Computer-Aided Design 43 (2011) 278–292

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

From engineering diagrams to engineering models: Visual recognition and applications

Luoting Fu, Levent Burak Kara *

Mechanical Engineering Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, United States

ARTICLE INFO

Article history:
Received 13 August 2009
Accepted 25 December 2010

Keywords:
Network-like engineering diagram
Visual recognition
Convolutional Neural Network
Sketch-based modeling
Engineering document image analysis
Engineering information retrieval

ABSTRACT

We present a computational recognition approach to convert network-like, image-based engineering diagrams into engineering models with which computations of interests, such as CAD modeling, simulation, information retrieval and semantic-aware editing, are enabled. The proposed approach is designed to work on diagrams produced using computer-aided drawing tools or hand sketches, and does not rely on temporal information for recognition. Our approach leverages a Convolutional Neural Network (CNN) as a trainable engineering symbol recognizer. The CNN is capable of learning the visual features of the defined symbol categories from a few user-supplied prototypical diagrams and a set of synthetically generated training samples. When deployed, the trained CNN is applied either to the entire input diagram using a multi-scale sliding window or, where applicable, to each isolated pixel cluster obtained through Connected Component Analysis (CCA). Then the connectivity between the detected symbols are analyzed to obtain an attributed graph representing the engineering model conveyed by the diagram. We evaluate the performance of the approach with benchmark datasets and demonstrate its utility in different application scenarios, including the construction and simulation of control system or mechanical vibratory system models from hand-sketched or camera-captured images, content-based image retrieval for resonant circuits and sematic-aware image editing for floor plans.

1. Introduction

Diagrammatic, graphical representations are suitably utilized to prototype, visualize and communicate rich sets of engineering concepts, and as such play a key role in many engineering scenarios involving design analysis, synthesis, collaboration and education [1–9].

Human engineers, if presented with an engineering diagram, are able to recognize the underlying engineering model conveyed by the diagram, externalize it in a domain-specific computational environment and perform down-stream tasks of interest. Those are the typical steps of the design workflow revolving around diagrams.

In contrast, computers or software agents cannot directly accomplish the same without access to the underlying model. For this reason, some digital diagrams created with computational aids are saved in a software-specific format that retains both the diagrammatic appearance and, more importantly, the model information. However, many engineering diagrams are encoded in the widely available, application-neutral format of pixelated images. Apart from pixel intensities, no model information is stored in those images, necessitating *engineering diagram recognition*.

It is desirable to solve engineering diagram recognition, i.e., the automated conversion from image-based engineering diagrams to engineering models. The motivations are twofold. Firstly, image-based engineering diagrams exist in non-trivial quantities in diverse design and education related scenarios. Examples include engineering diagrams in patents, textbooks and other technical archives digitized with scanners, drawings distributed on the web and indexed by search engines, instructional diagrams drawn on the white boards or tablet PCs, and prototypical concepts sketched at the back of an envelope and then digitally captured by a camera.

Secondly, diagram recognition enables a wide spectrum of model-based, engineering computations on diagrammatic images, thus enhancing the supportive value of diagrams in design. Computations of interest include model creation, simulation, animation, indexing, retrieval and diagram editing operations. With that, the representational advantage of engineering diagrams is augmented by the computational support from engineering software.

In this paper, we tackle the recognition of a particular subset of engineering diagrams: the *network-like diagrams*. Within a network-like diagram, pixels can belong to either *symbols* or *connectors*, and symbols and connectors do not overlap or contain each other. This definition encompasses a wide range of domains

* Corresponding author. Tel.: +1 412 268 2509.
E-mail addresses: luoting.fu@cmu.edu (L. Fu), lkara@cmu.edu (L.B. Kara).

0010-4485/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2010.12.011

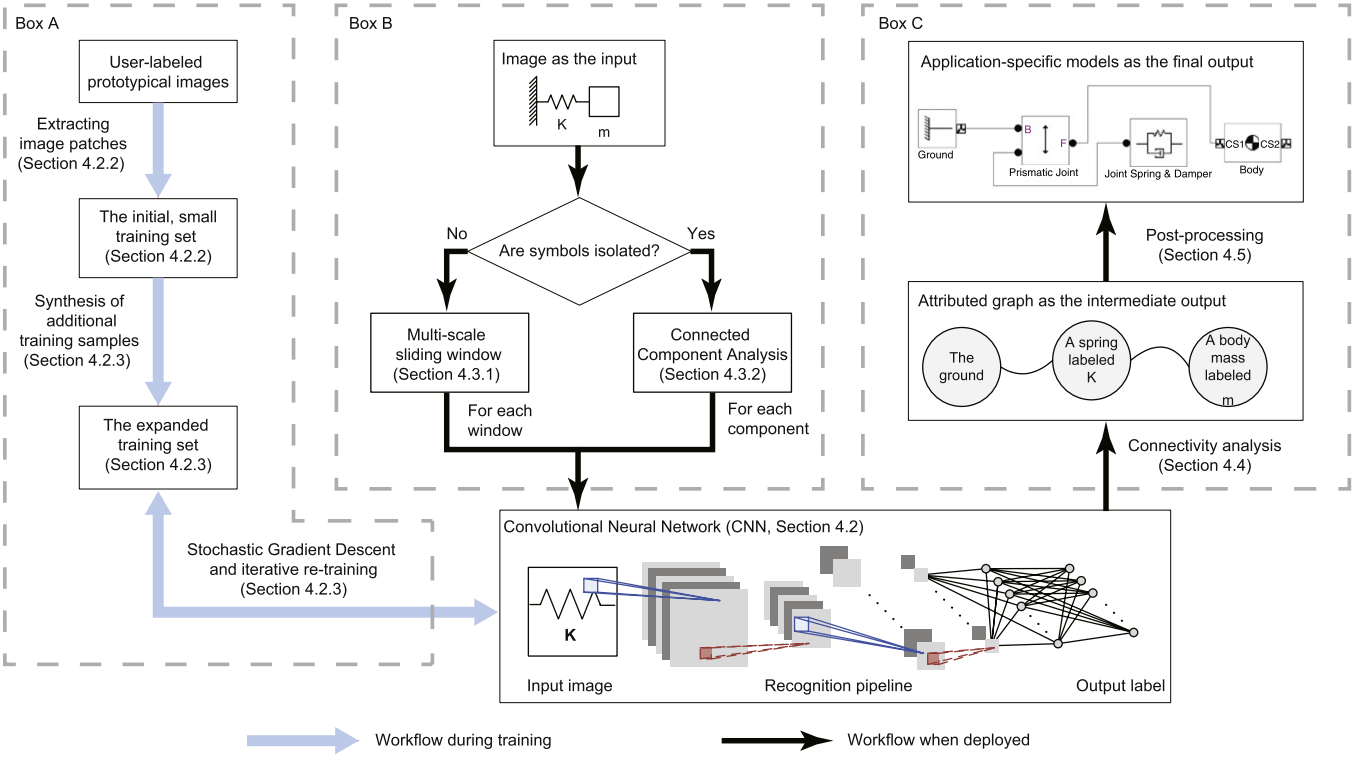


Fig. 1. Overview of the proposed approach.

in engineering, including multi-body vibration systems, control systems, bond graphs, electric circuits, algorithmic flowcharts, UML diagrams and state machines. In this work, we pursue a general approach to handle diagrams generated with the aid of drawing tools as well as diagrams sketched freehand.

