CIRCUIT SNAP

Nikhil Garg & Ankit Tandon

Electrical & Computer Engineering, University of Texas at Austin

ABSTRACT

Students learning circuits for the first time often have to solve many voltage-resistor circuits in order to learn Kirchoff's laws, KCL and KVL. However, most introduction circuits textbooks do not include answers to all their practice problems and examples. Without feedback, students often learn less than they could. We introduce an automated solution to this problem. Students will be able to take a picture of a circuit from a textbook, and our system will identify all the elements and solve the circuit. The system is robust and works with circuits from the most popular introdoction to circuits textbooks. We identify the most effective features and machine learning techniques to work under the noisy and small image constraints, along with evaluating those techniques that did not work. These include principle component analysis, k-nearest neighbors, and neural nets.

Note: Though we do not cite every function/command in this paper, Python, OpenCV, and Scikit-Learn were used to implement many of the modules of this project.

***Index Terms*—** PCA, SURF/SIFT, KNN, SVM, Neural Nets, Machine Learning, Object recognition

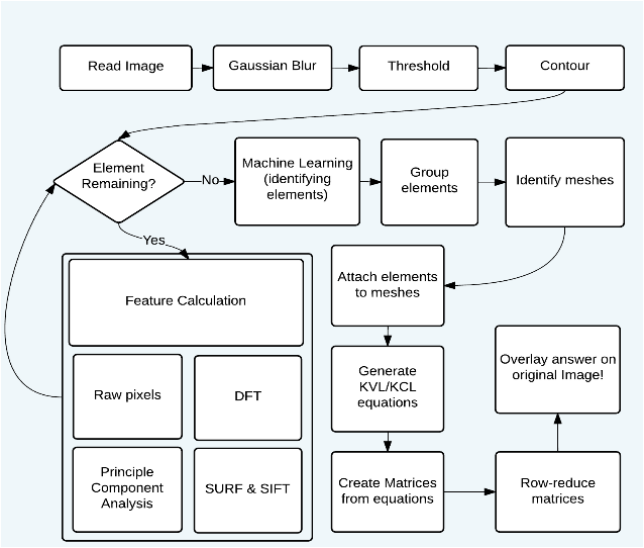
1. INTRODUCTION

In this report, we describe the implementation of an automatic circuit solver from a photo. The problem entails preprocessing, object recognition and association, and the final circuit solving. Figure 1 contains the system flowchart.

2. PREPROCESSING

Preprocessing an image before trying to identify contours is an essential part to making the system robust. Photos taken of circuits will often be under different lighting conditions and may be noisy. Preprocessing is used to normalize the images. Two key functions were used on the images. First, a gaussian blurring filter is used to decorrelate noise. Tweaking the window size and the variance of the noise were important, and a median noise filter is also considered. Ultimately, a 5x5 window with a gaussian blurring filter with unit noise is used to deblur image. After blurring the image to reduce noise, an adaptive threshold is used to convert the image to a binary black/white image. A binary image is especially useful for object recognition because it reduces the dimensions of the data to send to the machine learning component. It also makes the system more robust because some textbooks color their circuits while other textbooks do not. We first attempted a single threshold on the image, using the image histogram. However, this technique did not prove effective because of shadows introduced in the photo-taking processing. Figure 2 shows an example of a photo in which a static threshold failed. Next, adaptive thresholding was attempted. Two adaptive thresholding methods were tried. With an 11x11 block size, we first thresholded by the mean of each block. However, this method degraded the quality of numbers and other objects too much. Next, we used what is called a Gaussian threshold. This method also uses a sliding window. However, instead of thresholding by the mean of the block. the method thresholds by a weighted sum of the block, with pixels close to the center pixel given additional weight. This method reduces the degradation of small details and reduces dependence on the block size. The gaussian adaptive threshold is used in our system. Through these two processes, we are able to send a relatively clean and stanardized image to

Fig. 1. Flowchart for CircuitSnap



Thank you to Dr. Bovik for providing this project opportunity, motivating us, and teaching us image processing. Also, thanks to the ICIP journal for providing this lovely LaTeXtemplate. This work is completed in Fall 2014 for EE371R at UT Austin.

