

Article

Vision-Based Reconstruction of Electrical Schematics from Printed Circuit Board Photographs

Kamil Maliński  and Krzysztof Okarma * 

Faculty of Electrical Engineering, West Pomeranian University of Technology in Szczecin, 70-313 Szczecin, Poland; kamil.malinski@zut.edu.pl

* Correspondence: okarma@zut.edu.pl

Abstract

Reverse engineering of printed circuit boards is still largely manual when original computer-aided design documentation is unavailable. This paper presents a semi-automatic vision-based pipeline that prepares an editable KiCad schematic draft for use in an Electronic Design Automation (EDA) workflow from paired TOP and BOTTOM board images. The method combines color-profile estimation, pad and through-hole detection, trace segmentation, optical character recognition, component inference, an explicit evidence graph and schematic export with drawn wires. A separate readability step aligns symbols to a grid and reroutes the reconstructed nets with orthogonal wires; it does not change the reconstructed netlist. The primary quantitative evaluation used twelve synthetic KiCad fixtures and three solver configurations: the default sequential pipeline, an opt-in global component solver and an opt-in probabilistic contact solver. These fixtures provide controlled regression cases and are complemented by a small exploratory acquisition trial on real photographed boards. All configurations completed all runs and passed the export round-trip validation without falling back to label-only connectivity. This round-trip check confirms consistency between the internal reconstruction and the exported schematic, but it is reported separately from electrical correctness against the KiCad reference design. The stricter reconstruction-quality criterion still failed on four stress cases involving repeated component chains, long meandering variable-width traces, circular distractors near pads and two-sided transistor layouts. The probabilistic contact solver was therefore kept as an opt-in diagnostic mode rather than enabled by default; it reduced the global pin-to-pin netlist edit distance from 642 to 525 while preserving schematic export checks. The real-board trial indicates that pad and hole detection can transfer to simple photographs, with trace extraction remaining sensitive to uncontrolled illumination and weak copper contrast. The results support the use of the system as a human-in-the-loop reconstruction assistant and identify component grouping, trace-contact reasoning, real-photograph benchmarking and safe missing-edge activation as the main remaining research problems.



Academic Editor: Kefeng Ji

Received: 12 June 2026

Revised: 10 July 2026

Accepted: 13 July 2026

Published: 15 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: printed circuit board; reverse engineering; schematic reconstruction; netlist extraction; computer vision; image analysis; KiCad

1. Introduction

Printed circuit boards (PCBs) are usually documented by CAD projects containing schematics, layouts, netlists and bills of materials. In practice, such design files are often missing for legacy devices, prototypes, discontinued industrial equipment or repaired consumer electronics. In these cases, engineers reconstruct the circuit manually

by inspecting both sides of the board, identifying elements, following copper traces and rewriting the result into a schematic editor. This process is time-consuming, error-prone and strongly dependent on the operator's experience.

The motivation for the present work is the development of a tool that supports this reverse-engineering workflow using ordinary optical photographs or comparable TOP/BOTTOM board views. The goal is not to replace expert verification, but to transform a set of images into a structured draft: detected pads, trace masks, paired through holes, candidate components, a netlist and an editable schematic. Such a draft can reduce the amount of repetitive manual work and provide a controlled environment for corrections.

For example, on a small analogue board containing a connector, a resistor, a diode and a capacitor, the operator normally has to inspect both sides of the PCB, mark each pad, check whether a via connects the TOP and BOTTOM layers, and then redraw the circuit in an Electronic Design Automation (EDA) tool. The proposed workflow tries to automate the repetitive part of this task. It first proposes the pads, traces and text labels, then lets the user correct uncertain decisions, and finally exports a schematic that can still be edited in a conventional schematic editor.

The problem differs from the recognition of already drawn schematic diagrams. In schematic images, electrical connections are represented by symbolic wires and standardized symbols. In PCB photographs, the connectivity is only indirectly visible through copper traces, pads, vias, solder joints and sometimes hidden or partially occluded regions. Moreover, two sides of the board have to be aligned and interpreted jointly. Optical images also cannot reliably reveal traces under components, internal layers or connections hidden by solder mask. Therefore, the output must be treated as a hypothesis that should be verified and corrected by the user. In this paper, three outputs are kept separate. The *automatic* result is obtained without user edits. The *optimized* schematic is a readability-improved drawing of the same reconstructed netlist. The *manually corrected* schematic is produced only after explicit human changes to the evidence graph, components or contacts.

The contribution of this paper is an initial explainable pipeline for preparing a user-verifiable schematic draft from TOP and BOTTOM PCB images. The evaluation combines controlled synthetic KiCad fixtures for draft-generation tests with a small exploratory acquisition trial on photographed boards. In particular, the paper focuses on:

- A modular processing flow from PCB photographs to normalized views, trace masks, pads and OCR labels;
- An explicit graph representation joining pads, trace components, copper regions, TOP/BOTTOM hole pairs and inferred component pins;
- A conservative conversion of visual contacts into electrical net membership;
- A user-correctable editor in which human changes can be propagated to similar detections;
- Generation of an editable wired schematic and a separate readability-optimized schematic view.

The quantitative experimental part deliberately uses twelve synthetic KiCad fixtures. They are useful for controlled regression testing because the reference PCB and schematic data are known. The paper also includes a small qualitative trial on photographed boards to illustrate the current gap between synthetic and real-image conditions. Claims about practical deployment are therefore limited to the human-in-the-loop assistant scenario, with a broader real-photograph benchmark left as the next evaluation stage.

The rest of the paper is organized as follows. Section 2 discusses the state of the art. Section 3 formalizes the reconstruction task. Section 4 presents the image-processing pipeline, whereas Section 5 describes the graph-to-netlist and graph-to-schematic step,

including schematic autorouting. Section 6 reports the final synthetic evaluation and the exploratory real-photograph trial. Finally, Sections 7–9 summarize limitations, development directions and conclusions.

2. State of the Art

The considered task combines several research areas: PCB inspection, PCB reverse engineering, schematic image recognition, netlist extraction, OCR and graph reconstruction. However, the literature concerning these topics is strongly fragmented. Existing methods usually solve only one part of the problem, such as component detection, defect inspection, PCB netlist extraction from multi-modal imagery [1], schematic digitization or netlist prediction from already available schematic images [2].

2.1. PCB Analysis and Reverse Engineering

Vision-based PCB analysis is commonly considered in the context of inspection, defect detection, recycling and component recognition rather than schematic reconstruction. Recent PCB datasets, such as PCB-Vision [3], provide RGB and hyperspectral data with masks for selected component classes, which is valuable for training segmentation models. PCB component detection has also been studied from a hardware-assurance perspective; Zhao et al. [4] emphasized interpretable visual features such as color, shape and texture. Nevertheless, such datasets and detectors are not designed to recover electrical connectivity or to produce editable schematic diagrams.

A closely related but different direction is 3D PCB reconstruction from X-ray Computed Tomography. Phoulady et al. [5] presented an automated approach based on 3D semantic segmentation and subsequent netlist extraction. This confirms the importance of PCB netlist recovery, but the imaging modality and assumptions differ from low-cost optical photographs. X-ray CT may reveal internal layers and hidden traces, whereas the method proposed in this paper intentionally targets ordinary photographic input, accepting that some connections may remain uncertain or require user confirmation.

Classical PCB inspection approaches also use thresholding, morphology, contour analysis and machine learning to segment regions, detect defects or classify packages. Such methods are useful components of the proposed pipeline but do not by themselves solve the high-level reconstruction of circuit topology.

2.2. Schematic Image to Netlist Conversion

The most active recent line of work concerns converting images of circuit schematics into netlists. Hemker et al. [6] decomposed schematic analysis into component detection, line detection and text detection, followed by fusion into a high-level representation and netlist export. This decomposition is particularly relevant because a similar modular structure is needed for PCB photographs, although PCB traces and physical pads replace symbolic lines and component pins.

Several recent papers use deep learning or hybrid pipelines for schematic-to-netlist conversion. Hu et al. [7] proposed a graph attention model for parsing circuit topology from images. Fuchs et al. [8] combined U-Net preprocessing, pattern matching and graph extraction for digitizing printed circuit diagrams. Netlistify [2], Image2Net [9] and SINA [10] further demonstrate the growing interest in automatic circuit schematic image-to-netlist conversion.

These methods are important references, but they assume that the input image already depicts the schematic notation. In PCB reverse engineering, the image contains a physical realization of the circuit, where the component symbol is not visible and where

the correspondence between pads, traces, through-holes (vias) and schematic pins must be inferred.

2.3. Netlist-to-Schematic Layout and Orthogonal Routing

After extracting a netlist, an additional problem remains: the generated schematic should be readable. Netlists are precise but poorly suited for human inspection. Recent work on netlist-to-schematic generation, including Schemato [11], shows that the visual arrangement of elements is itself a non-trivial problem. General graph layout systems are also relevant here. The Eclipse Layout Kernel [12] and layered drawing with port constraints [13] are particularly interesting because schematic symbols have fixed pin ports rather than arbitrary node centers. In the proposed approach, the schematic layout is initialized from spatial relations observed on the PCB and then simplified to reduce wire length, crossings and symbol collisions, with net labels treated only as a fallback when a continuous routed connection cannot be drawn safely.

The optimization of a generated schematic can also be interpreted as an orthogonal routing problem. Classical grid-routing algorithms for VLSI and PCB design, such as Lee's maze router [14], provide the basic model of Manhattan paths on a grid with obstacles. Orthogonal graph drawing formalizes related readability objectives, including bend minimization and grid embeddings [15]. For schematic diagrams, the closest abstraction is orthogonal hyperedge routing, where one electrical net is drawn as a rectilinear tree connecting several ports rather than as independent pairwise wires. Wybrow et al. [16] and Helmke et al. [17] showed that this model is suitable for diagram and schematic editing, although the underlying optimization problem is difficult enough to require practical heuristics. This supports the design choice used here: symbols are first aligned to a grid, nets are treated as multi-terminal routing objects, and labels are used only as costly fallbacks rather than as the primary representation of connectivity.

2.4. Existing EDA Tools and Practical Workflows

Practical PCB reverse engineering is usually performed with general-purpose EDA tools. Free and web-based environments such as KiCad and EasyEDA, as well as commercial systems such as Altium Designer and Cadence OrCAD/Allegro, support schematic capture, PCB layout, library management, netlist import and design-rule checks [18–21]. KiCad also provides a command-line interface for exporting schematic netlists and running ERC checks [22]. These tools are mature editors and validators, but they typically assume that the user already knows the schematic or at least has a reliable netlist. They do not infer an editable schematic directly from ordinary photographs of an unknown physical board.

PCB legend and reference designators are relevant for reverse engineering because they often contain component references, polarity marks, pin-one marks and test-point labels. These markings are covered by practical PCB design standards and land-pattern conventions, including IPC-2221B [23] and IPC-7351B [24]. In the proposed system, OCR and silkscreen-like markings are therefore treated as evidence for component grouping rather than as decorative text.

In repair and service practice, engineers often combine these EDA tools with a microscope, a continuity tester, datasheets and manual notes. The proposed system targets the missing intermediate step: creating a first structured draft from images, so that the engineer starts from editable pads, nets and candidate components instead of an empty schematic sheet.

2.5. The Open Gap

To the best of our knowledge, there is no widely used fully automatic tool that reconstructs an editable electrical schematic directly from ordinary TOP and BOTTOM photographs of a PCB. Existing EDA tools support manual schematic capture, PCB layout, netlist import or design-rule checks, while research systems mostly address either PCB inspection or schematic-image parsing. Therefore, a practical and explainable pipeline for converting physical PCB photographs into a verifiable schematic draft remains an open problem.

This gap also limits the choice of direct baselines. A fair end-to-end baseline would have to accept paired TOP/BOTTOM board images and produce an editable schematic with component pins and nets. The works reviewed above provide useful partial references, but they generally solve different input–output problems: component localization, defect inspection, schematic-image parsing or netlist-to-schematic layout. For this reason, the experimental comparison in this paper emphasizes internal configurations of the same evidence pipeline and reports export consistency separately from electrical reconstruction accuracy.