The computational challenges herein is a lack of straightforward mapping from the pixel-based visual representation of a diagram to the underlying engineering model conveyed by the diagram. Specifically, the challenges include the need for a trainable image-based recognizer with desired invariance to a broad range of appearance variations, while providing interactive performance. Those challenges are detailed in Section 2.

Shown in Fig. 1, our visual recognition approach for network-like engineering diagrams consists of several modules. The core module is a Convolutional Neural Network (CNN) [10] symbol recognizer. The CNN is trained with labeled images prior to deployment, as illustrated in Box A of Fig. 1. The objective of training is for the CNN to learn the distinctive visual features of each symbol category. A large training dataset with several hundreds of examples per category is necessary to train a highly accurate CNN recognizer, but such a large dataset is usually costly and time-consuming to obtain. To avoid burdening the users with the collection many training samples, a small training set initially containing a few dozens of user-provided samples is iteratively and autonomously expanded, without requiring manual interventions from the user.

As Box B shows, the trained CNN is utilized with one of the following two localization modules to detect symbols within the input image. One module, based on the multi-scale sliding window, applies the CNN to the input image in an exhaustive manner. It achieves general applicability at the cost of speed and accuracy. The other module, based on Connected Component Analysis (CCA) [11], applies the CNN to the selected regions of the input image. It is applicable to images in which symbols are isolated at the pixel level, and produces improved accuracy and runtime than the sliding window.

Subsequently in Box C, the symbols detected by the CNN are subject to a connectivity analysis and then transformed into an attributed graph which is a domain-independent, intermediate representation. Domain-specific post-processing is then performed on the attributed graph to construct the engineering model represented by the input diagram. The modules of our approach are described in Section 4 and evaluated in Section 5. Applications built upon our approach are presented in Section 6.

1.1. Contributions

The contributions of our work are threefold. Firstly, it provides a generalized visual recognition approach applicable to multiple engineering domains within the defined scope of this paper, independent of how the diagrams are drawn (i.e., whether produced with computer-aided drawing tools, or sketched freehand) or acquired digitally (i.e., whether captured by webcams, scanners or screen shots, or drawn on a tablet PC). The domain independence is achieved because the recognizer is trained with symbols defined in the domain of interest and learns the distinguishing features of symbol categories. The input independence is achieved because the recognizer is built on visual patterns, rather than particular temporal or online patterns available only from tablet PCs.

In comparison, previous works in document image analysis have employed heuristics and/or thresholds valid only for neatly drafted diagrams or specific domains. Likewise, previous works in sketch understanding have relied predominantly on the stroke-level, temporal features of an input sketch and hence are limited to diagrams drawn with a digitizer on a tablet PC.

Secondly, the CNN recognizer yields a recognition performance competitive to existing recognition systems on published benchmark datasets. As the evaluation in Section 5 has shown, the CNN recognizer features invariance against man-made or device-induced appearance distortions that render symbol recognition difficult. Trained in a data-driven manner, it can obtain high accuracy as long as training data is provided. It is also free of the fixed

biases that are otherwise introduced by nonadaptive, hand-crafted recognition rules.

Thirdly, on the widely available, image-based engineering diagrams produced at various stages of design, our approach enables useful engineering computation. For example, the recognition of conceptualization diagrams at the early stages allows the designer to study the dynamic behaviors of the design concepts preliminarily depicted as static images, thus facilitating a quick exploration of design alternatives. The recognition of diagrams in design archives facilitates the mining, retrieval and reuse of past designs. The recognition of the diagrams being edited leads to human–computer interface that assists the users by simplifying complicated editing operations. Demonstrations of such scenarios are presented in Section 6.

2. Problem statement

2.1. Terminology

An engineering *diagram* refers to a two-dimensional symbolic representation of certain engineering information. The word *diagram* will be used interchangeably with the word *image* in the reminder of this paper, because we are focusing on diagrams represented as rasterized images.

A subset of all diagrams is the *network-like diagrams*, also known as *node-link diagrams*. It refers to diagrams in which pixels can belong to either *symbols* or *connectors*, and the symbols do not overlap with or contain each other. Symbols are well-defined pixel patterns corresponding to particular engineering objects. Connectors are undirected lines or directed arrows depicting the relationships between symbols.

2.2. Goal and scope

As stated in Section 1, the goal of this paper is to automate the recognition of an engineering model from its image-based, diagrammatical depiction. The input is an image-based diagram preprocessed for text removal and de-noising. The final output is the engineering model, constructed per the semantics of the input diagram, the engineering domain and the application of interest.

The scope of this paper is to recognize network-like diagrams independent of how they are produced, i.e., whether produced with computer-aided drawing tools, or sketched freehand, and independent of the means of acquisition, i.e., whether captured through webcams, scanners or screen shots, or through digital ink from tablet PCs.

2.3. Challenges

Kara et al. [12] summarized the two major computational tasks in sketched diagram recognition as *parsing* (also called *symbol localization*) and *symbol recognition*. During parsing, primitives of the input, be it pixels or pen strokes, are clustered together if they form a defined symbol. During the subsequent symbol recognition, the clusters of primitives are recognized as individual symbols.

For both tasks, different types of variations are the main challenges. Symbol localization is complicated by the global variations in terms of the symbols' locations and scales: the symbols to be located can be drawn at virtually any location and any scale within the input image. Symbol recognition is also rendered difficult because each defined category of symbols exhibits significant local visual variations despite the within-category similarities. Fig. 2 (a) exemplifies such variations: although a body mass is customarily defined as a rectangle, neither of the body mass symbols in this diagram features a perfectly rectangular shape; instead, they exhibit different shapes, scales, local distortions and even have extraneous, over-tracing strokes. Such rich variations preclude the use of trivial template-based algorithms such as pixel-based nearest neighbor. The same argument on variations also extends to diagrams formally drawn with drawing tools. The global location and scale variations still exist. And the local variations, though relatively limited, could still be induced by the differences in terms of shape definitions, aspect ratios and line widths. The acquisition of the diagram introduces additional variations. For example, image captured by a digital camera or a webcam are often subject to perspective and non-rectilinear distortions caused by the lens, shown in Fig. 2 (b), and occasionally the out-of-plane curvature of the drawing surface. To sum up, the first challenge for diagram recognition is the issue of variations. An ideal diagram recognizer should therefore feature *invariance* to symbols' locations, scales, local distortions, and missing or extraneous pixels.

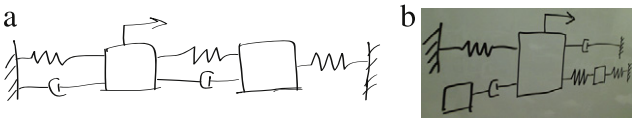


Fig. 2. Local visual variations are the major challenge for sketched symbol recognition. (a) Sketched diagram of a mechanical vibration system that exhibits high variability in drawing styles, scales, local distortions and over-tracing strokes. (b) A webcam snapshot of a rectilinear mechanical vibratory system that has undergone foreshortening due to the relative position between the webcam and the drawing board.

