Augmented Reality Circuit Learning

Hao Wang
Electrical and Computer Engineering
University of California, Davis
CA, USA
haowa@ucdavis.edu

Sen-Ching S. Cheung
Electrical and Computer Engineering
University of Kentucky
KY, USA
sccheung@ieee.org

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401464

***Abstract*—Building electronic circuits is one of the most common hands-on activities in learning STEM subjects. Beginning students often find it difficult to translate a circuit schematic into the construction of the physical circuit on a breadboard. In this paper, we describe an Augmented Reality circuit learning software that can provide students with step-by-step instructions by placing virtual circuit components on a physical breadboard. We propose a novel image processing pipeline that can robustly identify the planar structure of a breadboard, even in the presence of occluding circuit components on the breadboard. Using a commercially available library of 3D circuit component models, the estimated 3D structure of the breadboard allows us to render arbitrary circuit components on it in real time. Experimental results demonstrate that our algorithms are accurate and can produce realistic-looking virtual circuits.**

Keywords—vision-based circuit inspection, augmented reality, circuit learning

I. Introduction

Many studies have demonstrated that hands-on activities help students to achieve better academic performances in STEM (Science, Technology, Engineering, and Math) subjects than those who learn the same subjects in primarily teacher-centered lecture-based environments [1]. One of the most common hands-on educational activities is building electronic circuits. Electronic circuits are fundamental to all engineering disciplines and building circuits provides students with hands-on learning experiences of abstract physical concepts like voltage and current. While computer-based circuit simulators are now commonplace, hands-on circuit buildings have been shown to better motivate student learning and encourage more creative thinking [2].

At the university level, engineering students typically use discrete circuit components on a breadboard for the majority of their assignments. The standard breadboard is useful for building a variety of simple circuits but its hidden connections could be challenging for beginners to translate from a circuit schematics to its physical construction [3]. It is often difficult, if not impossible, for the teaching staff to work one-on-one with individual students to help them to debug their circuits. As such, a tool that can assist students in constructing and debugging breadboard-based circuits could alleviate the burden of the teaching staff and help students to build self-efficacy. In addition, it can be a key enabler to add hands-on activities to massive open online courses on basic circuit theory [4].

Code and Demonstration video can be found in this GitHub repository https://github.com/Hao-Wang-Henry/locate-and-track-a-breadboard

In this paper, we present a novel educational tool on a mobile camera platform that helps students to build breadboard circuits using augmented reality (AR). Using an ensemble of computer vision techniques, our proposed system can render in real-time virtual circuit components on a breadboard so that a student can follow the visualization to assemble all the components by inserting them into their proper locations. Our design is inspired by the recent development of reflective artificial intelligence technology (RAIT) that integrates modeling and feedback from physical manipulatives directly into a computer display [5]. The most notable examples of RAIT can be found in a series of popular AR games called OSMO developed by Tangible Play, Incorporated [6].

Our main technical contribution is a robust computer vision system to locate the breadboard and track its geometry in real-time for AR rendering. Past approaches on applying computer visions on circuit inspection such as [7], [8] focus on printed circuit boards (PCB), which are professionally manufactured in a controlled environment. Recently, [9], [10] also present AR-based circuit simulators for educational purpose, but they do not utilize a real breadboard so the learners' transition from virtual to real circuit building would be more difficult. A system similar to ours was recently suggested in [11], which does not demonstrate its feasibility with actual results.

The rest of the paper is organized as follows: Section II outlines the design of our AR circuit tool and discusses some of the challenges. Section III provides the details of our algorithmic design. Section IV contains experimental results to demonstrate the performance of our proposed system. We conclude in Section V with some of the future works.

II. System Overview

The goal of our tool is to provide a step-by-step circuit building instruction through an AR visualization of virtual circuit components on a physical breadboard. In order to deliver this mix reality rendering, the geometry of the physical breadboard with components must be determined from the captured video frames so that virtual components could be overlaid. In this section, we provide an overview of the major processing steps to accomplish this task.

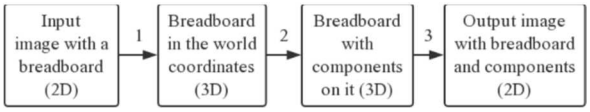

Fig. 1. General processing pipeline

978-1-7281-9201-7/21/$31.00 ©2021 IEEE

The key steps of our algorithm are shown in Fig. 1. In step 1, we determine the location and geometry of the breadboard in the image construct a 3D model. In step 2, we identify the components needed from a library of 3D circuit component models, make the necessary size adjustment, and render them at the specified hole locations on the 3D breadboard. Finally, in step 3, we perform the inverse calculation of step one and convert our 3D model back into a 2D image for display.