3. Problem Formulation

The input consists of two RGB images:

$$I_T, I_B \in \mathbb{R}^{H \times W \times 3}, \quad (1)$$

where I_T and I_B denote the TOP and BOTTOM sides of the PCB after geometric normalization, H is the height of the image and W is its width. The expected output is an electrical hypothesis:

$$\mathcal{H} = (\mathcal{P}, \mathcal{V}, \mathcal{C}, \mathcal{N}, \mathcal{S}), \quad (2)$$

where $\mathcal{P}$ is the set of detected pads and holes, $\mathcal{V}$ is the set of paired TOP/BOTTOM through-holes (vias) or plated through-hole pads, $\mathcal{C}$ is the set of inferred components, $\mathcal{N}$ is the netlist and $\mathcal{S}$ is an editable schematic representation.

The automatic pipeline estimates this hypothesis directly from the images and is denoted by $\mathcal{H}_{auto}$. The schematic optimizer then produces a new drawing $\mathcal{S}_{opt}$ from the same reconstructed netlist $\mathcal{N}_{auto}$; it should not be interpreted as a second electrical reconstruction. If the user edits pads, component assignments, trace contacts or manual graph edges, the recalculated result is treated as a separate human-corrected hypothesis $\mathcal{H}_{manual}$. The evaluation in Section 6 reports automatic and optimized outputs separately and uses manually corrected examples only to demonstrate the interactive workflow.

The word “hypothesis” is important, since the system does not claim that the output is a final certified schematic. Instead, it produces the most plausible interpretation of the visible evidence. For instance, if a resistor has two visible pads and the OCR detects the label R1 nearby, the hypothesis may contain a resistor component R1 connected between two candidate nets. If one trace is hidden under the component body, the corresponding net edge remains uncertain and must be verified by the user.

For each side $s \in \{T, B\}$ the algorithm estimates a binary copper trace mask:

$$M_s(x, y) \in \{0, 1\}. \quad (3)$$

s is the side index, T and B denote the TOP and BOTTOM PCB sides, respectively, and $M_s(x, y)$ is the trace-mask value at pixel coordinates (x, y) on side s . A value of 1 denotes trace evidence and a value of 0 denotes background or non-trace evidence.

The mask is not treated as ground truth. It is only one source of evidence, later combined with pad positions, OCR labels, manually corrected objects and TOP/BOTTOM pairing.

The i -th detected pad or hole is represented as:

$$p_i = (s_i, x_i, y_i, r_i, t_i, q_i), \quad (4)$$

where the index i identifies one detected pad or hole in $\mathcal{P}$, s_i is the side, (x_i, y_i) is the center, r_i is the estimated radius, t_i is the pad type and q_i is a confidence value. Two pads p_i and p_j on the opposite sides may represent the same physical hole if the distance between them is lower than the threshold τ_d :

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq \tau_d. \quad (5)$$

In this expression, i and j index two candidate pads, (x_i, y_i) and (x_j, y_j) are their normalized image coordinates, and $d(p_i, p_j)$ is the Euclidean distance between their centers. The threshold τ_d is the maximum allowed center distance for a TOP/BOTTOM pad pair. This representation makes it possible to distinguish electrical pads from non-electrical objects. A mounting hole may be detected as a circular structure, but after the classification it should not create a schematic pin or a net connection. Conversely, a plated through-hole pad visible on both sides should be treated as one physical connection, not as two independent component pins.

The central task is to decide which pads belong to the same electrical net. This is formulated as a graph problem rather than as direct pixel grouping. Nodes represent pads, trace components, copper planes, component pins and nets. Edges encode visual or manual evidence. Connected components of the accepted electrical graph define candidate nets.

The reconstruction is deliberately conservative. False positive connections are more harmful than missing ones, because a missing connection can be added manually, whereas a false connection may silently corrupt the generated schematic. Hence, the pipeline illustrated in Figure 1 is designed to prefer uncertain isolated nets over aggressively merged nets.

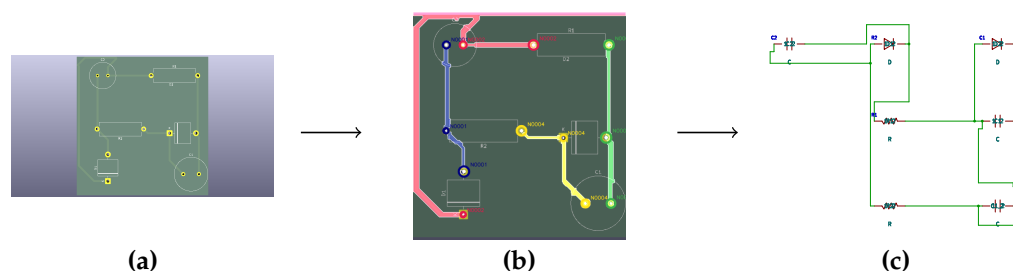


Figure 1. Illustrative processing sequence. (a) Raw PCB image. (b) Automatic analysis result. (c) Schematic generated after manual user correction and optimization.

4. Proposed Method

4.1. Overview

The proposed system is organized around two technically critical operations. The first operation is the recognition of PCB objects from TOP and BOTTOM images: pads, holes, trace regions, copper planes, text labels and candidate components. The second operation is the conversion of the reconstructed netlist into an editable KiCad schematic with explicit wires. The remaining stages, such as image import, quality scoring, correction editing and report generation, support these two operations as presented in the pseudocode in Table 1 as well as the flowchart illustrated in Figure 2.

The complete automatic pipeline contains eight stages:

- Image import and quality assessment;
- Board normalization and alignment of the BOTTOM image to the TOP coordinate system;
- Detection of pads, holes and vias;
- Trace and copper-plane segmentation;
- OCR-based extraction of reference designators and values;
- Construction of an intermediate electrical graph;
- Component inference and user correction;
- Generation of a netlist, a direct schematic draft and a readability-optimized schematic.

Table 1. Condensed pseudocode of the automatic reconstruction pipeline.

```

1  load TOP and BOTTOM board images
2  normalize scale, crop board region and align BOTTOM to TOP coordinates
3  estimate soldermask and copper color profile from HSV and CIELab samples
4  for each side do
5    detect circular pads with Hough transform
6    add contour and dark-hole fallback pad candidates when needed
7    remove duplicate and color-inconsistent pad candidates
8    generate trace-mask hypotheses from color, Canny and adaptive filters
9    vote trace hypotheses and extract connected trace and plane regions
10   run OCR variants and assign text boxes to nearby pad groups
11  pair TOP and BOTTOM through-hole pads by position and radius
12  create graph nodes for pads, traces, planes, components and nets
13  add via, trace-contact, plane-contact and manual-correction edges
14  infer component candidates from OCR, silkscreen, rows and footprint geometry
15  optionally solve mutually exclusive component or contact hypotheses
16  union active electrical edges into reconstructed net groups
17  assign exported symbol pins to physical pad nodes and nets
18  generate the direct KiCad schematic with routed wires
19  validate export by KiCad round-trip and diagnostic checks
20  optimize symbol placement and reroute without changing the netlist

```

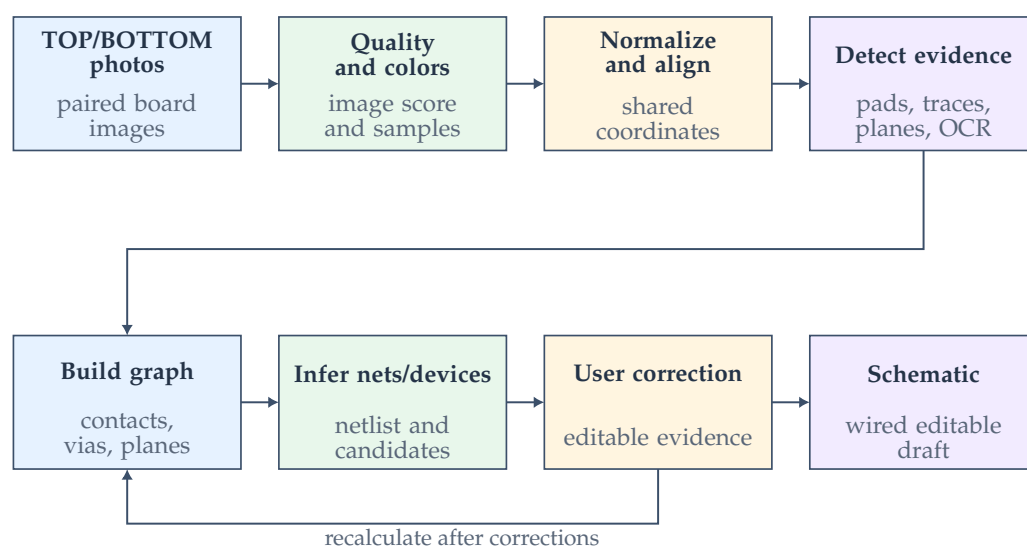


Figure 2. Flowchart of the implemented semi-automatic PCB-to-schematic pipeline.

The same deterministic pipeline is used in the automatic mode and in the synthetic regression tests. In the interactive mode, the user may additionally correct color samples, pad detections, OCR labels, trace masks, pad pairs, components and manual connections, and then recalculate the schematic. This is important because a PCB photograph does not contain all electrical information and because some errors have a very different risk profile. A missing trace usually creates an isolated pad that can be repaired. A false short can create a plausible but incorrect schematic and must therefore be avoided or at least explained.

4.2. Color Profile and Preprocessing

The recognition stage starts from normalized PCB images stored in OpenCV's BGR channel order. Conceptually, these are RGB color images and are converted by the implementation to grayscale, HSV and CIELab representations. HSV is used for hue, saturation and brightness decisions, whereas CIELab is used for perceptual color-distance thresholds. The program estimates dominant colors with $K = 4$ k-means clusters sampled from the image, following the standard partitioning model introduced by MacQueen [25]. The largest saturated cluster is treated as the soldermask candidate and the most distant bright or copper-like cluster is treated as the initial copper candidate. User-selected samples can override both values.

The fixed value $K = 4$ was selected because the rendered and photographed board views normally contain four visually dominant material classes: soldermask, copper or exposed metal, silkscreen and background or shadow. Larger values split highlights and antialiased borders into separate clusters and made the initial thresholds less stable on small fixtures. The implemented k-means termination criteria are 35 iterations or a centroid movement below 0.3 in the sampled color space, with two k-means++ initializations. These values are used only to obtain material hypotheses; later stages still combine color, shape and connectivity evidence.

For a pixel x and a reference color c , the CIELab distance is:

$$d_{\text{Lab}}(x, c) = \|\text{Lab}(x) - \text{Lab}(c)\|_2. \quad (6)$$

Here, $\text{Lab}(\cdot)$ is the CIELab representation of a pixel or color sample and $\|\cdot\|_2$ denotes the Euclidean norm. For HSV-based comparisons, let $\text{HSV}(x) = (H_x, S_x, V_x)$ and $\text{HSV}(c) = (H_c, S_c, V_c)$. The circular hue distance and the full HSV distance are:

$$d_H(x, c) = \min(|H_x - H_c|, 180 - |H_x - H_c|), \quad (7)$$

$$d_{\text{HSV}}(x, c) = \sqrt{d_H(x, c)^2 + (S_x - S_c)^2 + (V_x - V_c)^2}. \quad (8)$$

The variables H , S and V denote hue, saturation and value in OpenCV HSV coordinates, where hue is represented on the interval $[0, 180]$ and saturation and value on $[0, 255]$. The initial CIELab and HSV thresholds are derived from d_{Lab} and d_{HSV} between the estimated soldermask and copper samples. During segmentation, the implementation also uses d_H with separate saturation and value gates. This makes the method more robust than fixed green-board thresholds. The current implementation also detects non-green saturated soldermasks. When the dominant background hue is outside the green range and the trace color is either close in hue to the soldermask or significantly darker, an additional trace-color hypothesis is enabled.

Several simpler strategies were tested and rejected. A fixed rule for green soldermask worked on the first examples but failed on red, blue and brown boards. Pure grayscale thresholding reacted strongly to silkscreen text and shadows. A single global morphological closing improved some missing traces but created false shorts on dense boards. For this reason, the current pipeline keeps multiple weak hypotheses and combines them only after filtering.