In the second place, *trainable* recognizers, rather than hand-crafted recognizers based on heuristics and thresholds, are necessary for two reasons. Firstly, the former can be retrained to improve upon the unobtrusive feedback that users provide by confirming or correcting recognition results. In contrast, the latter are subject to fixed biases and error rates, unless tuned by expert users. Secondly, the former can be generalized to a wide range of engineering domains with user-supplied training samples in the target domain. The latter require expert users to tune the internal parameters, if they can be tuned at all.

A third but non-trivial challenge for the recognizer is the *interactive performance*, especially when the recognizer is to be employed in sketch-based modeling interfaces. This narrows the candidate pool of recognition algorithms down to efficient classifiers, such as feed-forward neural networks, that do not perform expensive, optimization-based inference at runtime.

Based on the above challenges and desiderata, the invariance properties, trainability and processing speed of the recognizer constitute the major design objectives underlying many decisions of our approach. How well our approach addresses those objectives is shown throughout Sections 5 and 6, and summarized in Section 7.

3. Related work

3.1. Document image analysis

The capability for a computer to recognize diagrammatical inputs has been long sought-after. The document image analysis community has previously contributed different recognition solutions to network-like diagrams in several engineering domains. Earlier works focused exclusively on diagrams in a single domain or a particular input format [13,14]. For example, Lin et al. [15] and Yu et al. [16] used pre-defined, fixed rules and pattern templates for symbol localization and recognition. Futrelle et al. [17,18] proposed a diagram understanding system using Context-based Constraint Grammars and formulated symbol localization as a depth-first constraints satisfaction problem. Messmer and Bunke's system [19] formulated symbol recognition as an error-tolerant subgraph matching between symbols defined previously and symbols drawn in the input diagram. However, those approaches either relied heavily on manually tuned parameters that are specific to the application domain [15,19], or assumed vectorized

inputs [17,18,16] which are themselves non-trivial to obtain from pixelated input images.

Recent works have begun to shift the paradigm to the use of trainable classifiers for the task of recognizing localized symbols, achieving improved recognition accuracy. Pre-defined features such as image statistics are extracted from the preprocessed input and then used as inputs to a general purpose classifier such as the Naive Bayes classifier. Exemplary works include [20–27] and comprehensive reviews are offered in [28,29]. It is noteworthy that those symbol recognizers require the localization of each symbol in the input diagram. In comparison, our work investigates both the localization and the recognition of the symbols.

3.2. Sketch understanding

The advent of the tablet PC technology and the surge of interest in pen-enabled software applications have spawned a number of recognizers for sketched diagrams. Earlier works used constraints to reduce the complexity of recognition but compromised the natural fluidity of sketching. For example, earlier works assumed each symbol to be drawn in a single stroke [30,31] or in a pre-defined order [32,33], or required explicit user indications (e.g., gesturing with the digitizer or pressing a key) to demarcate the various symbols [34]. The FEAsy system [35] featured a rule-based recognizer specialized in engineering symbols commonly used during Finite Element Analysis. LADDER [36], also using recognition rules, was designed to work across multiple domains. Expert knowledge is required to generate the recognition rules, and the resulting recognizer is not trainable, thus suffering from fixed biases and error rates. Probabilistic inference has been recently used as an alternative [37–40]. Such approaches incorporated the data-driven learning of parametric classifiers and could improve the recognition performance with the accumulation of training data and user feedback. However, they made limiting assumptions regarding the input diagrams to reduce the computational complexity. Kara et al. introduced a more tractable *mark-group-recognize* heuristic and applied it to several domains, including control system diagrams [41] and multi-body mechanical vibratory systems [12]. The definition of landmarks, crucial to the success of this approach, is domain-specific. In fact [41] and [12] chose different landmark symbols based on the specific graphical layouts of the different domains. This precludes an easy re-targeting of this approach to new engineering domains. Moreover, a common trait of the above approaches is that they rely on the temporal sequence or demarcation of the strokes. Hence they are dependent on specific input devices (e.g., tablet PCs, digital white boards) and cannot be utilized if the input diagrams are acquired as images.

Instances of image-based recognition techniques are yet insufficient to recognize complete engineering diagrams. Saund et al. [42] presented a system that uses Gestalt principles to search for the perceptually closed, low-level geometric primitives in diagrams, but without the recognition capability for semantically meaningful symbols. Notowidigdo and Miller's [43] offline sketch recognition technique used hard-coded rules to find simple geometric primitives (e.g., circles, rectangles) in a diagram and used empirical thresholds to filter false positives. The empirical thresholds may cause a high rate of false positives. Kara et al. [44] described an image-based recognizer for isolated, hand-sketched symbols featuring high geometric variations. They extracted template matching distances as features for recognition. It resorts to a separate, stroke-based technique to locate the symbols within the input image. Pu and Gur [45] described a sketch segmentation approach based on implicit function fitting and a greedy search algorithm. It is applicable to both stroke-based input from tablet PCs and pixel-based input from image acquisition devices. However, this segmentation approach results in low-level primitives such as individual lines and arcs, rather than semantic-level primitives such as symbols. No work has been reported on its integration with symbol recognizers. Ouyang and Davis's symbol recognizer [46] utilized a feed-forward feature extractor and a deformation template matching algorithm. It shows robustness in isolated symbol recognition. However, it lacks the symbol localization capability and its application to complete diagram recognition has not been reported.

4. Proposed approach

4.1. Overview

The problem of diagram recognition can be decomposed into four sub-problems: (1) What are the symbols drawn in this input image or image patch? (2) Where are the symbols located in this input image, or in other words, which image patch contains a symbol? (3) How are the symbols connected to each other? (4) Once the symbols and their connectivity are known, how is the engineering model built? Those four sub-problems are denoted as symbol recognition, localization, connectivity analysis and post-processing, respectively.

Our recognition approach (see Fig. 1) therefore consists of four functional modules, each addressing one of the sub-problems above. Synthetic and sketched engineering symbols and diagrams are used as the running examples to demonstrate the modules in the subsequent sections.

4.2. Module for recognition: convolutional neural network

4.2.1. Network architecture

The core module of our approach is a multi-layer Convolutional Neural Network (CNN, see Fig. 3). It performs symbol recognition by mapping a fixed-size input image to a *label* that indicates the symbol category of the input image. A detailed mathematical definition and analysis of the CNN is described in [10]. Here we briefly describe the network architecture and emphasize the intuition behind it.

Each layer of the CNN stores a collection of trainable parameters and defines a parametric function that computes the output from the input. The exact form of the parameters and the layer function depends on the type of the layer. For example, a convolutional layer such as Convolutional Layer 1 (CL1) in Fig. 3 stores N_f instances of $k_w \times k_h$ trainable image filters. These filters are essentially feature extractors that can be trained to respond to distinctive local features such as oriented edges, line ends and junctions. Given a $w \times h$ input image, CL1 performs 2D convolutions between the input image and each filter, applies nonlinear thresholding to the convolution results and produces N_f instances of $(w-k_w+1)\times(h-k_h+1)$ 2D intensity matrices called the *feature maps*. Each feature map represents the spatial distribution of a feature, reminiscent of a map of landmarks, hence the name.

The sub-sampling layer, another type of layer in the CNN, works in tandem with the convolutional layer. Consider the case of SL1 in Fig. 3, it takes the output feature maps from CL1 as its own input, and performs weighted local averaging to sub-sample its inputs by a factor of s, resulting in N_f instances of $\frac{(w-k_w+1)}{s} \times \frac{(h-k_h+1)}{s}$ feature maps. This process serves to blur the feature maps such that the subsequent layers will not be sensitive to the exact locations of the detected features. In doing so, the CNN gains invariance to moderate local distortions such as translations and rotations [47].