Fig. 2. Photo with significant shadowing

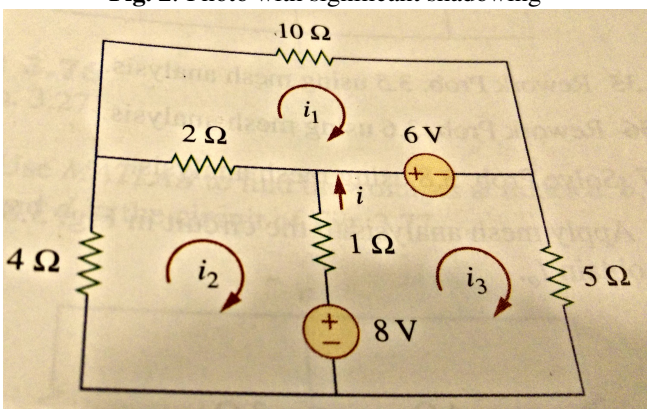
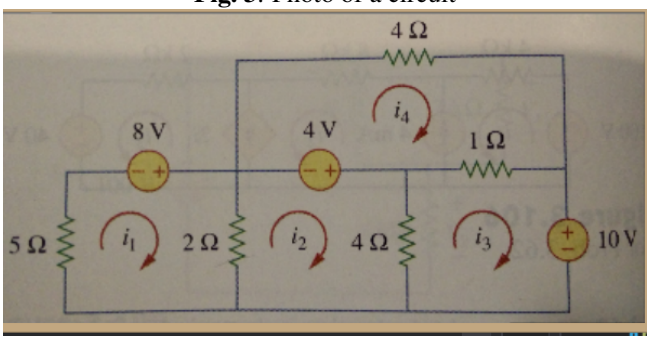


Fig. 3. Photo of a circuit



our detection algorithms. Figure 2 shows a sample original image, and Figure 2 shows the circuit after blurring and adaptive thresholding. This technique is not fool-proof, however. Figure 2 shows a image in which adaptive thresholding amplified noise due to the thin nature of paper in a textbook. These errors are handled downstream in the process.

3. CONTOURING TO ISOLATE OBJECTS

After obtaining a clean image from preprocessing, the next step is to identify every number, symbol, and element image inside the circuit. In order to be robust, feature detection must be scale and size invariant, and must be able to handle small deviations. Thus, template matching is off the table, and other techniques must be used. The first step to recognizing objects in a robust way is to isolate each circuit element, number, and symbol from each other. To do so, we use a contouring mechanism as described in [1]. This technique follows 'lines' or connected components in the image, and then clusters based on those lines. Rectangles are identified surrounding each group of connected pixels, and each of these rectangles represent a potential circuit element or value. Figure 3 shows the circuit from Figure 2 with potential objects recognized.

Note that resistors are not identified as potential elements by the contouring because the method identifies each connected line component as a potential element, and resistors are continuous with wires without any corner or without multiple lines converging into one. Thus, are were identified through traditional template matching with a twist. This method is discussed below.

Fig. 4. Circuit from Figure 2 after preprocessing

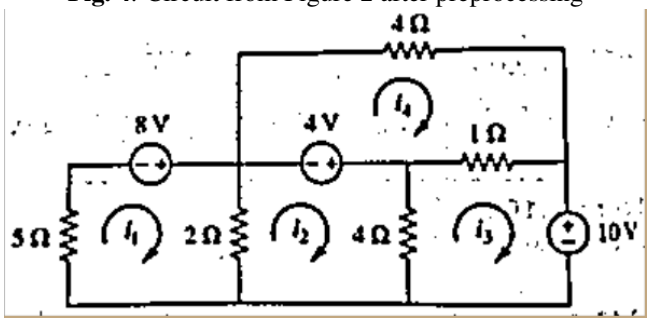
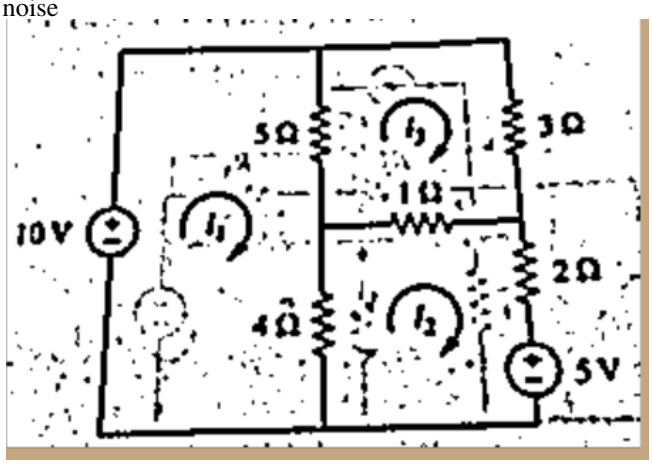


Fig. 5. Circuit for which adaptive thresholding introduces noise



Next, these potential objects are identified.