4.3. Pad and Hole Detection

Pad detection is performed before trace-to-net grouping because pads define the electrical points that must later appear in the netlist. The input image is converted to grayscale and smoothed with a 5×5 median filter. Circular pad candidates are detected with a Hough-transform-based circle detector, a standard technique for parametric shape

detection [26]. The radius range is derived from the smaller image dimension. For narrow and elongated non-green boards, the maximum radius is expanded because through-hole annular rings may appear larger than in the original green-board test set.

In the current configuration, the minimum Hough radius is $\max(3, \lfloor \min(W, H)/300 \rfloor)$ and the maximum radius is $\max(8, \lfloor \min(W, H)/35 \rfloor)$, where W and H are the image width and height. The detector uses $dp = 1.2$, a minimum center distance of twice the maximum radius, $param1 = 80$ for the internal Canny stage and $param2 = 18$ for circle accumulation. The radius range is image-scale-dependent rather than absolute because the same synthetic fixtures are rendered and cropped at different sizes. The accumulator threshold was kept permissive so that borderline annular rings remain candidates; later color and duplicate filters remove many false positives.

If too few circles are found, a contour fallback is applied. It combines local adaptive thresholding with a dark-percentile threshold, extracts contours and keeps only sufficiently circular candidates. On non-green saturated boards, an additional dark-hole fallback searches for dark circular drill holes and expands them into pad candidates. Finally, duplicates are removed by comparing pad centers and radii, and suspicious silkscreen-like candidates can be rejected using local HSV statistics.

The fallback is triggered when fewer than four circular candidates are found. It uses adaptive thresholding with a 31×31 local window and constant $C = 5$, together with a dark threshold based on the lower grayscale percentile. The dark-hole fallback uses a conservative drill threshold limited to the range 24–78 and expands the detected drill radius by a factor of 2.15 to recover the annular pad region. These constants were chosen from the fixture scale: the first fallback should recover small dark pad centers, whereas the expansion factor approximates the rendered relation between the drill hole and the through-hole copper ring. Candidates are then merged when their centers and radii overlap strongly, which prevents the Hough and contour paths from creating duplicate pads.

The practical ordering is therefore:

1. Grayscale conversion and median blur;
2. Hough circle detection;
3. Contour fallback when the circle count is too low;
4. Dark-hole fallback for low-count non-green boards;
5. Duplicate suppression;
6. Local color filtering for small candidate sets.

This order was chosen because Hough circles are precise when the annular ring is visible, while contour and dark-hole fallbacks recover boards where the ring and the soldermask are visually close. Applying the color filter too early was less reliable, because pads and traces can have almost the same hue as the soldermask in KiCad-style renders.

4.4. Trace Segmentation

The trace mask is built by voting over several classical filters. The automatic configuration currently evaluates $K = 15$ variants. For each variant, the following evidence is extracted:

- A CIELab/HSV copper-color mask;
- An under-soldermask trace mask, where the hue is close to the soldermask but the brightness or the CIELab distance changes;
- Bright-saturated and dark-saturated percentile masks;
- Canny edges [27] computed on the CLAHE-enhanced grayscale image, following the local histogram equalization family of methods described by Pizer et al. [28];

- Adaptive local thresholding, related to classical local thresholding methods [29,30], gated by nearby color evidence.

For each filtering configuration k , the candidate trace mask is denoted as $M_s^{(k)}$. The final mask after voting is defined as:

$$M_s(x, y) = \begin{cases} 1, & \sum_{k=1}^K M_s^{(k)}(x, y) \geq \tau_m, \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Here s denotes the trace class, (x, y) is the image-pixel location, k indexes one of the K filtering configurations, $M_s^{(k)}(x, y)$ is the binary trace vote produced by configuration k at that pixel, and τ_m is the minimum number of votes required to accept $M_s(x, y) = 1$. The threshold τ_m is set to 35% of the evaluated filters, rounded up. The result is cleaned with morphological close/open operations and connected component filtering. Electrical connectivity uses four-neighbor connected components, because diagonal one-pixel contact is often a rendering artifact and should not merge nets.

The 15 variants are selected from bounded parameter lists rather than random sampling. The CLAHE clip limit is drawn from 1.2 to 5.0, Canny thresholds from the range 22/75 to 90/240, adaptive-threshold windows from 21 to 91 pixels and adaptive constants from -2 to 12. Bright-percentile masks use percentiles from 52 to 82. With $K = 15$, the 35% rule accepts pixels supported by at least six filters. This threshold has been selected as a compromise between two observed failure modes: a higher vote threshold fragments weak traces on non-green boards, whereas a lower one admits isolated silkscreen and shadow edges. The post-processing uses small 3×3 -pixel morphological kernels so that it removes single-pixel artifacts without joining adjacent parallel traces.

For non-green soldermasks, a second mask is computed from the dominant trace-color hypothesis. This mask accepts pixels close to the trace sample and also pixels that are close to the soldermask hue but shifted in CIELab distance, saturation or value. Components are then filtered by shape: elongated regions, large enough regions and regions with low rectangular fill are kept, whereas small compact specks are rejected. This shape filtering was introduced after tests in which the color of the trace and the color of the board background were visually similar.

The system intentionally avoids one “tempting shortcut”: it does not use a large closing kernel to join all nearby trace fragments. Such a rule improves some over-split traces but produces false shorts on boards with vias, ground planes and close parallel tracks. Instead, small gaps are bridged in the graph layer only when supported by pad-trace contact evidence.

4.5. Text-Assisted Component Recognition

Component recognition is treated as a scoring problem rather than as a single detector. Optical character recognition (OCR) is performed with the Tesseract v5.3.0 OCR engine [31] on several preprocessed variants and rotations. The preprocessing variants use scaling, CLAHE, Otsu or adaptive thresholding and a restricted whitelist of short PCB labels. Otsu thresholding is used as a classical global thresholding baseline [32], while adaptive thresholding helps when silkscreen contrast changes locally. Detected text boxes are assigned to nearby pads and used as hints, not as guaranteed truth.

The OCR variants scale candidate text regions by factors from 2.0 to 4.0, apply CLAHE clip limits from 2.0 to 4.5, optionally use a 3×3 blur and evaluate Otsu, inverse Otsu and adaptive thresholds. Tesseract is run with page segmentation modes 6, 11 and 13 and a whitelist containing uppercase letters, digits, underscores and hyphens. Text is considered for nearby pads inside a scale-dependent radius limited by board diagonal, pad radius and

text-box size. This radius is intentionally generous because reference designators are often printed beside, not inside, the component footprint; the later component score decides whether the text is strong enough to affect the model.

The current component inference order is:

1. Group pads directly supported by OCR reference designators such as R1, C2, D1, L3, U1 or J2;
2. Search for local silkscreen or fabrication-line evidence connecting nearby pads;
3. Detect horizontal or vertical pad rows for connector-like groups;
4. Group close unassigned pads into discrete or multi-pin devices;
5. Infer two-pin devices from geometry and net evidence;
6. Create test-point placeholders only for remaining pads that belong to non-singleton nets.

The type of a component is inferred from the OCR prefix and the number of pads. For two-pin components without reliable text, the conservative fallback is a generic resistor-like device. Footprint hints are selected from pad radius and spacing, separating SMD-like and through-hole-like geometry. These hints are not final library assignments; they are intended to make the generated KiCad draft easier to inspect and correct.

Each candidate receives additive evidence terms. The OCR-supported reference designators receive the highest local bonus, followed by silkscreen evidence, row geometry, footprint geometry and net-pattern consistency. Manual edits receive an additional bonus, while fallback-only candidates receive a penalty. The score is also adjusted by whether the grouped pads belong to distinct reconstructed nets and whether the reference prefix is plausible for the candidate pin count. These weights are heuristic but explicit: they encode the observed reliability order of the evidence sources, with text and user confirmation above geometry, and geometry above fallback grouping.

For diagnostics, each inferred component stores the source of the decision (OCR/refdes, silkscreen, row_geometry, footprint_geometry, net_pattern, manual or fallback), a score, short decision reasons and the physical pad nodes used by every symbol pin. This separates four failure modes that were previously hidden inside one component-count number: pad detection, pad-to-component assignment, component type and pin order.

Using the OCR alone was not sufficient. Text may be absent, hidden, rotated, partially clipped or confused with board markings. Pure geometric grouping was also not sufficient, because connectors, resistors and repeated filter stages can have similar pad spacing. The implemented order therefore gives local text and silkscreen evidence priority, then uses geometry as a fallback.

This risk is illustrated by one rejected experiment. A normalization that treated short OCR strings such as 04 as possible diode labels could recover a plausible reference designator on one board, but it also introduced false component evidence and worsened pin-to-pin metrics. The production pipeline therefore keeps the OCR as a candidate source only; uncertain text must be supported by footprint, silkscreen or net evidence before it changes the reconstructed component model.

4.6. Solver Configurations

The main solver comparison uses three configurations that operate on the same detected evidence but make different decisions about component grouping and electrical contacts. Additional runtime ablations disable OCR evidence or ground-plane handling after the same image-processing stages. None of these configurations uses the KiCad reference netlist during reconstruction; reference data are used only after export, for evaluation.

The default configuration is the sequential recognizer with the contact solver disabled. Component hypotheses are considered in a fixed order: OCR-supported groups first, then silkscreen and row evidence, then local footprint geometry, net-pattern hints and finally conservative fallback components. Once a pad is assigned to a component by a higher-priority rule, later rules cannot reuse it. Electrical nets are built only from accepted deterministic evidence: trace contacts, plane contacts, via pairs and manual corrections. This configuration is intentionally conservative and easy to inspect, which is why it is the default program mode.

The global component solver changes only the component-selection stage. Instead of accepting hypotheses one by one, the program first builds a pool of competing component candidates from OCR, silkscreen, row geometry, footprint geometry and net-pattern evidence. Then the solver chooses a mutually compatible subset with the highest total score. Hard constraints prevent a physical pad from being used by two different components, prevent duplicate reference designators and reject impossible pin counts. Penalties are applied to weak fallback evidence, ambiguous grouping and component choices that would create high-risk electrical interpretations. This global view can remove many false pin-to-net relations caused by locally plausible but mutually inconsistent component groups. It is still opt-in because an incorrect scoring balance can replace a locally correct grouping with a globally convenient but wrong one.

The candidate scoring used by the global solver is therefore deliberately asymmetric. Strong OCR or manual evidence can outweigh a small geometric penalty, but fallback candidates are penalized unless they explain pads that would otherwise remain unassigned. A component candidate is rejected outright when it reuses a physical pad already selected by a higher-scoring candidate, duplicates an existing reference designator or has a pin count incompatible with the selected symbol family. This keeps the optimization close to a constrained set-packing problem rather than an unconstrained score maximization.

The third evaluated configuration is a probabilistic contact solver, not a second component solver. It keeps the sequential component model and optimizes selected electrical edges in the graph. Each candidate contact has a confidence derived from geometry, trace-mask support, pad radius, layer pairing and local diagnostic evidence. The solver may deactivate risky accepted contacts or activate a small number of local pad–trace proximity candidates when the evidence is strong enough and the resulting graph remains within safety constraints. This is useful for evidence-driven correction of missing or false trace contacts, but it can also create plausible false shorts if used too aggressively. For that reason, broader missing-edge hypotheses such as gap-bridging remain diagnostic evidence only and the probabilistic contact solver remains an opt-in mode.

The contact solver uses explicit activation and deactivation scores. Accepted trace contacts in very large non-plane components are candidates for deactivation when they touch many pads, create many possible pin pairs or are marked as relaxed non-green contacts. Missing-edge candidates are limited to local pad–trace proximity and short pad-gap evidence; they are considered only when the distance is below 45 pixels and the merged group remains small. A candidate is activated only if its distance bonus, evidence-type bonus and single-pair bonus overcome the false-merge penalty. The current safety limits allow only a small number of such activations per board, which is why the mode is useful diagnostically but is not used as the default reconstruction path.

4.7. Interactive Correction and System Demonstration

The algorithmic stages described above define the reconstruction method. The desktop interface described in this subsection (illustrated in Figure 3) is an implementation layer that exposes the intermediate evidence to the operator. It is included to show how the

semi-automatic workflow can be used in practice, but the automatic evaluation discussed in Section 6 is computed before any manual correction.