The third and last type of layer in the CNN is the fully connected layer commonly found in regular neural networks [48]. In Fig. 3 this corresponds to FCL and OL. For both layers, the input vector is multiplied by a trainable weight matrix and then nonlinearly thresholded, resulting in the output vector. FCL and OL function together as a two-layer neural classifier. The input vector to FCL

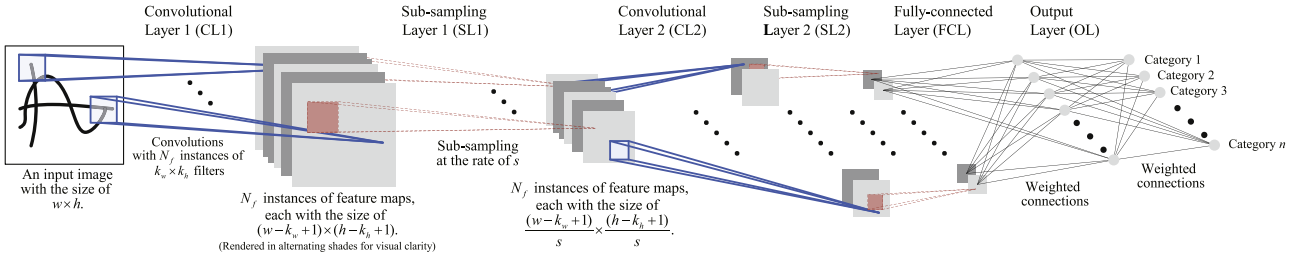


Fig. 3. The architecture of the Convolutional Neural Network. The input is an image patch featuring a Sine Wave signal source in the domain of control systems (defined in Fig. 14).

is a reshaped version of all the output feature maps (essentially a 3D tensor) from the sub-sampling layer immediately before FCL, namely SL2 in the case of Fig. 3. The output of FCL is forwarded to OL as the latter's input. The components of the output vector of OL, also known as labels, represent the pattern categories of interest in the target domain. The component with the maximal numerical value determines the category of the input image.

The CNN trained for several domains in this paper consists of two pairs of convolutional layers and sub-sampling layers in succession (i.e., CL1, SL1, CL2 and SL2), followed by a fully connected layer FCL and finally an output layer OL. The trainable parameters (e.g., filters, weights matrices) of each layer are optimized in a data-driven fashion during training, in order to adapt to the visual features of a specific domain. Refer to Section 4.2.2 for more details regarding the training of the CNN.

The structural parameters of the CNN, such as the number of layers and the number of trainable filters on each layer, are empirically determined during training. A simple rule-of-thumb [10] that works well is to start small and increase those numbers until further increments do not lead to significant performance improvement, or until the training and recognition speeds become unbearable. This is automated using a control script which underlies our choice of a six-layers CNN with the particular amount of trainable filters.

The unique insight is that, by training the CNN parameters, input images in the same category would result in similar intermediate feature maps and eventually the same output label, even in the presence of geometric distortions among them. With such invariance to distortions, robust symbol recognition performance is achieved, as shown in Sections 5 and 6. Moreover, by using a recognizer on image-based inputs, our approach is not limited to stroke-based diagrams from tablet PCs.

4.2.2. Training

Before deployment, the CNN is trained with the Stochastic Gradient Descent algorithm [10], otherwise its internal parameters remain randomly initialized and are expected to perform poorly for the recognition tasks. During training, parameters are updated to minimize the discrepancy between the reference labels and the computed labels on the training set. In doing so, the CNN extracts the visual knowledge that encodes the distinctive features and desired invariance properties of the symbol categories from training samples. We refer the readers to [10,47,49] for detailed mathematical derivations of the gradients and the parameter update equations in CNN. The following paragraphs will focus on the collection and synthesis of training samples.

To initiate training, the users are required to provide a few *labeled* prototypical diagrams on which they have marked the bounding boxes for all symbols and has attached categorical labels (e.g., spring, damper, mass) to the bounding boxes. This is the only manual intervention required from the user. The rest of the training procedure is automated.

The training samples are therefore pairs of image patches cropped from the prototypical diagrams, and their categorical labels. Samples of the defined symbol categories are extracted directly from the user-provided bounding boxes and labels. In addition to the defined categories, one additional "non-symbol" category is initialized by randomly cropping three types of regions from the prototypical diagrams, namely regions without labels, regions having negligible overlaps with the user-labeled regions and regions containing multiple labels. The non-symbol category is introduced to help the CNN suppress false positives that occur when the CNN is used in conjunction with the sliding window method for symbol localization (described later in Section 4.3). The cropped image patches are scaled and padded to fit the input size of the CNN.

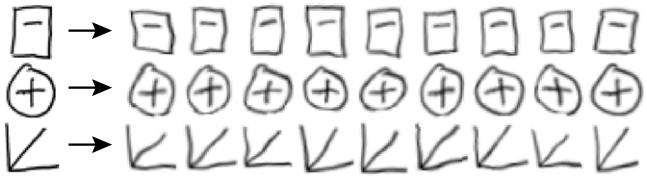


Fig. 4. Seed training samples (first column on the left) and additional training samples synthetically generated from the seeds by random local distortions and global transformations (the other columns). The images are aliased due to quantization. Top row: Scope. Middle row: Sum. Bottom row: Ramp signal.

4.2.3. Expanding the training dataset

Theoretical and empirical evidence [50,10,51] suggests that the training dataset must have a substantial volume and rich variations to train a highly accurate and robust CNN. Therefore the training dataset is expanded to include synthetically generated training samples, without requiring the user to manually create a large training dataset.

Synthetic samples in the symbol categories are generated by applying a set of random local distortions [52] and global affine transformations to the initial samples. The local distortions are applied by computing a new intensity value for every pixel as a smoothed stochastic interpolation of the neighboring pixels. The global transformations include non-uniform scaling, in-plane rotation and shear of the entire image. They mimic the style variations and shape distortions that users and/or image acquisition devices would naturally introduce. Fig. 4 shows several additional training samples generated from seed training samples of hand-sketched symbols in control systems.

To generate more samples in the non-symbol category, the CNN is retrained and additional non-symbol patches are sampled in the process. The CNN is first trained with the initial training sets and tested against the prototypical diagrams with user-supplied labels. Regions causing false positives are then selectively added to the non-symbol training set. A commensurate number of additional training samples are generated using random distortions to keep the numbers of samples balanced across the symbol and non-symbol categories. The CNN is then retrained with the expanded training set. This process is re-iterated to ensure that the network is trained with an adequate number of samples to suppress false

positives on the training set and generalize such ability to testing diagrams. The false positive rate and the accuracy of the CNN can be monitored to determine the termination of such iterations.

Another viable way to generate more training samples is to collect user feedback such as corrections of misclassified and false positive patterns, as well as the absence of corrections, namely the tacit confirmations, of correct yet low-confidence patterns. By adding such patterns into the training set and retraining the CNN with them, the CNN's accuracy can be improved throughout its life cycle. This is a benefit unavailable with non-trainable systems.

