means that 99.85% of holes without components will not have energy larger than three standard deviations from the mean. It is important to note that this algorithm assumes that the breadboard is minimally populated, meaning that the majority of the holes do not have components. Due to this caveat, the algorithm will decrease in accuracy as more components are added.

III. Results and Discussion

Overall, the algorithm identified most of the holes and red wires on breadboard images as shown in Fig. 5. The results and interpretations of the data from each block of the algorithm are discussed in the following subsections.

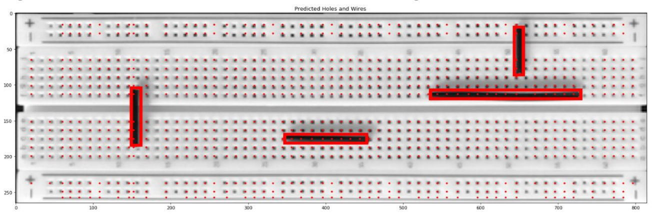

Figure 5. Breadboard with predicted holes and wires

a) *Image Correction Algorithm*

The algorithm was tested on 40 images of empty breadboards and 43 images of breadboards with wires. The image correction block successfully corrected 77.5% of the images of empty breadboards and 83.7% images of breadboards with wires. This is due to slightly different lighting conditions between the two sets of images. An example output of image correction for empty breadboards is shown in Fig. 6.

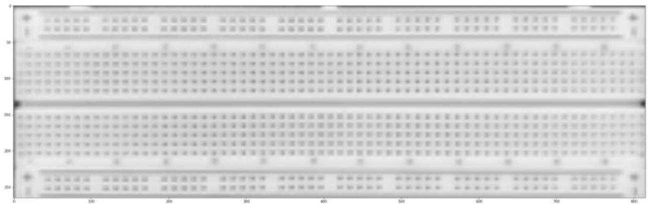

Figure 6. Breadboard image after Image Correction

There are two main reasons that breadboards failed to be correctly identified. The first reason is tilted images. Standard breadboards are inherently rectangular, but they have three external plastic connection points that are often detected as corners. To filter these out, a basic filtering system was applied that looked for the four outer edges, but this filtering relied on the breadboard boundaries being relatively parallel to the image boundaries or else a corner could be missed. The second reason is that under dim light, the center divide is detected as a breadboard edge. The images therefore had to be taken under an LED desk light, which solved the majority of the incorrectly identified images.

b) *Hole Identification*

The beat tracking approach to identify holes was tested on 31 images of empty breadboards that passed through the image correction block. An example of a perfect hole identification output overlayed on the breadboard image is shown in Fig. 7.

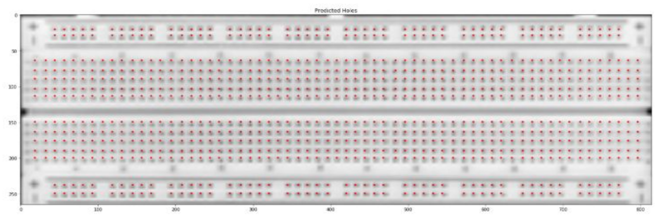

Figure 7. Breadboard with predicted holes from beat tracking algorithm.

Because there is no set formula to derive the optimal hyperparameter value from Eq. 3, it needs to be determined experimentally. It is important to understand that for the beat tracking algorithm, the λ value generally needs to be within the correct order of magnitude to affect the system performance. Various λ were tested for the rows and column beat tracking. The results of the tests are summarized in Tables 1 and 2, respectively.

Table 1: Correctly identified rows for various row λ values. Note that rows added statistic is the number of rows added over the number of rows that should be present.

Row Lambda	Rows Correctly Identified	Rows Added
0.1	85.0%	4.3%
0.01	95.9%	4.2%
0.001	95.4%	4.4%
0.0001	95.8%	4.4%
0	95.8%	4.4%

For the row λ values, the results are relatively stable once λ is less than 0.1. At a λ of zero, there is no penalty for off beat onsets. Due to this, it is likely that the power rail markings (in sections 1 & 4 of Fig. 4) get counted as beats. This is a challenging problem to filter out with beat tracking because the distance from the rail to the holes is similar to that between holes, hence regardless of an increased λ value, the power rail still often got caught as a beat. Contrarily, if the λ of the beat tracking algorithm is too large, then onsets that do not specifically fall on beats will get disregarded. This results in many of the rows of sections 2 and 3 (from Fig. 4) to not be counted as beats. For these reasons, the row λ should be between 0 and 0.1 so it was kept at 0.001.