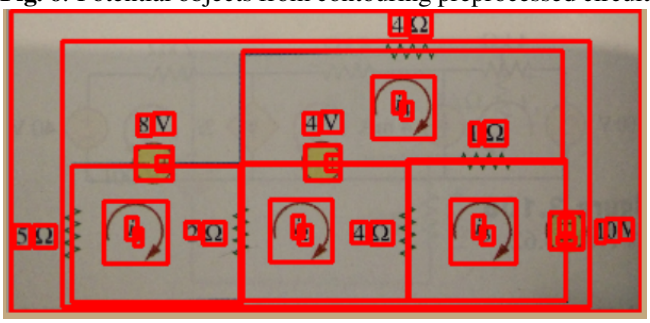
4. OBJECT RECOGNITION

The next, most important component of the system is to evaluate and use object features to recognize circuit elements, values, and symbols in the circuit. This process is split into two components – feature calculation and machine learning – with resistor recognition following a different path than the rest of the components.

4.1. Object Features

For each potential object identified by the line tracing described above, features are calculated to attempt to uniquely identify the element. First, the region of interest is resized to a 20x20 block. This resizing makes the system robust against photos of different sizes. If the region needs to be reduced in size, the region is subsampled. If it needs to be increased

Fig. 6. Potential objects from contouring preprocessed circuit



in size, the region is oversampled and then the missing pixel values are filled in with cubic interpolation.

Next, the system extracts features from each region. Many potential features were tried, and the best-performing features were chosen for the system. First, we attemped to calculate SIFT and SURF features from the region of interest. However, the small block size (20x20) along with poor image quality resulted in poor performance in object recognition. Three features were finally identified to be sent to the machine learning component. First, the system sends the raw pixels in the entire 20x20 block. This 'feature' works because the system eventually uses K-Nearest Neighbors as its separating algorithm. Next, the system sends the entire 20x20 FFT of the block. Sending the FFT creates a more robust system because though different fonts or textbook pictures may differ slightly, most symbols are drawn approximately the same and contain the same frequencies in each direction. With adequate training, these features performed decently, with about 60% element identification. The final feature, which improves element detection significantly, is the top components identified by Principle Component Analysis. Principle Component Analysis is "Linear dimensionality reduction using approximated Singular Value Decomposition of the data and keeping only the most significant singular vectors to project the data to a lower dimensional space"[2]. In other words, we identify the most significant eigenvectors of the region image matrix, and send those values to the machine learning component. PCA is especially resistant to noise and small deviations, and it effectively separates elements.

4.2. Resistors

As mentioned above, resistors are not identified as potential objects by contouring. Thus, template matching is used to identify resistors. However, typical template matching would not be scale and rotation invariant, and it would not work with new textbooks or resistor drawings. Thus, we used an array of templates and attempted to match the templates at various scales and rotations, with differing thresholds to handle noise. However, this method introduces many false positives, on the order of 7 false positives for resistors for every correctly identified resistor. Thus, we used fuzzy weighing to identify the possible resistor that would best explain the rest of the elements. In particular, we weighed how close potential resistors were to values for resistors, along with how strong of a match the resistor was with a given template. The system also takes into account wires/lines in identifying most probable resistor. Figure 4.2 contains a circuit in which the resistors are correctly identified. However, this method is not fool-proof, and in some circuits the skew is too high to identify resistors using our method. Figure 4.2 contains an example of such a circuit.