In the first step, we need to identify the locations of all the empty holes on the breadboard. A breadboard is a planar surface so the typical approach would be to identify the regular hole features and obtain the plane homographic with respect to the reference world plane [12], [13]. However, such an approach can be affected by circuit components already on the breadboard - a large number of detected image features could land on the components rather than on the breadboard. Even with a robust hole detector, misalignment can occur due to occlusion caused by these components. Instead of focusing on local features, we take a global approach by assuming the general shape of a breadboard and detecting it as a whole. This approach works regardless of occlusions due to components on the breadboard, provided that the majority of the boundary of the breadboard is visible. Once the breadboard is localized, its 3D geometry can be recovered based on the intrinsic parameters of the camera obtained based on its manufacturer and model.

Once the geometry is recovered, the rest of the steps are fairly straightforward. We obtain the 3D models of a large number of circuit components from an online library at the Component Search Engine [14]. We then use AR Studio [15], a popular AR development tool from Facebook, to place the selected components at the desired locations and create the resulting rendering.

III. Methodology

First, we review the projective geometry of planar services and identify the information needed to achieve AR rendering on planar surfaces. Second, we discuss how this information can be obtained from circuit images.

A. Coordinate transformation

In order to transform objects between the 2D image coordinate system (*u, v*) and the 3D world coordinate system (*p, q, r*), it is necessary to incorporate the camera coordinate system (*x, y, z*) that incorporate specific intrinsic camera parameters like camera center, focal length and pixel size. The relationship between these three systems is depicted in Fig. 2. The camera coordinates and world coordinates can be described by the extrinsic relationship as shown in (1), where the $[R\ t]$ is the external calibration matrix denoting the rotation and translation from the arbitrarily chosen world coordinate system.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} p \\ q \\ r \\ 1 \end{bmatrix} = [R_1\ R_2\ R_3\ t] \begin{bmatrix} p \\ q \\ r \\ 1 \end{bmatrix} = [R\ t] \begin{bmatrix} p \\ q \\ r \\ 1 \end{bmatrix} \quad (1)$$

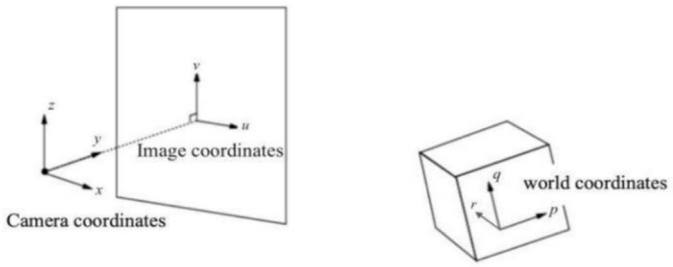

Fig. 2. Image, camera, and world coordinate systems

The coordinate transformation between camera coordinates and image coordinates is described by the intrinsic camera relationship in (2). Here we assume a simple pinhole camera model by using a simplified camera calibration matrix *A* characterized by only the focal length *f* obtained from the manufacturer.

$$k \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & f \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = A \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2)$$

Assuming that the breadboard is on the *PQ* plane so the *r* coordinate of the entire board would be constantly 0. The image of the breadboard is then related to its world coordinates based on a simple homography (3), where $H \propto A \cdot [R_1\ R_2\ t]$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto H \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} \quad (3)$$

As the homography *H* has only eight degrees of freedom, we can set $h_{33} = 1$ and use four corresponding pairs of (*u, v*) and (*p, q*) to calculate the remaining eight values. The (*p, q*) are chosen to be the 4 corners of the breadboard based on the physical measurements of the board. The corresponding (*u, v*) will be detected from the image using the method described in III.B. Once *H* is computed, we can calculate the extrinsic matrix using (4) and (5) and reverse the projection from (1) and (2) to project any AR object at (*p, q, r*) to the image coordinates (*u, v*).

$$[R_1\ R_2\ t] = A^{-1}H \quad (4)$$

$$R_3 = R_1 \times R_2 \quad (5)$$

B. Getting the coordinates

The image processing pipeline of locating the corners is shown in Fig. 3. The pipeline consists of four steps. Step 1 performs image resizing and edge detection. Step 2 uses a blurred edge map to localize the breadboard. Step 3 uses Hough line detection to detect the boundary and the four corners are detected in Step 4.