The tool is intended to operate automatically, but almost every stage can be corrected. The editor displays both PCB sides simultaneously, highlights paired pads using the same colors and allows the user to edit pads, pair vias, classify pad types, adjust trace masks, create components and assign OCR labels. When a correction is made, the system can search for similar detections and propose applying analogous changes. This supports the practical case where the same error pattern occurs many times on one board.

The correction environment is implemented as a desktop application for PC workstations with a graphical user interface and can be run on Windows, macOS and Linux. In addition to manual editing, the application performs automatic diagnostics of electrical errors, including missing connections, suspicious shorts, unpaired vias and pads that remain outside the reconstructed netlist. After accepted changes, the connectivity graph is recalculated and the schematic layout can be regenerated and optimized so that the visual representation remains consistent with the corrected electrical model.

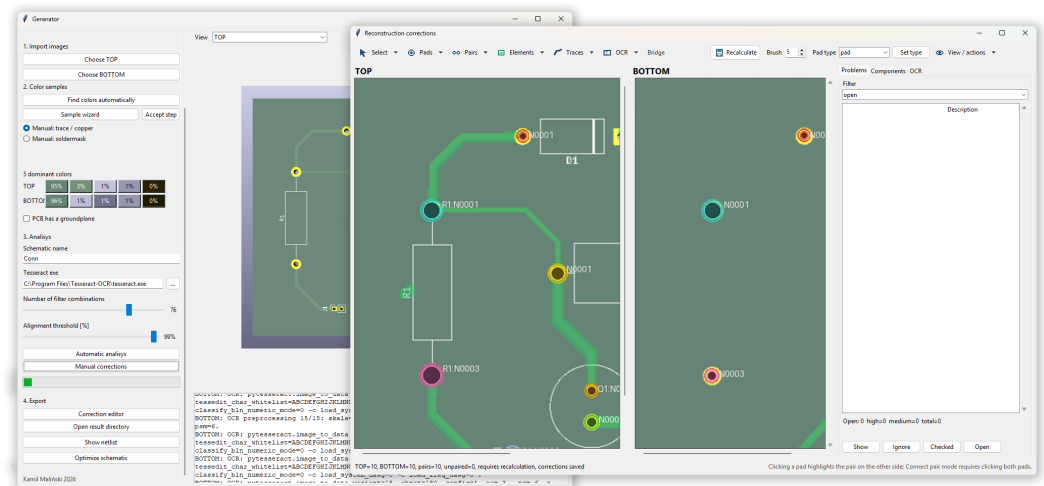


Figure 3. Screenshot of the developed PCB analysis application with the manual correction editor open.

5. Generating Electrical Schematics from Netlists

The second central part of the method is the transformation of uncertain visual evidence into an editable schematic as illustrated in Figure 4. The program does not draw wires directly from bitmap traces. Instead, it first constructs a typed electrical graph, extracts net groups, assigns component pins to these nets and only then generates a KiCad schematic.

5.1. Electrical Graph

The intermediate graph is defined as:

$$G = (V, E), \quad (10)$$

where V contains pad nodes, trace-component nodes, copper-plane nodes, component nodes and net nodes created after grouping. Edges store the source of electrical evidence:

$$E = E_{trace} \cup E_{plane} \cup E_{via} \cup E_{pin} \cup E_{manual}. \quad (11)$$

Only active electrical edges are used for union-find grouping. Rejected or weak edges remain available in the diagnostics for later inspection.

This graph layer is important because image processing confidence and electrical confidence are not the same. A pixel mask may be visually plausible but electrically dangerous if it creates a false short. A manually confirmed connection may be electrically strong even if the local image evidence is weak. The graph therefore stores the evidence type, confidence, activity flag, geometric sample points and textual explanation for every edge. This diagnostic record makes it possible to trace each reconstructed net back to the visual and manual evidence that created it.

5.2. Pad-Trace Contact

For a pad p_i and a trace component C_j , the strict contact rule checks whether trace pixels exist inside a radius-dependent disk around the pad. The minimum distance is:

$$d_{\min}(p_i, C_j) = \min_{(x,y) \in C_j} \sqrt{(x - x_i)^2 + (y - y_i)^2}. \quad (12)$$

Here i indexes pads, j indexes trace components, (x_i, y_i) is the center of pad p_i , and (x, y) denotes a trace-mask pixel belonging to component C_j . The contact edge is accepted when $d_{\min}(p_i, C_j)$ is below a radius-dependent threshold and the local component contains enough pixels. This prevents a trace that only passes near a pad from being connected because of a large rectangular bounding box.

Before trace components are labeled, local pad disks are removed from the trace mask for paired through-hole (via) points. This prevents one round pad from becoming an artificial bridge between multiple trace fragments. However, on non-green boards this can also cut the trace too aggressively. The current implementation therefore adds a controlled relaxed contact mode for paired THT pads. It can attach a larger horizontal trace component near a pad only when the board is detected as non-green and saturated. This rule was introduced after early failures on boards with low copper-background contrast: it recovered missing connections while keeping the green soldermask regression cases stable.

An earlier attempt to simply increase the allowed distance between all trace components improved some over-split nets but caused false merges on boards with dense passive sections, sparse isolated nets and large copper pours. That rule was removed. The present rule is narrower: the additional contact is limited to paired pads, non-green soldermask evidence and shape-compatible trace components.

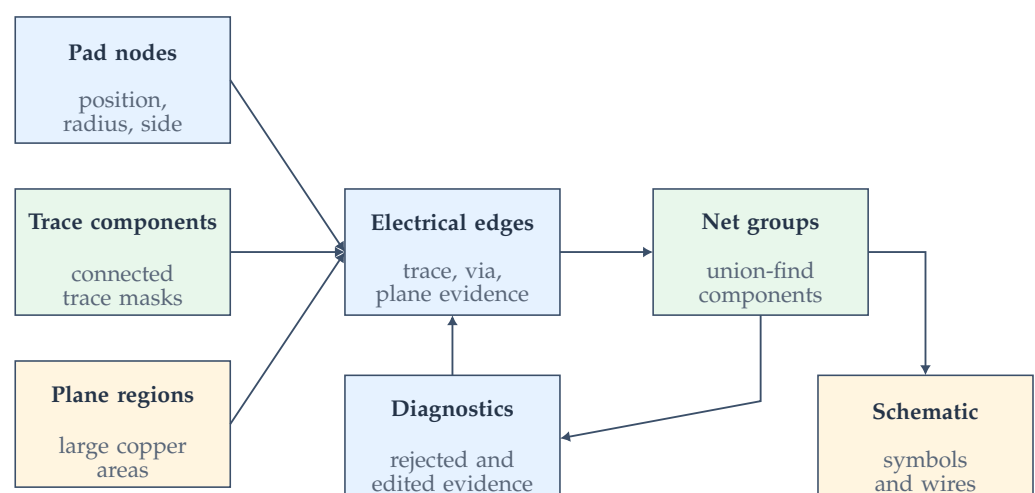


Figure 4. Conversion of visual detections into a typed connectivity graph and an editable schematic.

5.3. Plane and via Handling

TOP/BOTTOM hole pairs are represented as *via_pair* edges. A physical through-hole pad therefore joins the TOP and BOTTOM pad nodes even when the visible trace evidence appears only on one side. Copper-plane regions are detected separately and converted into plane nodes. A plane contact is active only if the same physical pad is not already explained by trace evidence on one of the paired sides. This conservative rule avoids a common false short where a large plane-like mask visually surrounds pads that should remain separate.

5.4. Net Extraction

After trace contacts, plane contacts, via pairs and manual connections are created, disjoint-set grouping is applied:

$$\text{net}(p_i) = \text{component}_G(p_i). \quad (13)$$

Here G is the reconstructed connectivity graph after trace, plane, via and manual edges have been inserted, whereas $\text{component}_G(p_i)$ denotes the connected component containing the i -th pad node p_i . Each connected component containing at least one pad becomes a candidate net. Plane-backed groups may be named as ground candidates, while other groups receive generated net identifiers. The resulting netlist is a debuggable intermediate representation and also the input to component-to-schematic generation.

5.5. Component Pin Model

A component candidate is represented by:

$$c_k = (\text{ref}_k, \text{type}_k, \text{value}_k, F_k, P_k, A_k), \quad (14)$$

where ref_k is the reference designator, type_k is the component type, value_k is the value, F_k is a footprint hint and P_k is an ordered list of associated pads. The additional map A_k records which physical graph pad node is used by each exported symbol pin. Pins are assigned to nets through the current pad-to-net mapping:

$$\text{pin}_m(c_k) \mapsto \text{net}(p_m). \quad (15)$$

Here c_k is the k -th component candidate, $\text{pin}_m(c_k)$ is its m -th exported symbol pin, and p_m is the physical graph pad node assigned to that pin by the map A_k . Automatically detected components may be replaced by manually defined components. This is necessary because TOP and BOTTOM views of the same physical pad must not be considered as independent pins, and because OCR or visual grouping may produce duplicate references.

The explicit pin-to-pad map is also used during validation. When a false or missing pin-to-pin relation is reported, the diagnostic record can list the expected KiCad net, the reconstructed program net, the component pins involved and the graph evidence that produced the relation. This makes component assignment and pin-order errors distinguishable from trace-contact or plane grouping errors.

5.6. KiCad Schematic Generation

The base schematic is generated as an editable KiCad schematic with a small local symbol library. The schematic layout is initialized from the PCB geometry, but it is not intended to reproduce the board layout. A readable schematic should emphasize logical relationships, not physical routing.

The exporter performs the following steps:

1. Group component pins by reconstructed net;
2. Place symbols using PCB-derived coordinates and then separate collisions;
3. Reserve short pin stubs before global routing;
4. Route each multi-pin net with orthogonal wires;
5. Track occupied routing cells per net;
6. Use fallback outside lanes if the direct route is blocked;
7. Add a local net label only as a name anchor on an already routed net;
8. Export and validate the schematic with KiCad CLI when available.

The most important design decision is that labels are no longer used as a substitute for wires. A previous version avoided drawing difficult routes and placed labels at pins instead. This made the draft electrically valid in KiCad but practically useless for reverse engineering, because the user still had to trace the same connections manually. The current exporter draws wires for multi-pin nets and uses labels only to preserve names.

The routing layer uses an occupancy grid. A cell occupied by one net may be reused by the same net, but not by another net. Short stubs near symbol pins are reserved first so that the later global route does not block its own endpoint. If A*-style orthogonal routing cannot find a clear path, a deterministic fallback creates an outside lane assigned to the net. This is not a replacement for a full schematic-layout engine, but it prevents the most common failure mode: unrelated nets accidentally sharing a wire segment.

The direct exporter uses the KiCad schematic grid as the unit system. Symbols are placed on a 2.54 mm grid and routed on a 1.27 mm routing grid. A short pin stub is always reserved before global routing, and occupied cells are tracked per net. These values were selected because they match the common KiCad schematic editing grid while still allowing the router to pass between symbols when a one-grid-unit symbol placement would be too coarse. They also keep the exported schematic easy to edit manually: wires and labels land on predictable grid positions instead of arbitrary pixel-derived coordinates.

5.7. Optimized Schematic Autorouting

The base KiCad schematic is intended to be electrically traceable, whereas the optimized schematic is intended to be easier to inspect. The optimization step does not change the reconstructed netlist. It first records which symbol pins belong to each net, removes the existing wire geometry, aligns symbols to a regular grid and then routes each net again with orthogonal segments. At least five grid units are reserved between symbol bounding boxes to leave channels for wires.

The default optimization budget is 30 s per schematic. The placement search uses a population of 96 layouts with a mutation rate of 0.18, and each candidate layout is routed with up to 24 net-order variants. The scoring function rewards shorter wires and penalizes wire crossings, wire–symbol conflicts, close symbol placement and label jumps. In the current configuration, crossings are much more expensive than length because a slightly longer but readable schematic is preferable to a compact drawing that is difficult to verify. The optimizer is therefore a readability pass over a fixed netlist, not a second reconstruction algorithm.

The routing problem is treated similarly to a single-layer PCB autorouting task, but with schematic-specific costs. Symbols are obstacles, symbol pins are ports and each electrical net is a multi-terminal object. This is close to the orthogonal hyperedge-routing model used in diagram and schematic layout systems [16,17]. The router therefore builds a rectilinear tree for each net instead of drawing all pin pairs independently. Candidate paths are evaluated by length, number of bends, wire–symbol conflicts, backtracking and use of labels, following the same readability concerns that motivate bend minimization in

orthogonal graph drawing [15]. A label jump is allowed only as a costly fallback, analogous to a via in a single-layer routing metaphor; if a continuous wire can be drawn, the wire is preferred. The grid search itself follows the classical maze-routing idea introduced for layout routing by Lee [14].