4.2.4. Recognition

A trained CNN works as follows to recognize an unknown image patch: it first convolves the input image patch with the set of learned image filters to extract features, then sub-samples the resulting feature maps for invariance against distortions. And the alternation of convolution and sub-sampling is repeated as needed until finally the feature maps are classified using a two-layer fully connected neural network. The final output is a normalized vector whose length equals to the number of categories to be recognized, including the defined categories and the non-symbol category. The n-th output value can be seen as the likelihood that the input belongs to the n-th category. Fig. 5 shows the input image (a Scope symbol), the internal states, the parameters (e.g., learned image filters and weights) and the output vector of a CNN trained for recognizing control system diagrams. Here the output of the CNN corresponds to the correct label of the input image.

The CNN is biologically inspired. Physiologists [53] have discovered "simple cells" that function similarly to a convolutional layer and "complex cells" that function similarly to a sub-sampling layer in the visual cortex of cats. Successful applications of the CNN in optical character recognition [52] and visual detection of human faces from photos [47] have demonstrated the strength of the CNN to handle large variations in the inputs. To the best knowledge of the authors, the proposed approach is the first to apply the CNN to the visual recognition of image-based engineering diagrams.

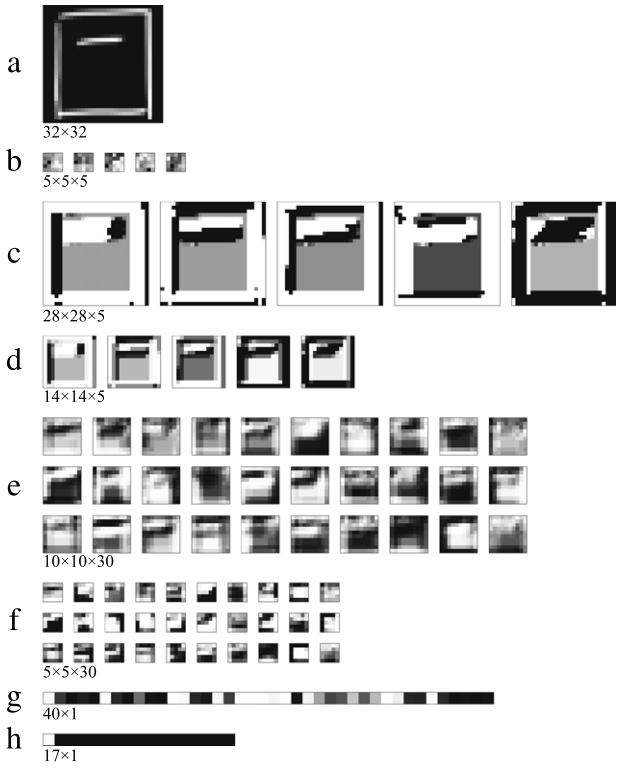


Fig. 5. The internal states and the parameters of a CNN trained for recognizing control system block diagrams. The expression below each row denotes the dimension of that row. (a) A 32-by-32 input image of a Scope block. (b) 5 sets of 5-by-5 filters in CL1. (c) 5 sets of 28-by-28 feature maps produced by CL1. (d) 5 sets of 14-by-14 feature maps outputted by Sub-sampling Layer 1. (e) 30 sets of 10-by-10 feature maps outputted by CL2. (f) 30 sets of 5-by-5 feature maps outputted by SL2. (g) A 40-element vector outputted by FL1, each painted as a gray-scale cells. (h) A 17-element vector outputted by the last layer, each representing the likelihood that the input belongs one of the 17 defined categories. Here the most likely category of the input is the first one, namely the cell with the lightest shade on the left. For visual clarity, each feature map and cell has been normalized to 256-level gray-scale such that the minimum values correspond to black pixels and the maximum values to white.

4.3. Modules for localization

The CNN described in the foregoing section has an input layer of limited size that can be utilized to recognize a single image patch. Referring back to the problem decomposition in Section 4.1, an isolated symbol recognizer such as CNN must therefore be used in conjunction with a localization module to yield a complete recognition of an input diagram. To this end, we propose the following two localization modules: the multi-scale sliding window and the Connected Component Analysis. Both methods take a diagram as input, and output the locations of the symbols in the diagram, without requiring explicit user demarcations or prior definition of heuristic landmarks.

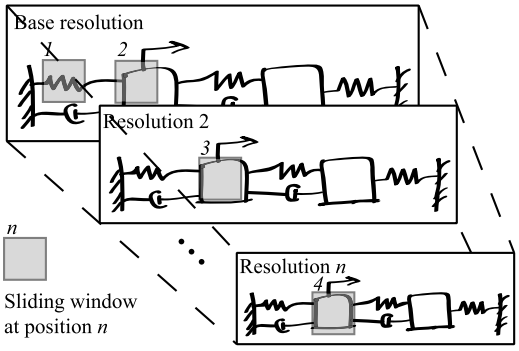
4.3.1. The sliding window and the multi-scale image pyramid

Recall that the major challenges for symbol localization are the scale and location variations of the symbols drawn in the sketch. To handle such challenges, one possible design rationale for the localization module is to search for symbols exhaustively across different locations and scales of the input diagram.

A trivial search scheme is to apply the CNN to the input image through a fixed-size sliding window. This essentially involves searching across different locations at a single scale, and will only yield detections having the same scale as the input window, or figuratively the field-of-view, of the CNN. Symbols larger than that will not fit completely into the sliding window. Symbols substantially smaller could appear too small and blurred due to quantization. In both cases, the CNN will not identify all the distinctive features necessary for symbol detection.

Our solution is to complement the sliding window idea with a multi-scale image representation so as to search across different locations as well as scales. The input diagram is first converted to a gray-scale image and successively sub-sampled to produce a series of images with lower resolutions than the original one. Those images form a multi-scale representation known as the *image pyramid*, as exemplified in Fig. 6. A sliding window of fixed size is then applied to different levels of the image pyramid with a unit step size in both axis directions and thereby traverses every position within the image. The region inside the window is extracted as a single image patch and then passed to the CNN for recognition.

The sub-sampling factor f between successive levels of the image pyramid is chosen to complement the built-in and learned scale invariance of the CNN, such that symbols drawn at any scale within a large, continuous range can be detected. Consider an image pyramid $I_0, I_1, \ldots, I_n$ where I_n is generated by sub-sampling I_0 for n times, each time for a factor of f ($f > 1$). Assume, without losing generality, that a CNN with a $w_0 \times w_0$ square sliding window on I_0 is able to detect symbols whose sizes are within the range of $R_0 = [w_0 \cdot l, w_0 \cdot u]$, because of the CNN's built-in and learned scale invariance. Here l and u are coefficients indicating the lower and upper bounds of detectable symbol sizes, normalized by w_0. Per this definition, $0 < l < u \leq 1$. It is straightforward to see that, if the $w_0 \times w_0$ sliding windows is applied on I_1, then it is equivalent to applying a larger, $(w_0 \cdot f) \times (w_0 \cdot f)$ window on I_0,



Sliding window at position n	Pixels inside the window	Recognition result
1		Spring
2		Not a symbol
3		Not a symbol
4		Body mass

Fig. 6. Sliding windows on different levels of the image pyramid and the recognition results at four window positions.

because I_1 is created by sub-sampling I_0 once for a factor of f. With this $(w_0 \cdot f) \times (w_0 \cdot f)$ window on I_0, it is now possible to recognize symbols whose sizes are within the range of $R_1 = [w_0 \cdot f \cdot l, w_0 \cdot f \cdot u]$. If $f \leq \frac{u}{l}$, then the two ranges R_0 and R_1 will overlap, ensuring that symbols sized within the expanded, continuous range of $R_0 \cup R_1$ can be detected. With an n-level image pyramid, the detectable symbol size range would therefore be equal to $R_0 \cup R_1 \cup \cdots \cup R_n = [w_0 \cdot l, w_0 \cdot u \cdot f^{n-1}]$.