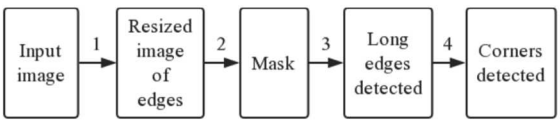

Fig. 3. Image processing pipeline

Step 1: Image resizing and edge detection

As the image resolution differs from camera to camera, we first resize the image to a fixed size in order to ensure the reliability of the algorithm. Due to the presence of the hole patterns which manifest as rapid changes in intensity values, high-frequency features like edges can help locating the breadboard surface. Edge features are also more robust against changes in illumination. In this step, we apply the canny edge detector to obtain the edge map in Fig. 4b from the original image in Fig. 4a.

Step 2: Breadboard localization

As shown in Fig. 4b, the dense hole structure on the breadboard attracts a high density of edge pixels. In order to recover the global shape of the breadboard, we apply a box filter on the edge map in Fig. 4a to obtain a smoothed edge map in Fig. 4c. Using threshold, we obtain the binary mask shown in Fig. 5a in which the central white blob clearly reveals the shape of the breadboard.

Step 3: Hough line detection

Instead of directly locating the four corners, which could be occluded by wires and other circuit components, a more robust approach would be to identify the long boundaries of the breadboard. Hough line transform is an effective way of detecting straight lines in a picture, so we apply the Hough line transform to locate the long edges of the board. The results, however, were not very satisfactory as there were many false positives caused by lines joining corners of the different circuit components on the breadboard.

Taking advantage of the unique hole patterns on the breadboard, we improve the detection of lines parallel to the boundary by first applying a corner detector to uncover the holes' locations (Fig. 5b), followed by another round of edge detection on the corner map to join these closely spaced corners into edge lines (Fig. 5c). The output is a group of lines, the majority of which parallel to the long edges of the breadboard as they are the most dominant line structures in the image. After identifying the largest group of detected parallel lines, we sort them by their distances to the upper left corner. As shown in Fig. 6a, the lines with the smallest and largest distances respectively represent the two long edges of the breadboard.

Step 4: Detection of the corners.

After detecting the long edges of the breadboard, we can identify the four corners of the breadboard by identifying the two short edges and calculating the intersections. In order to obtain a robust estimate of the two short edges, we identify the boundary points of all the parallel lines identified in Step 3 with the breadboard mask obtained in Fig. 5a. A RANSAC procedure is then used to robustly fit a regression line to each side of the mask to obtain the two short edges of the breadboard. The detected corners are obtained and shown in Fig. 6b.

Fig. 4. (a) the original picture; (b) edge image; (c) blurred edge image

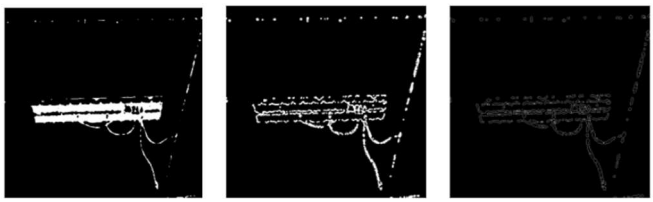

Fig. 5. (a) the binary mask; (b) corners detected on binary mask; (c) edge map of the corner mask

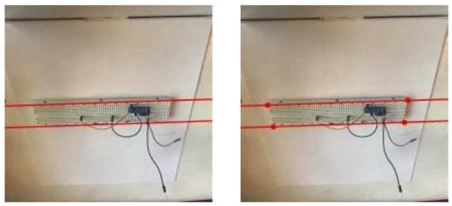

Fig. 6. (a) breadboard with long edges detected; (b) detected corners

IV. Result

We implemented the digital image processing part of the software in Python 3.7.3 and OpenCV 4.1.0. The AR rendering was implemented using Spark AR Studio v73.0.0.10.240. The resolution of all the images is 800×800.

A. Locating and tracking a breadboard

We tested our corner localization algorithm on three different test images - Fig. 7a has an empty breadboard on a clear background; Fig. 7b also uses an empty breadboard but the background is more complex, and Fig. 7c has a breadboard with a number of circuit components and wires. In all three cases, our calculations can identify all the four vertices of the breadboard.

To quantify the measurements, we need the real coordinates of the corners of the breadboard. We use the *labelme* software [16] to manually locate the coordinates of the four corners in each of the three test images and compare them with those estimated by our algorithm. Table I shows that the average coordinate error ranges from 2.13 to 4.88 among the three test images.