Fig. 7. Circuit with resistors correctly identified

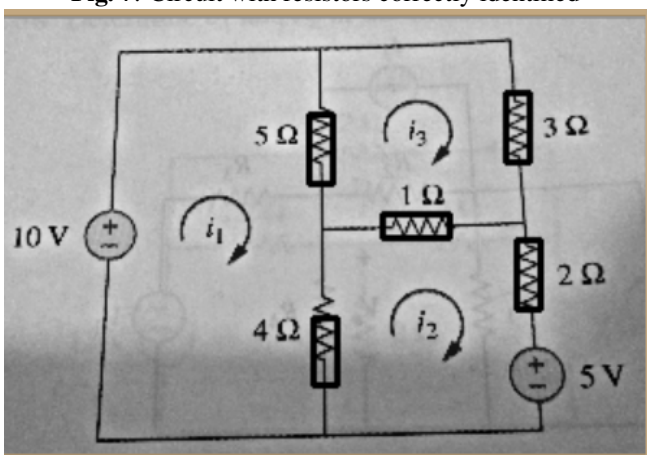
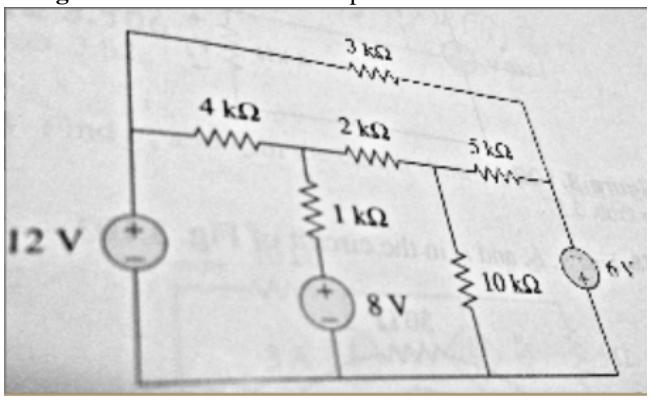


Fig. 8. Circuit in which skew pevents resistor detection



4.3. Machine Learning

The features calculated for each potential object identified through contouring were sent to a machine learning module. Three types of modules were considered. First, Support Vector Machines (SVMs) were used. SVMs attempt to find a line in the high dimensional space that separates different types of objects. However, our data was too noisy and the features too high dimensional (the FFT and the pixels themselves contributed 800 dimensions, before PCA), and SVMs produced

Fig. 9. Identified values, symbols, and elements for circuit in Figure 4.2

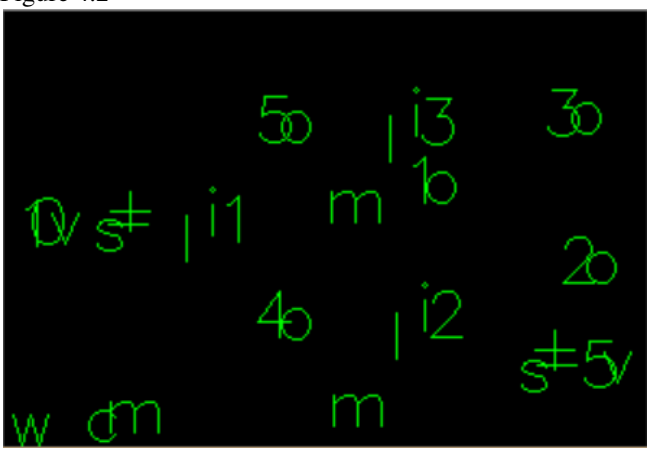


Fig. 10. Red values indicate voltage drops and Blue values represent mesh currents

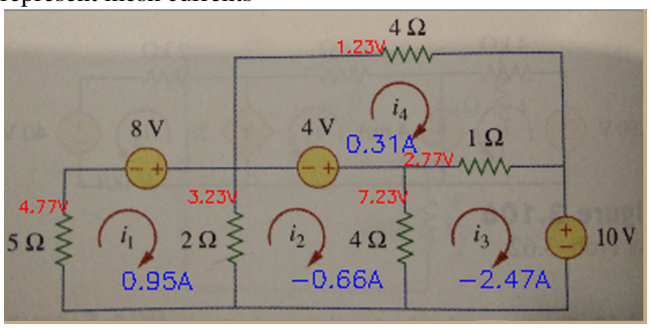
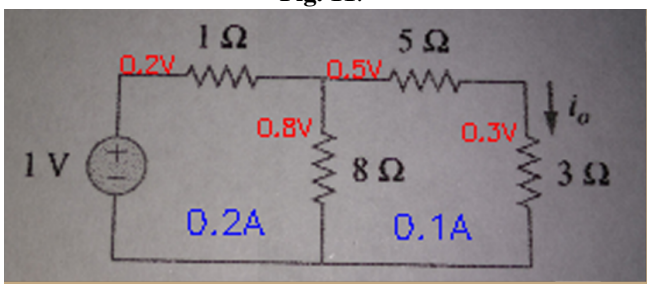