The number of levels in the image pyramid is determined by the largest permissible symbol size which is usually constrained by the dimension of the drawing surface. In the absence of such a bound, the input image can be sub-sampled repeatedly until the whole diagram fits into the sliding window.

A proper choice of those parameters would result in an exhaustive scan of the input with the multi-resolution sliding window, and the chances for the CNN to detect symbols at different locations and scales are thus maximized. For instance, the left-most body mass symbol in Fig. 6, oversized for the sliding window at the base resolution, is gradually shrunk to fit into the sliding window at location 4 in a lower resolution and subsequently recognized by the CNN.

A large number of computations performed by the CNN for overlapping input patches are convolutions using a common set of filters and are therefore redundant. As suggested in [10], we implement the sliding window as follows to improve computational efficiency. Rather than scanning the 6-layer CNN on each level of the image pyramid in a window-by-window fashion, the first four layers of the CNN are first applied to each level in full, producing feature maps approximately four times smaller than the input. Then a sliding window four times smaller than the original one scans the feature maps and feeds the contents inside the window to the last two layers of the trained CNN for recognition. Such operations reuse the convolution results shared between overlapping inputs and produce the same results that would be otherwise obtained without reusing the redundant computation.

Fig. 7 (a) shows that symbols can be detected within adjacent sliding windows or at multiple resolutions, because the CNN is robust to modest translations and scale variations. In such cases, multiple detections by the CNN are grouped according to their

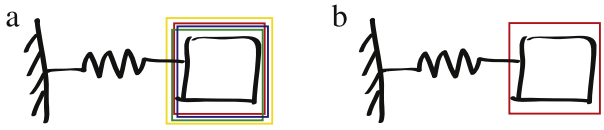


Fig. 7. The detection of a body mass symbol in a simple vibratory system diagram. (a) Multiple detections of the body mass, each with different sliding window position, scale and likelihood output. (b) The detection with the highest likelihood is retained.

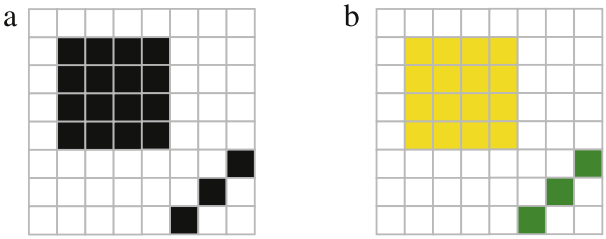


Fig. 8. The result of CCA on an 8×8 image containing two connected components. Each cell in the grid represents a pixel. (a) The input image. (b) CCA results. Colored pixels indicate different connected components.

spatial proximity and the amount of common overlap. The one with the maximum likelihood value is retained while the rest are discarded, as shown in Fig. 7 (b).

4.3.2. Connected component analysis

As an alternative to the multi-scale sliding window, Connected Component Analysis (CCA) [54–56] localizes symbols more efficiently by considering the pixel-level connectivity of the input diagram.

A pixel at location (x, y) is said to be connected to the pixels at the coordinates $(x \pm 1, y)$, $(x, y \pm 1)$, $(x \pm 1, y \pm 1)$ and $(x \pm 1, y \mp 1)$. CCA algorithms output a list of such *connected* pixel regions within a binary input images. Fig. 8 shows an example of CCA result on a toy example containing two connected components: a square blob and a slanted edge.

When a diagram features non-overlapping connectors and symbols that each form a connected component, CCA can be applied in such cases as an efficient and accurate alternative to the sliding window. Fig. 9 shows the following steps of applying CCA to a control system block diagram where the symbols are drawn following the definitions in Fig. 14.

First a binary version of the input diagram (Fig. 9 (a)) was produced by thresholding the gray-scale input image. Pixels with intensities greater than the median pixel value are classified as foreground and those below the median as background. CCA finds the bounding boxes and pixel coordinates of each connected components, as shown in Fig. 9 (b). Note that at this step nothing is passed to the CNN for recognition yet.

Two components are merged into one, if the overlapping ratio ρ (defined below) of the two components under consideration exceeds a trainable decision parameter δ. The overlapping ratio ρ of two components is defined as the overlapping area between their bounding boxes, divided by the bounding box area of the smaller component. $\rho = 1$ corresponds to the special case that the larger competent completely contains the smaller one. The trainable parameter δ is set to the minimum of the overlapping ratio of any two components belonging to the same symbol from the training dataset. Fig. 9(b) and (c) illustrate such a merging process. The small components annotated with arrows in Fig. 9(b), such as the horizontal stroke, the question mark and the plus sign, are merged with components encircling them to form connected components that represent the Scope, Sum and Random symbols, respectively. In the domain shown in Fig. 9 and further detailed in Section 6, the value of δ is found to be 0.75.

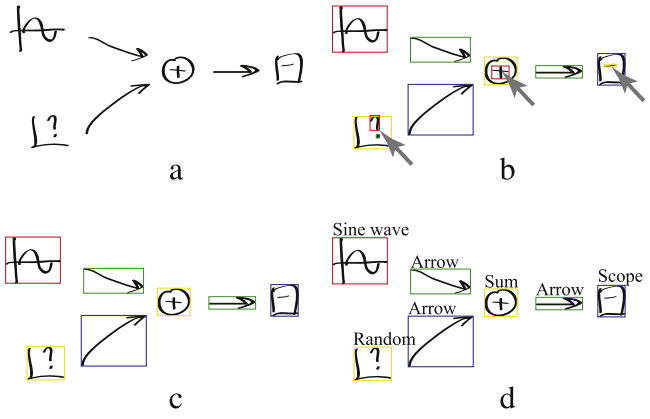


Fig. 9. The exemplary results of CCA on a fragment of a control system block diagram. (a) The input diagram. (b) Initial CCA results. Colored bounding boxes indicate different connected components. Gray arrows indicate small components that are encircled and merged by other components. (c) Merged CCA results. Each merged component corresponds to a symbol. (d) The recognition results of each component produced by the CNN. Note that colors are used to differentiate connected components, rather than symbols. The symbols drawn here are defined in Fig. 14.

Each merged component, corresponding to a defined symbol or connector, is then extracted from the gray-scale input image. The extracted component is then scaled and padded to fit into the input window of the CNN for recognition.

The recognition result of each connected component in Fig. 9(d) will be used in the subsequent steps of connectivity analysis and post-processing. Note that in Fig. 9(d) colors are used to differentiate connected components, rather than color-coding different symbol categories.