After routing, redundant segments are reduced to a tree so that a single net does not contain visually confusing loops. The optimized schematic is kept separate from the direct export because it changes only the drawing, not the reconstructed netlist. The same optimization can be applied after manual corrections, which lets the operator compare the direct reconstruction with a cleaner view of the corrected circuit.

The schematic export is validated by re-reading the generated schematic netlist and comparing its pin groups with the internal program model. This round-trip test checks whether the schematic preserves the reconstructed netlist. It is intentionally separate from the stricter comparison against the original KiCad reference design. A round-trip pass means that the exporter is consistent; it does not prove that the visual reconstruction was fully correct. The effect of schematic autorouter after manual corrections is presented in Figure 5.

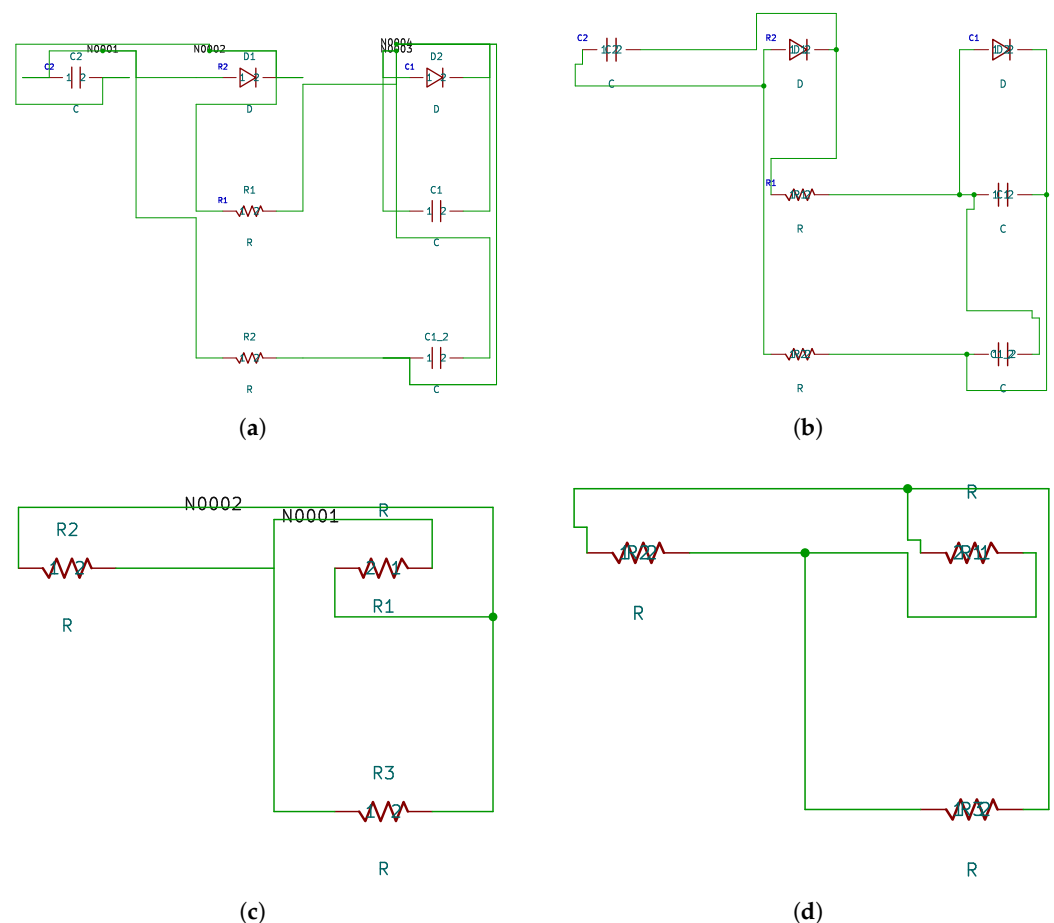


Figure 5. Effect of the schematic autorouter after manual correction recalculation. (a) Crossover board after correction. (b) The same crossover board after optimization. (c) Long meandering trace after correction. (d) The same meandering trace after optimization. The optimized views preserve the corrected netlist while rearranging symbols and wires for readability.

6. Experimental Protocol and Results

The quantitative evaluation used twelve synthetic KiCad fixtures and six configurations supported by the runtime. The baseline configuration is the default sequential pipeline: component hypotheses are selected by the sequential recognizer

and the contact solver is disabled. Three solver variants were measured separately: a global component solver, a risk-averse contact solver and a probabilistic contact solver. Two additional ablations disabled OCR evidence or ground-plane handling. These modes are reported as diagnostic experiments and are not used as the default program configuration. An additional exploratory trial on two photographed boards is reported qualitatively in Section 6.5.

The comparison is therefore an internal solver and ablation comparison, not a claim of superiority over a complete external PCB-photograph-to-schematic system. As discussed in Section 2, directly comparable end-to-end tools are not readily available. The reported configurations instead separate the stable schematic-export layer from the less mature recognition and contact reasoning layers.

6.1. Dataset and Evaluation Criteria

The evaluated dataset contains twelve synthetic boards: *Cascade*, *Connector*, *Crossover*, *Divider*, *Elements*, *Filter*, *Floating*, *Groundplane*, *Meander*, *Round*, *Simple* and *Transistors*. The fixtures are not intended to model final electronic products. They isolate specific visual and structural difficulties: paired through-holes, non-green solder masks, copper pours, deliberately tangled traces, repeated components, component-side assignment and schematic export.

The main dataset illustrated in Figure 6 is synthetic and controlled. It contains KiCad-derived board views with known reference schematics, which makes strict regression testing possible. This is intentional at the current stage of the work: the synthetic fixtures isolate pad detection, trace reasoning, component grouping and schematic export before uncontrolled acquisition effects are introduced. The results should therefore be read as validation of the current pipeline on a controlled benchmark and as preparation for a broader real-photo benchmark, not as a complete deployment study.

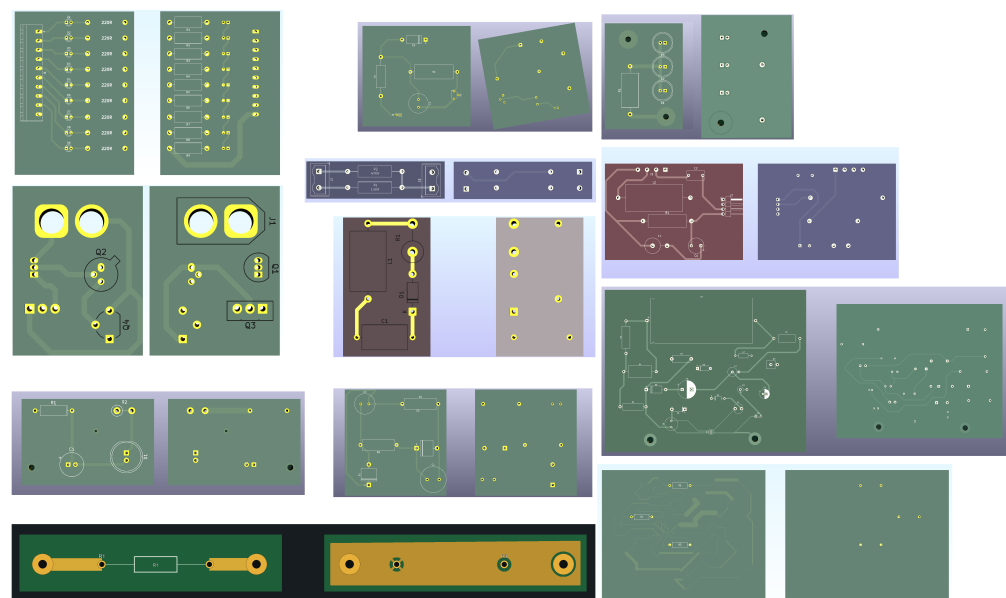


Figure 6. TOP and BOTTOM renders of all synthetic test boards.

Pad detection is evaluated using true positives (*TP*), false positives (*FP*) and false negatives (*FN*). The PCB image background does not define a useful bounded set of true negatives; hence the reported object-level accuracy is:

$$ACC = \frac{TP}{TP + FP + FN}. \quad (16)$$

The netlist evaluation uses pin-to-pin relations induced by the KiCad reference netlist (reference data may be found in Table 2) and by the reconstructed component assignments. Therefore, the netlist edit distance (NED) is defined as the sum of false positive and false negative pin relations. Schematic export is checked separately by round-trip validation, label-only net detection, symbol collision detection and wire-routing preflight. Round-trip validation means that KiCad can re-read the exported schematic and that the exported schematic preserves the program's reconstructed netlist. It does not mean that the reconstructed netlist is electrically identical to the reference KiCad design; that stricter question is measured by the pin-to-pin relation metrics.

Table 2. Reference data extracted from the KiCad projects.

Board	Components	Pads	Vias	Zones	Segments	Nets
Cascade	19	46	0	0	70	11
Connector	6	13	0	1	23	4
Crossover	6	12	0	0	23	4
Divider	4	8	0	0	10	3
Elements	20	40	2	0	112	11
Filter	7	18	0	0	40	7
Floating	4	8	2	1	7	4
Groundplane	4	8	2	2	7	3
Meander	3	6	0	0	155	2
Round	4	8	0	0	6	4
Simple	1	2	2	1	4	2
Transistors	5	14	0	0	32	6

6.2. Baseline Results

The default sequential pipeline completed all twelve boards without runtime errors. All generated schematics passed round-trip validation and used drawn wires rather than label-only net representation. These checks confirm that the exporter preserves the program's own reconstruction as an editable schematic draft. They do not prove that the reconstructed electrical content matches the KiCad reference.

In Table 3, RT is an export-consistency result only; reference electrical agreement is reported later by pin-relation and net-overlap metrics.

Table 3. Shared export and diagnostic checks for the six runtime-supported configurations. RT—round-trip validation; Opt.—optimized schematic generated; Q—boards passing the current reconstruction-quality thresholds.

Configuration	Errors	RT	Opt.	Labels	Routing	Q
Baseline	0	12/12	12/12	0	0	8/12
Global component solver	0	12/12	12/12	0	0	8/12
Risk-averse contact solver	0	12/12	12/12	0	0	8/12
Probabilistic contact solver	0	12/12	12/12	0	0	8/12
No OCR evidence	0	12/12	12/12	0	0	8/12
No ground-plane handling	0	12/12	12/12	0	0	8/12

The stricter reconstruction-quality criterion still failed on four stress cases, as shown in Table 4, involving repeated component chains with large false net merges, long meandering traces with changing width, circular distractors near genuine pads and two-sided transistor layouts with dark silkscreen. The failure set is unchanged across the six reported configurations and should be interpreted as a limitation of the recognition stage, not as a schematic export failure.

Table 4. Default sequential reconstruction results. Abbreviations: Pad R/P—pad recall/precision; Pair R/P—TOP/BOTTOM pair recall/precision; Δ Nets—difference between reconstructed and reference net count; NetSig R/P—net-size signature recall/precision; Comp.—recognized components over reference components; Prob.—open diagnostic problems.