Table 2: Correctly identified columns for various column λ values. Note that columns added statistic is the number of columns added over the number of rows that should be present.

Column Lambda	Columns Correctly Identified	Columns Added
100	33.3%	4.28%
10	95.9%	0.05%
1	98.7%	0.05%
0.1	100%	0.05%
0	100%	0.05%

From the table above, it would appear that the lower the column λ value, the better the performance. While this is true for the metric that was used for these experiments, when the λ value was set to zero, it resulted in many of the holes to have onsets in the center of the holes instead of their actual onset location, which should be to the left of the holes. For this reason, it would be best to decrease λ until all columns are correctly identified. However, if λ is decreased much past this point, it negates the impact of the beat tracking algorithm which makes it more likely for onsets to occur in the middle of holes.

c) Wire Identification

Wire identification was tested on 36 images of breadboards consisting of 14 different configurations of wires. Each configuration included zero to seven straight red wires of various lengths randomly located on the breadboard. In total, 128 wires were tested. A correctly identified wire is fully encompassed within a rectangular box as shown in Fig. 8.

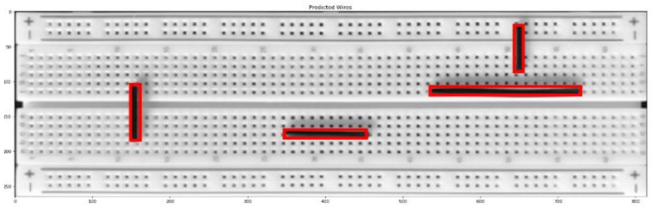

Figure 8. Breadboard with identified wires

All the tested images were first passed though image correction and hole identification blocks. The results of wire identification are summarized in Table 3.

Table 3: Results of wire identification

	Number of wires
Correctly identified	38.3%
Partially identified	38.3%
Completely missed	23.4%

It is important to note that the algorithm depends on a correct prediction of hole locations. If the hole identification missed a row or a column where a component is located, then the wire identification would not be searching for that component.

Additionally, it is interesting to note that as the number of wires increases, it becomes more difficult to tell which energies are from empty holes versus holes with components. This is shown in Fig. 9, which compares a histogram of a single wire, with one clear outlier in the histogram versus an image of seven wires where the boundary between empty and component holes is less clear. Furthermore, this boundary is based on the standard deviation and mean which evidently begin to drastically change when many wires are added.

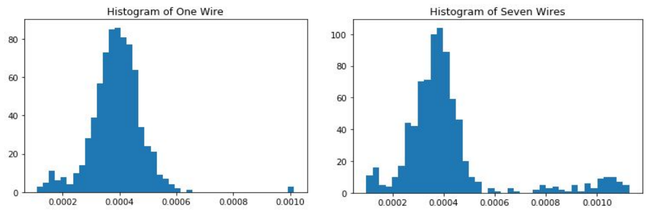

Figure 9: Histogram of energy surrounding hole for one wire (left) versus seven wires (right)

IV. Conclusion

This report described a beat tracking approach to identify breadboard holes and used these hole locations to find red wires on the board. From images that were successfully corrected and cropped, the hole identification algorithm was able to correctly predict the location of 95.4% of rows, 100% of columns on an empty breadboard. The wire identification algorithm correctly identified 38.2% of components on populated breadboards.

The results of these experiments demonstrate that using a beat tracking algorithm to identify holes is a viable approach to hole detection. However, using energy detection based on these hole locations is not a robust method to correctly identify components.

V. Acknowledgements

We wish to thank professor Timothy Tsai for teaching us introductory techniques in signal processing and for his feedback throughout various points of this project.

VI. References

[1] Sahir, S. (2019, January 25). Canny Edge Detection Step by Step in Python — Computer Vision. Retrieved from https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

[2] Tsai; Timothy. "Beat Tracking Lecture." Engineering 190AV. Harvey Mudd College. Lecture