With the CCA, the CNN is applied only once per connected component. The number of components is typically far smaller than the number of sliding windows. This leads to increased recognition speed over the sliding window and reduces false positive detections, as observed in Sections 5 and 7.1.3. Moreover, unlike the sliding window, only one detection per symbol is produced, thus eliminating the need to merge and suppress duplicate detections. However, due to the assumption on the disconnected pixel connectivity in the input image, it is not as generally applicable as the sliding window. We are investigating symbol localization schemes that combine the advantages of both. Recent works [57,58] in computer vision have provided pointers in this direction.

4.4. Modules for connectivity analysis

Given an input diagram, the output from the two previous modules, i.e., symbol localization and symbol recognition, are the bounding box location, size, and label of each symbol contained in the diagram.

In some engineering domains, such as rectilinear mechanical vibratory systems, the above information is sufficient to derive the engineering model embodied in the diagram, because the relative locations of the symbols implicitly determine the connectivity between the symbols. For example, a horizontal spring is always connected to the symbol to its immediate left or right. In such cases, it is unnecessary to analyze whether there are foreground pixels connecting the spring symbol to others.

In other domains, however, the connectivity between detected symbols requires further analysis. For example, electric circuits feature line connectors between the symbols. In this case, the connectivity can be analyzed as follows: First, the connectors are isolated as individual connected components. If CCA is used, then

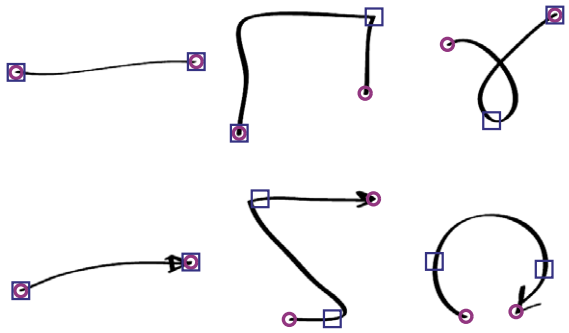


Fig. 10. Locating connector ends with LLE and PCA. Magenta circles denote ends located by LLE and blue squares denote PCA results. PCA fails when the connecter exhibits nonlinear structure.

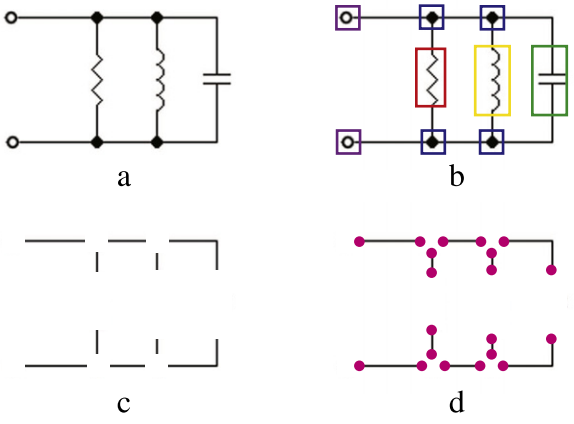
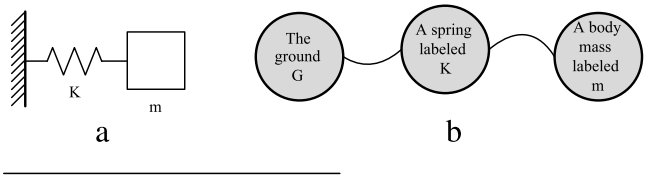


Fig. 11. Connectivity analysis applied to a simple RLC circuit. (a) The input image. (b) Recognition results by the sliding window. Colored bounding boxes indicate different symbol categories. (c) Recognized symbols are masked with white pixels and connectors are left as individual connected components. (d) The ends of the connectors are located using LLE and shown in magenta dots.

connectors are already known to be the connected components discarded by the CNN as non-symbols. If the sliding window is used, all pixels inside the bounding boxes of the recognized symbols are masked by the background pixels such that the canvas only shows the connectors as individual connected components.

Next, each individual connector undergoes morphological thinning to reduce the number of pixels. Afterwards, a locally linear, one-dimensional embedding of the 2D coordinates of the remaining pixels is computed for each connected component using the Locally Linear Embedding (LLE) algorithm [59]. The pixels with the minimum and maximum coordinate values in the one-dimensional embedding are picked as the two ends of the connector. This is similar to [60] which used Principle Component Analysis (PCA, [61]) to order sample points from multiple over-tracing strokes. The primary difference is that here LLE maps the 2D pixel coordinates to an embedded, nonlinear 1D manifold within the 2D pixel coordinate space, whereas in [60] PCA maps the point coordinates to a 1D straight line passing through the 2D point cloud representing the over-tracing strokes. Therefore, LLE is able to unroll curved connectors that form nearly closed loops, whereas PCA cannot handle such cases, as shown in Fig. 10 and observed by [60].

Finally, each end of a connector is assigned to the nearest detected symbol. Two symbols assigned to the same connector are thus connected. Fig. 11 shows an example of the above procedure applied to a simple RLC circuit. Note that two different types of sliding windows (i.e., square and rectangular) are used in order to



Vertex	Attributes
K	A spring located at (120, 110)
m	A body mass located at (260, 110)
Ground	Immovable ground, located at (10, 110)

c

Edge	Connected vertices
Km	K, m
KG	K, Ground

d

Fig. 12. (a) A simple diagram depicting a mass–spring system. (b) The corresponding attributed graph after recognition, localization and connectivity analysis. (c) The attributes of each vertex. (d) The unattributed edges between vertices.

obtain tight bounding boxes of symbols and avoid misclassifying pixels of neighboring symbols or connectors.

Diagrams of some engineering domains feature directed connections. For example, in control system block diagrams, arrows are drawn between symbols to indicate the flow of signals. A directed connector is none other than an undirected connector featuring additional pixels on one end known as the arrow head. To locate the arrow head, a small (e.g., 10×10) image patch centered around either ends of the connector under consideration is extracted and the one with greater density of foreground pixels is marked as the arrow head.

At this point, all pieces of information required to derive the engineering model (i.e., the locations, labels and connectivity of the symbols) are extracted from the input diagram. Next, an attributed graph similar to that shown in Fig. 12 is constructed, with vertices corresponding to detected symbols, edges corresponding to detected connectors, and attributes corresponding to the categorical labels and the default model parameters of the symbols. This attributed graph is an application-independent interpretation of the diagram and an abstraction of the model conveyed by the diagram. It decouples the application-dependent post-processing from generalized diagram recognition. From a system builder's perspective, it facilitates system modularization and verification.

4.5. Application-specific post-processing

Application-specific post-processing designed by domain experts can be applied to the attributed graph to yield the final engineering model and perform the engineering computations of interest. For model construction and simulation, programmable interface of simulators (e.g., `add_block` and `add_line` functions in MATLAB Simulink) can be called to automatically produce the engineering model from the attributed graph. For searching and editing purposes, the attributed graph is just stored for subsequent use and the search or editing routines will access those graphs as needed. Section 6 illustrates those applications.

5. Evaluations

5.1. Symbol recognition performance of the CNN

This subsection evaluates how well the CNN symbol recognizer handles the fundamental challenge in symbol recognition: the shape variations due to distortions. We evaluate the CNN against a standard symbol recognition benchmark dataset: the HHreco dataset of sketched symbols [62]. This dataset includes 7791 shapes in 13 categories. The symbols were collected from 19 different users. Originally the data was recorded on a tablet PC as time series of pen tip motions. Here they are quantized as uniform-width gray-scale images. Examples are shown in Fig. 13.