Board	Pad R	Pad P	Pair R	Pair P	Δ Nets	NetSig R	NetSig P	Comp.	Prob.	Status
Cascade	1.00	0.90	1.00	0.92	+11	0.18	0.09	19/19	2	fail
Connector	0.77	1.00	0.77	1.00	−1	0.25	0.33	3/6	0	pass
Crossover	1.00	1.00	1.00	1.00	0	1.00	1.00	5/6	0	pass
Divider	1.00	1.00	1.00	1.00	+1	0.67	0.50	3/4	0	pass
Elements	0.85	1.00	0.83	1.00	−2	0.45	0.56	8/20	2	pass
Filter	0.94	1.00	0.94	1.00	+2	0.71	0.56	4/7	0	pass
Floating	1.00	0.89	1.00	1.00	+1	0.50	0.40	4/4	1	pass
Groundplane	0.75	1.00	0.75	1.00	+1	0.67	0.50	1/4	0	pass
Meander	1.00	1.00	1.00	1.00	+4	0.00	0.00	3/3	0	fail
Round	1.00	0.57	1.00	0.57	+9	0.00	0.00	1/4	1	fail
Simple	1.00	1.00	1.00	1.00	0	1.00	1.00	1/1	0	pass
Transistors	1.00	0.70	1.00	0.70	+6	0.50	0.25	4/5	0	fail

The failed boards also illustrate different error sources and improvement paths:

- *Cascade*: repeated sections encourage a large false merged net and wrong pin grouping; stronger component-hypothesis constraints, repeated-pattern consistency checks and conservative false-short penalties address this failure mode.
- *Meander*: the long variable-width trace is fragmented or grouped incorrectly; this points to region-adjacency trace modeling and safer missing-edge activation guarded by component consistency.
- *Round*: circular distractors reduce pad precision and create additional candidate nets; stronger false-pad rejection can use local copper continuity, silkscreen context and paired-side evidence.
- *Transistors*: two-sided component interpretation and ambiguous pad candidates expose component grouping and pin-order weaknesses; this motivates constrained component templates, side-aware pin ordering and stronger OCR/reference-designator validation.

These cases motivate the later separation between pad detection, component inference, contact reasoning and schematic export.

At the pin-relation level, the default run produced 4 true positive relations, 402 false positive relations and 240 missing relations, giving NED = 642. This metric is intentionally strict. It penalizes a plausible-looking generated schematic when the component pins are connected to the wrong reconstructed nets.

To make the practical impact more interpretable, an additional estimate was computed for the number of manual netlist-correction operations that would be needed to transform the generated netlist into the reference netlist. It is derived from pin-to-net assignments. A generated net that contains pins belonging to several reference nets contributes split operations. A reference net whose pins are fragmented across several generated nets contributes join operations. Missing or extra component-pin identifiers contribute pin-identity operations, because they require adding, removing, renaming or remapping a component pin before the netlist can match the reference. The estimated manual netlist corrections (EMNC) are the sums of these three terms as presented in Table 5.

The default reconstruction therefore corresponds to 210 estimated group-level manual netlist corrections, substantially fewer than the 642 strict pin-pair edits. The difference is expected: one false merged net can create many wrong pin pairs, but an operator would normally repair it by splitting a generated net group rather than by editing every induced pair independently.

Table 5. Estimated manual netlist-correction effort for the default sequential reconstruction. Split—generated net groups that must be separated. Join—reference net groups that must be reconnected. Pin ID—missing or extra component-pin identifiers. The most relevant EMNC (presented in bold) is the sum of these estimated operations. NED is the strict pin-pair edit distance.

Board	Split	Join	Pin ID	EMNC	NED
Cascade	8	6	59	73	446
Connector	3	2	6	11	20
Crossover	2	2	6	10	16
Divider	0	1	6	7	9
Elements	5	6	32	43	77
Filter	0	0	25	25	20
Floating	2	2	4	8	12
Groundplane	0	0	6	6	9
Meander	0	4	0	4	6
Round	0	0	6	6	4
Simple	0	0	0	0	0
Transistors	3	3	11	17	23
Total	23	26	161	210	642

A second, KiCad-based electrical comparison was added to separate export validity from reconstruction correctness. KiCad XML netlists were exported from each reference schematic and from each generated schematic. Power symbols and nets with fewer than two non-power component pins were ignored, because they do not define component-to-component connectivity. The comparison reports three percentages that give partial information instead of reducing every imperfect net to zero.

For a reference net R and generated net G , the partial overlap is defined as:

$$J(R, G) = \frac{|R \cap G|}{|R \cup G|}. \quad (17)$$

For each reference net, the best generated overlap is averaged with weights proportional to the reference net size. The same operation is then performed in the reverse direction for generated nets. *NetOverlap* is the harmonic mean of these two weighted averages. This gives partial credit for a net that shares some correct pins with the reference, while still penalizing false extra pins and spurious generated nets.

For pin relations, the Table 6 reports *PairF1* rather than a Jaccard-style agreement:

$$A_{pair,F1} = \frac{2P_{match}}{2P_{match} + P_{missing} + P_{extra}}. \quad (18)$$

Finally, *SizeF1* compares only the multiset of net sizes. It ignores component references and pin names, so it cannot prove electrical correctness, but it helps distinguish a wrong label or component assignment from a more basic failure to reconstruct the approximate net topology. The empty *Simple* comparison is reported as 100% by convention because neither schematic contains a multi-pin non-power component net under this definition.

The resulting values are more informative than a single all-or-nothing net identity score. *NetOverlap* reaches 15.9%, showing that some generated nets partially overlap the reference connectivity. The anonymous *SizeF1* score reaches 41.0%, indicating that part of the net-size structure is preserved even when component references, pin assignments or trace contacts are wrong. The low *PairF1* score of 1.2% shows that exact pin-to-pin electrical identity remains poor. These results confirm the interpretation used throughout the paper: the current exporter reliably preserves the program's own reconstructed netlist, while the recognition layer still fails to recover the reference electrical connectivity on several stress fixtures.

Table 6. Electrical agreement between KiCad XML netlists exported from the reference and generated schematics. NetRef/NetGen count multi-pin non-power component nets. NetOverlap gives partial credit for best matching net groups. PairF1 compares exact pin pairs. SizeF1 compares only anonymous net sizes.

Board	NetRef	NetGen	NetOverlap [%]	PairF1 [%]	SizeF1 [%]
Cascade	11	3	8.0	0.4	28.6
Connector	4	3	29.6	9.1	57.1
Crossover	4	4	40.9	20.0	50.0
Divider	3	2	22.0	0.0	40.0
Elements	10	5	13.9	0.0	26.7
Filter	7	4	9.8	0.0	72.7
Floating	3	2	23.4	0.0	40.0
Groundplane	2	0	0.0	0.0	0.0
Meander	2	0	0.0	0.0	0.0
Round	4	0	0.0	0.0	0.0
Simple	0	0	100.0	100.0	100.0
Transistors	6	4	22.1	0.0	60.0
Total	56	27	15.9	1.2	41.0

6.3. Runtime Ablations and Solver Comparisons

The global component solver, the risk-averse contact solver and the probabilistic contact solver were measured as separate opt-in experiments. Two additional runtime ablations disabled OCR evidence and ground-plane handling. These experiments were generated by rerunning the full pipeline on the same twelve fixtures. Only switches exposed by the current implementation were ablated; via pairing and Hough-only pad detection are therefore analyzed through the diagnostic decomposition rather than by changing private internal functions.

In Table 7, NED is the main aggregate electrical-error measure for comparing solver variants. RT confirms export consistency, Pin F1 gives strict reference agreement and EMNC estimates correction effort. The default configuration is selected by balancing aggregate NED against false-short and component-selection risks.

Table 7. Aggregate runtime ablations. RT—round-trip validation. Pad F1 and Pair F1 evaluate object and TOP/BOTTOM pairing quality. Pin F1 and NED evaluate reference pin-to-pin electrical agreement. EMNC is the estimated manual netlist-correction effort. NetSize F1 compares only anonymous net-size signatures.

Configuration	RT	Pad F1 [%]	Pair F1 [%]	Pin F1 [%]	NED	EMNC	NetSize F1 [%]	Open
Baseline	12/12	91.9	92.1	1.2	642	210	37.7	5
Global component solver	12/12	91.9	92.1	3.1	313	182	37.7	5
Risk-averse contact solver	12/12	91.9	92.1	1.5	525	207	37.2	5
Probabilistic contact solver	12/12	91.9	92.1	1.5	525	207	40.3	5
No OCR evidence	12/12	91.9	92.1	1.2	642	210	37.7	5
No ground-plane handling	12/12	91.9	92.1	1.2	642	210	37.7	5

The global component solver strongly reduced aggregate false positive pin relations and NED, most visibly on the repeated-chain example with severe false merges. It improved Pin F1 by 1.9 percentage points, reduced NED by 329 and reduced EMNC from 210 to 182, but it changes selected component hypotheses and is not enabled by default. The result is useful as a benchmark for candidate scoring and constraints, not as the final production configuration. This is why the default pipeline remains the sequential recognizer even though the global solver obtains a lower aggregate NED and EMNC in this synthetic run.

The two contact solvers reduced NED by 117 while preserving all export checks. The risk-averse mode deactivates risky accepted contacts, whereas the probabilistic mode

can additionally activate a small number of local pad-trace proximity hypotheses. In this benchmark, both modes produced the same aggregate Pin F1 and reduced EMNC only slightly, from 210 to 207, while the probabilistic mode preserved a slightly better anonymous net-size signature. The improvement is real but limited, and both modes remain opt-in because false-short risk is more serious than a missing tentative edge.

The OCR and ground-plane ablations are negative controls for this dataset. Disabling OCR evidence did not change the aggregate results, which indicates that the current synthetic fixtures are dominated by geometry, silkscreen and net-context evidence rather than by successfully recognized text. Disabling ground-plane handling also produced no aggregate change in NED or EMNC, showing that these fixtures do not yet stress plane-contact reasoning strongly enough.

As shown in Table 8, the decomposition confirms that schematic export is not the limiting stage in the present benchmark. No routing or round-trip export errors were attributed to the schematic layer. Most diagnostic errors are instead concentrated in trace-contact reasoning and pin assignment, followed by component grouping, pad/pair detection and false pad rejection. This explains why the exported schematics are structurally valid KiCad files while their reference electrical agreement remains low.

Table 8. Baseline error decomposition by diagnostic module. Evidence is the number of objects, candidate relations or exported symbols inspected by the diagnostic layer. Errors are diagnostic proxy counts and are not mutually exclusive; one physical defect can contribute to more than one module.

Module	Evidence	Errors	Boards
Pad detection	183	30	8
False pad rejection	189	18	4
TOP/BOTTOM pairing	183	30	7
Component grouping	83	69	10
Pin order and assignment	406	436	9
Trace-contact reasoning	438	683	12
Plane and large-net handling	0	2	2
Schematic export and routing	56	0	0

The probabilistic change presented in Table 9 is most informative on boards dominated by circular pads and short fragmented traces. There, the contact solver reduced the net-count delta from +9 to +7 and reduced two diagnostic symptoms: missing trace contacts and groups of single physical pads. It did so without increasing false positive pin relations. The change was still insufficient to make the board pass the quality thresholds.

Table 9. Most relevant changes caused by the probabilistic contact solver. The most informative probabilistic changes are marked with bold fonts.

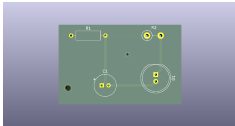
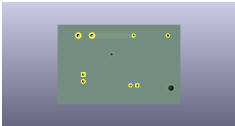
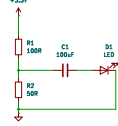
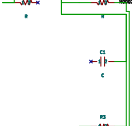
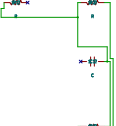
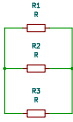
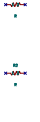
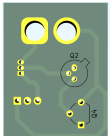
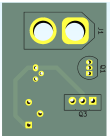
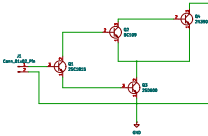
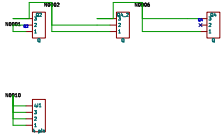
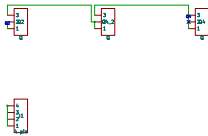
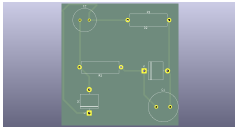
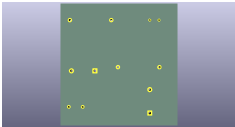
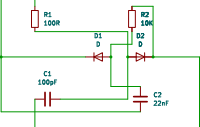
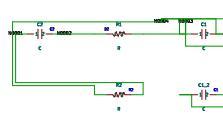
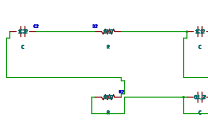
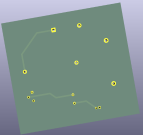
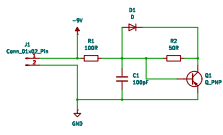
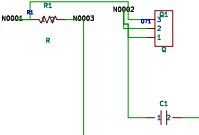
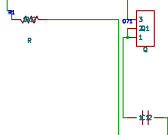
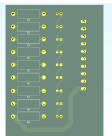
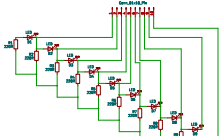
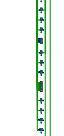
Board	FP Default	FP Probabilistic	NED Default	NED Probabilistic
Cascade	352	235	446	329
Connector	4	4	20	20
Round	0	0	4	4
Transistors	13	13	23	23

One attempted extension was deliberately rejected. Activating a local gap-bridging contact candidate improved the net-count delta on the board with a long variable-width meandering trace from +4 to +1, but it removed all recognized components on that board (3 to 0). The candidate type was therefore left as observe-only diagnostic evidence. This failed attempt is important because it shows why a single net metric must not be optimized without protecting component consistency.