Fig. 11.



results on the order of 15% recognition at best. Next, neural nets were attempted. However, we did not have enough training data for such a high dimensional deep neural net to be effective. Our system requires training on the order of 20 images, and neural nets typically work effectively only on the order of hundreds or thousands of training samples. Ultimately, K-Nearest Neighbors was used for the machine learning component. This method is resistant to noisy samples and errors. It also requires few training samples. For some objects, especially some of the more identifiable numbers, a single training sample works in identifying the number across different textbooks, fonts, and scales. This method works very well in identifying objects. Figure 4.3 shows elements recognized for a sample circuit. Note that though there are several errors, most of the elements have been recognized correctly.

5. OBJECT GROUPING

After all the individual objects have been recognized, they must be combined in a way as to accurately reflect the target circuit. This module must handle minor errors in object recognition and combine associated elements, such as multi-digit numbers and voltage sources, together. To do so, the system employs logic to find best explanation for the recognized objects in such a way that minimizes error distances and punishes repetitions. A similar method as the resistor pruning method described above is ultimately used.

6. CIRCUIT ANALYSIS AND RESULTS

Once the circuit elements have been identified there is still significant work to be done in order to make sense of the circuit and solve for desired values. Initially our plan was to leverage a software package called Ahkab, which solves circuits. The reason we didn't use Ahkab is two-fold: first, Ahkab does not perform systematic KVL analysis, instead it analyses circuits by performing a simulation across all elements and stepping the different sources at a specified granularity. Second, Ahkab is primarily built for AC and much more complicated circuits for which analysis by hand is too complicated. Circuit Snap is targeted towards analysis of simple circuits. We decided to build out our own circuit analysis module that we could adapt later on as we choose to handle more complicated circuits. Our approach to solving circuits is to calculate the current around every mesh by using mesh analysis and then solve for the voltage drop across all elements. We do this by first discarding extraneous meshes i.e. ones that have no elements or ones that are supermeshes. Once we have finalized our list of meshes we attach elements to them in the appropriate branch (top, bottom, left or right). This involved analyzing the spatial relationships between elements and all the meshes. We had to allow for some tolerance regarding how close elements can be to be considered part of a mesh. Now that the elements have been attached to meshes we make a pass through every mesh and loop through its elements in a clockwise fashion to add up the resistance values. If an element exists on multiple branches we must subtract the current running against the mesh being inspected. Doing this analysis leaves us with a matrix of equations and we can now solve for the unknown mesh currents by performing Gaussian elimination. We used the python package numpy to perform this row reduction. Figure 10 and Figure 11 show the resulting image. We display the voltage values to the top left of the resistors and the mesh currents go toward the bottom of the meshes. From the figures one can see that our system is ro-

bust enough and works across various types of circuits (from screenshots of pdfs to noisy images taken of textbooks).

7. WHAT WE LEARNED

We learned that achieving robustness is hard but essential and certainly possible. At every step along the development process we had to carefully evaluate our design decisions to make sure that we were building a tolerant, robust system that would work across different types of circuits and images. We also learned that features that might work in theory may not work in practice. Certian features like SIFT work well in most images but since our circuit images were small sized they were performant enough to deliver solid results. The same thing goes for Machine Leanring techniques. While SVMs are a great idea, our data was too high dimensional for SVMs to work well. Similarly Neural Nets would have seemed like a great idea to use however we lacked sufficient training data to leverage any real advantage from neural nets. Finally, we learned about how essential preprocessing images are. Without our preprocessing steps, image analysis would have been nearly impossible. Preprocessing also makes our system more robust as it normalizes all the images we process and removes noise.

8. WHO DID WHAT

Nikhil handled the element detection, feature learning and evaluation, and image preprocessing. Ankit handled the circuit analysis, image overlay, and KVL/KCL loop calculation. Together we brainstormed various approaches to tackle the problem at a high level; implementation was split up amongst members.

9. REFERENCES

[1] R. S. Ramakrishna, S. K. Mullick, and R. K. S. Rathore, "A new iterative algorithm for image restoration," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 47–55, 1985.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.