The algorithm is time-efficient on a MacBook with Intel 1.8 GHz Intel Core i5 CPU running Mojave OS 10.14.6, it takes about 800ms to process one frame. Although this is not quite real-time, the corner detection algorithm only needs to run occasionally since the geometry of intervening frames can be maintained by a feature tracker that can run in real-time.

(a) (b) (c)

Fig. 7. Results in three different images

TABLE I. COMPARISON (COMP) BETWEEN ESTIMATED (EST) AND GROUNDTRUTH (GT) COORDINATES OF THE FOUR CORNERS AS WELL AS THEIR DIFFERENCES (DIFF)

Comp	Image in (a)			Image in (b)			Image in (c)		
	Est	***GT***	***Diff***	***Est***	***GT***	***Diff***	***Est***	***GT***	***Diff***
x1	111	110	1	107	109	2	107	105	2
y1	255	248	7	330	336	6	406	404	2
x2	84	83	1	101	99	2	127	122	5
y2	378	383	5	422	422	0	493	490	3
x3	737	742	5	698	699	1	629	630	1
y3	412	420	8	402	396	6	475	477	2
x4	719	725	6	668	671	3	641	642	1
y4	286	280	6	310	316	6	389	390	1
error			4.88			3.25			2.13

B. Augmented reality Rendering

Our breadboard, shown in Fig. 8, is a rectangle with a length of 16.3 cm and a width of 3.6 cm. As such, we fix the world coordinates (*p, q, r*) of the four corners as (0, 0, 0), (3.6, 0, 0), (3.6, 16.3, 0) and (0, 16.3, 0). In order to estimate the *H* matrix in (3), we fix the scale by setting $h_{33} = 1$ and estimate the remaining 8 entries by relating the world coordinates of the four corners with their estimated image locations. A rectified view using the estimated *H* of the breadboard from Fig. 7c is shown in Fig. 9.

By using the 3D models of the breadboard and circuit components, we can render a virtual circuit on the physical breadboard at different viewpoints. Fig. 10 and 12 show the same virtual circuit rendered on a physical breadboard captured at different orientations. For comparison, we also constructed the actual circuit on the breadboard and took pictures using the same camera angles as shown in Fig. 11 and 13. The AR rendered results are similar to real circuits.

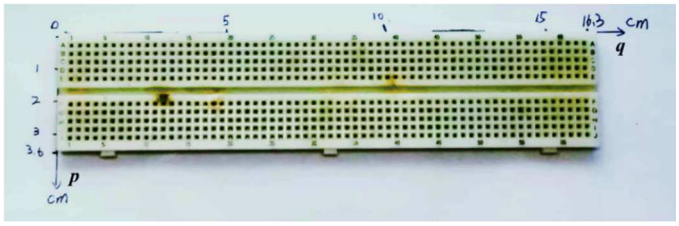

Fig. 8. Our breadboard

Fig. 9. Rectified view of the breadboard from Fig. 7c.