6.4. Visual Summary

Table 10 shows selected examples rather than all twelve boards. The chosen subset covers sparse isolated nets, long meandering traces, two-sided transistor layouts, a simple crossover, a connector-like layout and a repeated chain with severe false merges. The TOP and BOTTOM images are raster board renders, whereas the three schematic columns are vector drawings exported from KiCad schematics and cropped to the schematic content. The *Automatic* column shows the direct default reconstruction. The *Optimized* column shows the same automatic netlist after readability layout and rerouting, not after manual correction. Some paired TOP and BOTTOM renders intentionally differ in scale, crop or rotation; these fixture variations isolate alignment-sensitive failure modes rather than representing a fully registered acquisition setup.

Table 10. Selected examples of board views and generated schematics.

Board	TOP	BOTTOM	Reference	Automatic	Optimized
Floating					
Meander					
Transistors					
Crossover					
Connector					
Cascade					

6.5. Exploratory Trial on Real Photographs

Although the quantitative benchmark was synthetic, the current prototype was also run on two simple photographed PCB modules. This two-board trial is an exploratory

feasibility check rather than a quantitative real-photo validation set. It shows how the same pipeline behaves when the input contains non-uniform lighting, specular pad reflections, weak copper–soldermask contrast and partial transparency of the board material.

The result shown in Figure 7 illustrates that the current implementation is not yet ready for reliable fully automatic analysis of arbitrary photographed PCBs. It can still find many circular pad or through-hole candidates on these simple boards, but trace extraction is much less stable. On real boards, conductive traces are often substantially weaker than in the synthetic renders, change appearance with illumination angle and may partly show through the substrate or soldermask. Uneven lighting then causes the trace mask either to disappear in low-contrast regions or to include broad regions of the board background. These failures directly affect contact reasoning and make the generated netlist unsuitable as an electrical validation result.

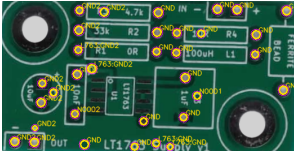
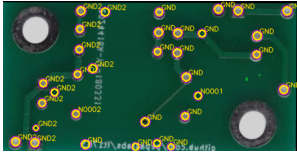
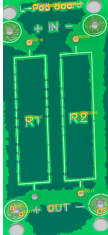
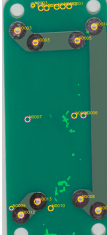
Board	TOP photo	TOP analysis	BOTTOM photo	BOTTOM analysis
Supply module				
L-pad module				

Figure 7. Exploratory qualitative runs on two real photographed PCB modules. The analysis previews show detected pad or hole candidates and trace-mask evidence overlaid on the normalized board views; the figure is not a quantitative real-photo benchmark.

This exploratory trial therefore supports the staged evaluation strategy used in this paper. At this point, synthetic boards are the better benchmark for verifying the foundations of the pipeline because they provide known reference schematics and isolate specific algorithmic failure modes. Real photographed boards remain the intended next evaluation stage, but they require a dedicated acquisition protocol and stronger trace evidence before robust quantitative claims can be made.

7. Discussion

The final results separate two aspects of the system. The schematic export layer is stable on the synthetic benchmark: every generated schematic passes KiCad round-trip validation, uses drawn wires and avoids label-only net representation. The automatic optimization step also produces a cleaner schematic view for each evaluated board. The recognition layer remains the harder part of the problem. Four fixtures still fail the quality thresholds, and the strict pin-to-pin relation metric remains the most demanding measure. The practical interpretation is therefore that the system produces an editable draft and a structured diagnostic record for human verification, not a certified autonomous reconstruction.

This distinction is important for printed circuit board (PCB) reverse engineering. Optical photographs cannot reveal all electrical evidence. Traces may be hidden under components, internal layers are invisible, solder-mask contrast can be weak and copper pours can connect regions that are difficult to segment reliably. Missing a connection is usually visible as an

isolated pad or incomplete component. A false merge is more problematic because it can create a plausible but incorrect net. For this reason, the default pipeline is kept conservative and the higher-risk solvers are exposed as opt-in diagnostic modes.

The synthetic fixtures are deliberately controlled. They do not replace a dedicated real-photograph benchmark, but they are useful at the current stage because they isolate algorithmic failures that would be harder to diagnose in photographic noise. The exploratory real-board trial illustrates the main limitation of the current real-photo path: the system can still detect many pads or holes on simple photographs, while trace masks degrade under uneven illumination, weak copper contrast and partial transparency of the board material. The most difficult synthetic cases combine long meandering traces with changing width, circular structures that compete with genuine pads, two-sided component placement under dark silkscreen and repeated sections that encourage large false net merges. These cases explain why incremental threshold tuning is no longer the main path forward.

The current evaluation also estimates the amount of netlist correction that would be required before the generated schematic matches the reference design. For the default pipeline, the estimated manual netlist-correction effort is 210 group-level operations, compared with 642 strict pin-pair edits. This difference is important because a single large false merge can induce many incorrect pin pairs but can often be inspected as one net-splitting problem. The manually corrected examples demonstrate that the graph and schematic can be recalculated after user edits, but they are still not a controlled user study and should not be read as a measured correction-time benchmark. A full practical evaluation should compare the measured editing effort with manual reconstruction from an empty EDA sheet.

The contact-solver ablations illustrate both the value and the risk of correction driven by evidence. Deactivating selected high-risk contacts reduces the number of false positive pin relations on the repeated-chain examples and lowers aggregate NED from 642 to 525. Limited activation of missing-edge hypotheses improves the anonymous net-size signature without breaking the export checks. Concurrently, the rejected `pad_gap_bridge` experiment on the long meandering-trace example shows that a net metric can improve while the component model degrades. A production solver must therefore optimize graph connectivity and component consistency together, rather than treating missing-edge activation as a local image-processing decision.

The global component solver provides a complementary result. It reduces NED from 642 to 313, improves Pin F1 from 1.2% to 3.1% and lowers the estimated manual netlist-correction effort from 210 to 182, especially when the board contains repeated sections or visually ambiguous transistor groupings. However, it changes component selection decisions and depends on a candidate pool that is still noisy. The result is valuable as an offline benchmark for scoring and constraints, but it does not yet justify replacing the sequential component recognizer in the default workflow.

The diagnostics added before the final freeze are useful for future work. The program records physical pad nodes behind exported symbol pins, the source of component decisions, candidate contact edges, rejected risky edges and pin-relation errors. These data make failures reproducible and help distinguish between pad detection, component grouping, pin assignment and trace-contact errors. In the present manuscript this decomposition is summarized quantitatively in the baseline error table and used qualitatively in the board failure analysis. They also create a path toward learning from manual corrections: similar pad classes, repeated OCR patterns and recurring false contacts can be found and corrected in batches.

The method should therefore be positioned as a human-in-the-loop assistant. It reduces repetitive manual tracing by preparing normalized views, trace masks, detected pads, paired through-holes, candidate components, an explicit graph, a schematic draft,

a readability-optimized view and diagnostics. The operator reviews a structured draft rather than starting from an empty sheet, especially on dense boards or boards with ambiguous OCR evidence. In the intended workflow, the software prepares evidence and draft schematics, while the operator verifies and corrects electrical connectivity.

8. Future Development

The project can be developed in several complementary directions. The first direction is a better PCB object recognition. The current pipeline uses classical image processing, color sampling and explicit shape filters. This approach is transparent and easy to correct, but it is still sensitive to illumination, soldermask color, weak trace contrast and dense local geometry. The current non-green fixes improved several low-contrast passive boards, but the method should be expanded into a controlled real-photograph benchmark with more soldermask colors, uneven lighting conditions and measured electrical references.

Future evaluation should extend the present runtime ablations with deeper instrumentation of stages that are not yet independently switchable, especially via pairing, Hough-only pad detection, relaxed trace contacts and schematic optimization. The error analysis should also be repeated on a larger set of photographed boards, because pad detection, component grouping, pin order and contact reasoning fail in different ways under acquisition noise. Claims about robustness on real boards require a new dataset of photographed PCBs with reliable electrical references.

The second direction is related to the component recognition. This is currently the most important research topic after pad detection, but the latest results suggest that another small local heuristic is unlikely to be enough. The OCR results, silkscreen outlines, footprint geometry, pad spacing, polarity marks and local net context should be converted into explicit component hypotheses and selected by a global solver. Such a solver could treat the board as a constrained matching or set-packing problem: pads should normally belong to one component, references should not be duplicated, common footprints should have plausible pin counts and high-risk shorts should be penalized. Future reports should separate the following questions: whether the physical pads were found, whether they were grouped into the correct component, whether the component type was correct and whether the pin order was correct. These are different problems and should not be hidden behind one component-count metric.

The third direction is netlist correctness. The current net-size signature is useful for early regression testing, but pin-to-pin relation metrics are more informative. Future versions should show which graph edges created each false positive relation and which missing relation is blocked by missing trace contact, missing component grouping or wrong pin order. For trace recognition, the next non-incremental alternative is a region-adjacency or superpixel graph: image regions would be scored as pad metal, trace metal, soldermask, silkscreen or shadow, and electrical edges would be created from region contact rather than from one global mask. Large trace components and relaxed non-green pad contacts should be explicitly marked as risk evidence in the editor.

The fourth direction is schematic readability. The exporter now creates both a base schematic and an optimized schematic, but the autorouter is still a practical heuristic rather than a complete schematic-layout engine. Future layout optimization should better separate functional blocks, reduce wire crossings, prefer readable buses or net labels for global signals, keep related components close to each other and preserve useful manual edits. Port-aware layout algorithms, orthogonal hyperedge routing and rip-up-and-reroute strategies are promising because schematic pins are fixed connection ports and because one net should often be drawn as a shared tree rather than as many independent pairwise connections.

The fifth direction is learning from human corrections. When the user changes one repeated object, for example marks one group of circular detections as mounting holes or

corrects an OCR pattern, the system should search for similar cases on the same board and propose the same correction. Over time, these confirmed corrections can form a training set for segmentation, component detection and trace-contact classifiers. Before adding a machine-learning model, the project should first create a dataset-first benchmark with separate targets for pad detection, pad-to-component assignment, type recognition, pin order, trace contact and net grouping. The current synthetic fixtures should remain part of this benchmark because they isolate basic failures before real-photograph noise is introduced.

The most essential practical tasks are therefore:

- Prototype a global component-hypothesis solver on boards with dense passive sections, repeated chains, copper pours and visually ambiguous transistor groupings while keeping the current baseline as the default;
- Prototype a region-adjacency trace model for soldermask-independent contact evidence, especially for variable-width meandering traces and boards with circular distractors near true pads;
- Reject false pad candidates on non-standard colors and black silkscreen without hardcoding the current fixture colors;
- Reduce false and missing pin-to-pin relations;
- Add a component-safety model before activating more `pad_gap_bridge` or missing-edge hypotheses;
- Expose risky graph evidence directly in the generated diagnostics and editor;
- Collect corrected masks, components, pin orders and net contacts as benchmark data;
- Improve schematic autorouting while preserving the rule that non-trivial nets must be represented by wires, not only by pin labels.

9. Conclusions

This paper presented a semi-automatic pipeline for preparing an editable KiCad schematic draft from paired TOP and BOTTOM PCB images. The system combines image normalization, color-based trace evidence, pad and through-hole detection, OCR-supported component grouping, explicit graph construction, netlist extraction, schematic export with routed wires and a separate optimized schematic generated by grid placement and orthogonal autorouting.