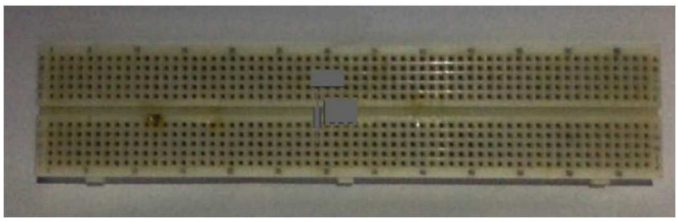

Fig. 10. Augmented reality model A

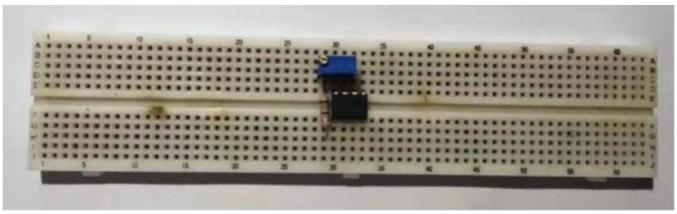

Fig. 11. Real circuit board for comparison

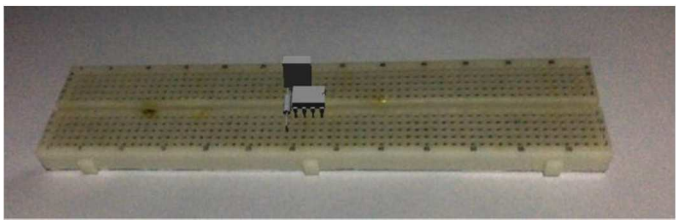

Fig. 12. Augmented reality model B

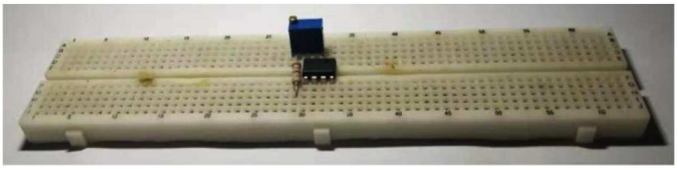

Fig. 13. Real circuit board for comparison

V. RESULT

In this paper, we have proposed an AR circuit learning tool that is capable of identifying a physical breadboard and rendering virtual circuit components on it. Experimental results have shown that our system is able to robustly detect the breadboard and accurately obtain its geometry for AR applications. We are currently integrating this tool with the curriculum of an entry-level circuit learning class and conducting a study to see how effective this tool can help students in overcoming the initial difficulties in learning to build breadboard circuits.

VI. REFERENCES

[1] S. Caglak, “Does Hands-on Science Practices Make an Impact on Achievement in Science? A Meta-Analysis,” JOURNAL OF EDUCATION IN SCIENCE ENVIRONMENT AND HEALTH, vol. 3, no. 1, pp. 69–87, 2017.

[2] A. Ekmekci and O. Gulacar, “A Case Study for Comparing the Effectiveness of a Computer Simulation and a Hands-On Activity on Learning Electric Circuits,” Eurasia Journal of Mathematics, Science & Technology Education, vol. 11, no. 4, 2015.

[3] P. Blikstein and A. Sipitakiat, “QWERTY and the Art of Designing Microcontrollers for Children,” in Proceedings of the 10th International Conference on Interaction Design and Children, Ann Arbor, Michigan, 2011, pp. 234–237.

[4] P. F. Mitros et al., “Teaching electronic circuits online: Lessons from MITx’s 6.002 x on edX,” in 2013 IEEE International Symposium on Circuits and Systems (ISCAS2013), 2013, pp. 2763–2766.

[5] M. Broda and A. Frank, “Learning beyond the screen: assessing the impact of reflective artificial intelligence technology on the development of emergent literacy skills,” in E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, 2015, pp. 753–758.

[6] “ Osmo | Real Play, Real Learning.” [Online]. Available: https://www.playosmo.com/en/.

[7] J. Ma, “Defect detection and recognition of bare PCB based on computer vision,” in 2017 36th Chinese Control Conference (CCC), 2017, pp. 11023–11028.

[8] G. Mahalingam, K. M. Gay, and K. Ricanek, “PCB-METAL: A PCB Image Dataset for Advanced Computer Vision Machine Learning Component Analysis,” in 2019 16th International Conference on Machine Vision Applications (MVA), 2019, pp. 1–5.

[9] T. Kreienbühl, R. Wetzel, N. Burgess, A. Maria Schmid and D. Brovelli, "AR Circuit Constructor: Combining Electricity Building Blocks and Augmented Reality for Analogy-Driven Learning and Experimentation," 2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Recife, 2020, pp. 13-18, doi: 10.1109/ISMAR-Adjunct51615.2020.00019.

[10] Villanueva, A., Zhu, Z., Liu, Z., Peppler, K.A., Redick, T., & Ramani, K. (2020). Meta-AR-App: An Authoring Platform for Collaborative Augmented Reality in STEM Classrooms. Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.

[11] N. Thiwanka, U. Chamodika, L. Priyankara, S. Sumathipala, and G. T. Weerasuriya, “Augmented Reality Based Breadboard Circuit Building Guide Application,” in 2018 3rd International Conference on Information Technology Research (ICITR), 2018, pp. 1–6.

[12] juangallostra, “Augmented reality with Python and OpenCV (part 1),” Bites of code, 12-Sep-2017. [Online]. Available: https://bitesofcode.wordpress.com/2017/09/12/augmented-reality-with-python-and-opencv-part-1/.

[13] juangallostra, “Augmented Reality with Python and OpenCV (part 2),” Bites of code, 16-Sep-2018. [Online]. Available: https://bitesofcode.wordpress.com/2018/09/16/augmented-reality-with-python-and-opencv-part-2/.

[14] “ SamacSys Component Search.” [Online]. Available: https://componentsearchengine.com/.

[15] “ Spark AR Studio - Documentation - Facebook for Developers,” Facebook for Developers. [Online]. Available: https://developers.facebook.com/docs/ar-studio/.

[16] K. Wada, labelme. Github [Online]. Available: https://github.com/wkentaro/labelme.