The primary quantitative evaluation on twelve synthetic KiCad fixtures shows that the developed program can complete the end-to-end workflow without runtime errors and can produce internally consistent KiCad schematics that pass round-trip validation. This result is limited to the controlled synthetic benchmark. The additional qualitative trial on two photographed boards indicates that the current implementation can detect many holes and pads on simple real modules, but trace extraction is not yet robust to uneven lighting and weak copper–soldermask contrast. For all six reported solver and ablation configurations, direct and optimized schematic views were generated for all synthetic boards. This is a meaningful engineering result because the exported schematic is not a label-only sketch: the generated wires preserve the program’s internal reconstruction and can be edited in a standard EDA tool. The round-trip result, however, validates export consistency rather than electrical correctness against the reference design.

The same evaluation also shows that the recognition problem is not fully solved. The default pipeline still fails the reconstruction-quality thresholds on stress cases involving repeated component chains, long variable-width meandering traces, circular distractors near pads and two-sided transistor layouts. The strict pin-to-pin relation metric remains the most demanding indicator of electrical correctness. The default generated netlists require an estimated 210 group-level manual netlist corrections to match the reference netlists, whereas the strict pin-pair edit distance is 642. These failures are concentrated

in component grouping, trace-contact interpretation, rejection of false pad candidates and protection against large false net merges.

The contact solvers provide useful but limited opt-in improvements. They reduce the aggregate netlist edit distance from 642 to 525 and keep all schematic export checks passing, but they reduce the estimated manual correction effort only slightly, from 210 to 207, and do not make the benchmark pass 12/12 quality checks. The global component solver reduces the aggregate false positive count more strongly, lowers NED to 313 and lowers the estimated manual correction effort to 182, but it also remains an opt-in diagnostic mode because component selection must be protected against regressions. The no-OCR and no-ground-plane ablations do not change the aggregate result on the current synthetic dataset. A rejected pad_gap_bridge variant showed that improving net fragmentation alone can damage component recognition. For that reason, both solver families remain diagnostic modes rather than default program behavior.

The method is useful as a human-in-the-loop reverse-engineering assistant. It prepares the operator's workspace, generates a structured schematic draft and exposes diagnostic evidence for correction. The resulting draft provides an editable and diagnosable starting point for expert verification on the evaluated synthetic fixtures, while real-photo inputs still require more robust trace extraction. Future work should focus on constrained component hypotheses, safer missing-edge activation, region-adjacency trace modeling, deeper stage-level ablations, learning from manual corrections and a larger real-photograph benchmark developed alongside the synthetic stress cases.

Author Contributions: Conceptualization, K.M. and K.O.; methodology, K.M.; software, K.M.; validation, K.M.; formal analysis, K.M.; investigation, K.M.; writing—original draft preparation, K.M.; writing—review and editing, K.M. and K.O.; visualization, K.M.; supervision, K.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The program release is publicly available on Zenodo [33]. The synthetic KiCad fixtures, input images, generated measurements, evaluation materials and generated KiCad schematics are publicly available as a separate Zenodo dataset [34].

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CIELab	Commission Internationale de l'Eclairage color space
CLAHE	Contrast Limited Adaptive Histogram Equalization
CT	Computed Tomography
EDA	Electronic Design Automation
ERC	Electrical Rules Check
HSV	Hue Saturation Value color space
NED	Netlist Edit Distance
OCR	Optical Character Recognition
PCB	Printed Circuit Board
SMD	Surface-Mount Device
THT	Through-Hole Technology
VLSI	Very Large Scale Integration

References

1. Craig, P.; Varshney, N.; Roy, A.; Ghosh, S.; Patil, C.; Dalir, H.; Asadizanjani, N. Multi-Modal Printed Circuit Board Netlist Extraction with X-ray and Optical Imaging. In *Developments in X-Ray Tomography XV*; Proceedings of SPIE; Müller, B., Wang, G., Eds.; SPIE: Bellingham, WA, USA, 2024; Volume 13152, p. 131520Q. [\[CrossRef\]](#)
2. Huang, C.Y.; Chen, H.I.; Ho, H.W.; Kang, P.H.; Lin, M.P.H.; Liu, W.H.; Ren, H. Netlistify: Transforming Circuit Schematics into Netlists with Deep Learning. In Proceedings of the 2025 ACM/IEEE 7th Symposium on Machine Learning for CAD (MLCAD), Santa Cruz, CA, USA, 8–10 September 2025; pp. 1–8. [\[CrossRef\]](#)
3. Arbash, E.; Fuchs, M.; Rasti, B.; Lorenz, S.; Ghamisi, P.; Gloaguen, R. PCB-Vision: A Multiscene RGB-Hyperspectral Benchmark Dataset of Printed Circuit Boards. *IEEE Sens. J.* **2024**, *24*, 17140–17158. [\[CrossRef\]](#)
4. Zhao, W.; Gurudu, S.; Taheri, S.; Ghosh, S.; Sathiaselvan, M.A.M.; Asadizanjani, N. PCB Component Detection Using Computer Vision for Hardware Assurance. *Big Data Cogn. Comput.* **2022**, *6*, 39. [\[CrossRef\]](#)
5. Phoulady, A.; Suleiman, Y.; Choi, H.; May, N.; Shahbazmohamadi, S.; Tavousi, P. Automated 3D Semantic Segmentation of PCB X-ray CT Images and Netlist Extraction. *Sci. Rep.* **2025**, *15*, 2230. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Hemker, D.; Maaloulou, J.; Mathis, H.; Klos, R.; Ravanian, E. From Schematics to Netlists – Electrical Circuit Analysis Using Deep-Learning Methods. *Adv. Radio Sci.* **2024**, *22*, 61–75. [\[CrossRef\]](#)
7. Hu, W.; Zhan, X.; Tong, M. Parsing Netlists of Integrated Circuits from Images via Graph Attention Network. *Sensors* **2024**, *24*, 227. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Fuchs, L.; Diesse, M.; Weber, M.; Rasim, A.; Feinauer, J.; Schmidt, V. Using Convolutional Neural Networks and Pattern Matching for Digitization of Printed Circuit Diagrams. *Electronics* **2025**, *14*, 2889. [\[CrossRef\]](#)
9. Xu, H.; Liu, C.; Wang, Q.; Huang, W.; Xu, Y.; Chen, W.; Peng, A.; Li, Z.; Li, B.; Qi, L.; et al. Image2Net: Datasets, Benchmark and Hybrid Framework to Convert Analog Circuit Diagrams into Netlists. In Proceedings of the 2025 International Symposium of Electronics Design Automation (ISED), Hong Kong, China, 9–12 May 2025; pp. 807–816. [\[CrossRef\]](#)
10. Aldowais, S.; Karumanchi, Y.; Chiang, K.C.; Noorzad, S.; Fayazi, M. SINA: A Circuit Schematic Image-to-Netlist Generator Using Artificial Intelligence. In Proceedings of the 2026 Design, Automation & Test in Europe Conference (DATE), Verona, Italy, 20–22 April 2026; pp. 1–3. [\[CrossRef\]](#)
11. Matsuo, R.; Uhlich, S.; Venkitaraman, A.; Bonetti, A.; Hsieh, C.Y.; Momeni, A.; Mauch, L.; Capone, A.; Ohbuchi, E.; Servadei, L. Schemato—An LLM for Netlist-to-Schematic Conversion. In Proceedings of the 2025 ACM/IEEE 7th Symposium on Machine Learning for CAD (MLCAD), Santa Cruz, CA, USA, 8–10 September 2025; pp. 1–7. [\[CrossRef\]](#)
12. Domrös, S.; von Hanxleden, R.; Spönemann, M.; Rüegg, U.; Schulze, C.D. The Eclipse Layout Kernel. *arXiv* **2023**, arXiv:2311.00533. [\[CrossRef\]](#)
13. Spönemann, M.; Schulze, C.D.; Rüegg, U.; von Hanxleden, R. Drawing Layered Graphs with Port Constraints. *J. Vis. Lang. Comput.* **2014**, *25*, 89–106. [\[CrossRef\]](#)
14. Lee, C.Y. An Algorithm for Path Connections and Its Applications. *IRE Trans. Electron. Comput.* **1961**, *EC-10*, 346–365. [\[CrossRef\]](#)
15. Tamassia, R. On Embedding a Graph in the Grid with the Minimum Number of Bends. *SIAM J. Comput.* **1987**, *16*, 421–444. [\[CrossRef\]](#)
16. Wybrow, M.; Marriott, K.; Stuckey, P.J. Orthogonal Hyperedge Routing. In *Diagrammatic Representation and Inference. 7th International Conference, Diagrams 2012, Canterbury, UK, 2–6 July 2012*; Cox, P., Plimmer, B., Rodgers, P., Eds.; Lecture Notes in Artificial Intelligence—LNAI; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7352, pp. 51–64. [\[CrossRef\]](#)
17. Helmke, S.; Goetze, B.; Scheffler, R.; Wrobel, G. Interactive, Orthogonal Hyperedge Routing in Schematic Diagrams Assisted by Layout Automatism. In *Diagrammatic Representation and Inference. 12th International Conference, Diagrams 2021, 28–30 September 2021*; Basu, A., Stapleton, G., Linker, S., Legg, C., Manalo, E., Viana, P., Eds.; Lecture Notes in Artificial Intelligence—LNAI; Springer: Cham, Switzerland, 2021; Volume 12909, pp. 21–36. [\[CrossRef\]](#)
18. KiCad. A Cross Platform and Open Source PCB Design Suite. Available online: <https://www.kicad.org/> (accessed on 31 May 2026).
19. EasyEDA. Easy-to-Use & Free PCB Design Software. Available online: <https://easyeda.com/> (accessed on 31 May 2026).
20. Altium Designer. The Industry’s Leading PCB Design Software. Available online: <https://www.altium.com/altium-designer/> (accessed on 31 May 2026).
21. Cadence Design Systems. OrCAD and Allegro PCB Design. Available online: https://www.cadence.com/en_US/home/tools/pcb-design-and-analysis.html (accessed on 31 May 2026).
22. KiCad Command-Line Interface Documentation. Available online: <https://docs.kicad.org/master/en/cli/cli.html> (accessed on 2 June 2026).
23. Global Electronics Association. IPC-2221B: Generic Standard on Printed Board Design. 2012. Available online: <https://www.ipc.org/TOC/IPC-2221B.pdf> (accessed on 2 June 2026).
24. IPC-7351B; Generic Requirements for Surface Mount Design and Land Pattern Standard. Global Electronics Association: Bannockburn, IL, USA, 2010. Available online: https://ohm.bu.edu/~pbohn/_Engineering_Reference/pcb_layout/pcbmatrix/IPC-7x51%20Padstack%20Naming%20Convention.pdf (accessed on 2 June 2026).

25. MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*; University of California Press: Oakland, CA, USA, 1967; Volume 1, pp. 281–297.
26. Duda, R.O.; Hart, P.E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Commun. ACM* **1972**, *15*, 11–15. [[CrossRef](#)]
27. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698. [[CrossRef](#)]
28. Pizer, S.M.; Amburn, E.P.; Austin, J.D.; Cromartie, R.; Geselowitz, A.; Greer, T.; ter Haar Romeny, B.; Zimmerman, J.B.; Zuiderveld, K. Adaptive Histogram Equalization and Its Variations. *Comput. Vision Graph. Image Process.* **1987**, *39*, 355–368. [[CrossRef](#)]
29. Niblack, W. *An Introduction to Digital Image Processing*; Prentice Hall: Englewood Cliffs, NJ, USA, 1986.
30. Sauvola, J.; Pietikäinen, M. Adaptive Document Image Binarization. *Pattern Recognit.* **2000**, *33*, 225–236. [[CrossRef](#)]
31. Smith, R. An Overview of the Tesseract OCR Engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Curitiba, Brazil, 23–26 September 2007; pp. 629–633. [[CrossRef](#)]
32. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [[CrossRef](#)]
33. Maliński, K. *PCB2SCH: Software Release*, version 0.1.1; Zenodo: Meyrin, Switzerland, 2026. [[CrossRef](#)]
34. Maliński, K. *PCB2SCH: Synthetic PCB Reconstruction Dataset*, version 0.1.0; Zenodo: Meyrin, Switzerland, 2026. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.