Our choice of this dataset for evaluation is based on two reasons. First, the task of recognizing these symbols is representative of

Fig. 13. Sketched shapes from the HHreco dataset. Top row: ellipse, heart, trapezoid, pentagon, arch, hexagon, square. Bottom row: triangle, parallelogram, moon, call-out, cube, cylinder. Images are aliased due to quantization.

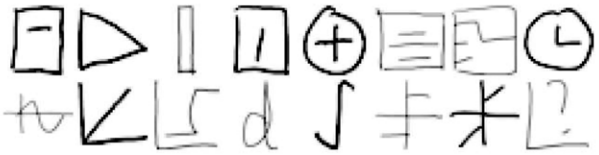


Fig. 14. 16 defined symbol categories of control system block diagrams. Top row: Scope, Gain, Mux, Constant, Sum, Transfer Function, Switch, Clock. Bottom row: Sine Wave, Ramp, Step, Derivative, Integrator, Sign, Columbus and Viscous Friction, Random. Images are aliased due to quantization.

symbol recognition tasks in different engineering domains. Second, the testing set contains a large number of highly variable, hence challenging, test cases.

In line with published works on this dataset, we train the CNN with data from 18 out of the 19 users available and test on the data from the hold-out user without generating supplemental training data. An accuracy of 97.7% was obtained. This result is higher than most published results ranging from 92.2% to 96.7% [62] and is competitive to the state-of-the-art result of 98.2% by Ouyang and Davis [46].

5.2. Sketch recognition performance of the overall system

This subsection evaluates how well the overall approach performs in different application domains with diagrams acquired with different devices. In particular, we evaluate the performance of recognizing control system diagrams and mechanical vibratory system diagrams.

For the control system diagrams domain, 16 symbol categories are defined, following the display styles in MATLAB Simulink. The symbol definitions are shown in Fig. 14. The connectors in this case are arrows. Considering the disconnected nature of this domain, CCA localization is used.

A total of 30 sketches are drawn by 11 users with a digital tablet, pen-and-paper or Microsoft Word line drawing tools. A total of 328 symbols belonging to the 16 defined categories are labeled. Three sketches are used as the initial training set and approximately 400 synthetic training samples per category are generated. The rest, containing 264 symbols, are used as the test set to benchmark the accuracy of the recognition approach. 253 symbols (96%) are correctly recognized and converted to Simulink functional blocks. Fig. 16 shows several frequently occurring cases of misclassifications of very similarly shaped symbols. Because CCA localization is used, there is no false positive detection of symbols, only misclassifications.

To demonstrate the CNN's robustness against stylistic variations, we have additionally constructed a miniature dataset containing 80 symbols shown in Fig. 15. These symbols feature significant variations that are not present in the training samples, such as over-tracing strokes, dot-dash strokes, strokes with non-uniform ink widths and occasionally noise pixels, bent or sheared strokes, and wavy strokes. Our CNN recognizer is able to correctly recognize 71 out of the 80 symbols. We note that such accuracy, though indicative of the CNN's robustness, does not directly

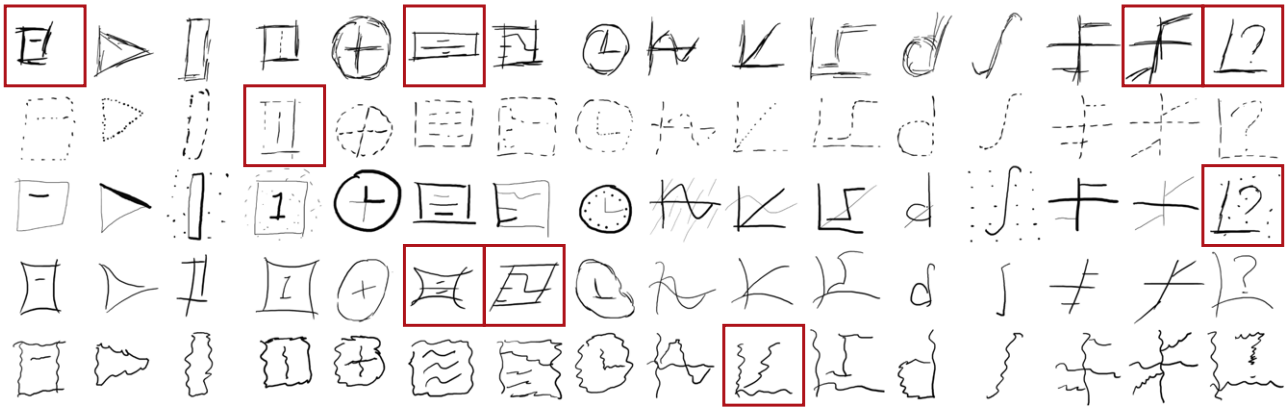


Fig. 15. A miniature dataset of 80 control system symbols featuring stylistic variations not present in the training dataset. Each row from top to bottom: over-tracing strokes, dot-dash strokes, strokes with non-uniform widths and occasionally noise pixels, bent or sheared strokes, and wavy strokes. The 9 symbols with bounding boxes are misclassified by the CNN, while the rest are correctly recognized.

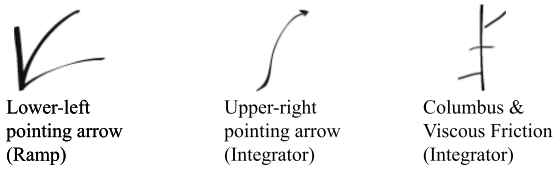


Fig. 16. Misrecognitions of symbols in control system block diagrams. The text annotations outside the parentheses are ground truth and the text annotations inside the parentheses are the incorrect labels outputted by the CNN.

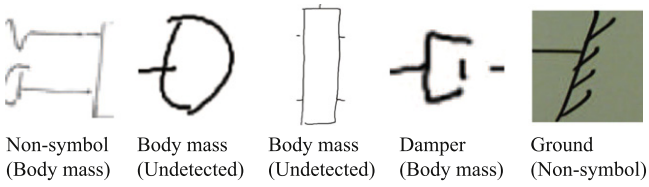


Fig. 17. False positives and misrecognitions of vibratory system diagrams. The text annotations outside the parentheses are ground truth and the text annotations inside the parentheses are the incorrect labels outputted by the CNN.

correlate with the recognition accuracy on real-world data, due to the extreme nature of the variations here.

For the mechanical vibratory diagrams, five symbol categories are defined following engineering convention, including body mass, spring, damper, excitation and ground. The sliding window approach is used for localization, because the CCA approach is not applicable due to the connected nature of vibratory system diagrams.

A total of 36 diagrams are captured from tablet PC sketches, hard-copy technical journals and white boards. 359 symbols belonging to the five defined categories are drawn and labeled by 9 users. Four diagrams are used as the initial training set and the training set was expanded to contain approximately 600 synthetic training samples per category. The rest of the diagrams, containing 307 symbols, are used as the test set to benchmark the accuracy of the recognition approach. Up to 283 symbols (92%) are correctly recognized.[1] In addition, 10 cases of false positives are produced after merging the symbol detections outputted by the CNN,[2] which constitute a very small false positive rate given the huge number of CNN evaluations on the input image. Exemplary cases of recognized diagrams are shown later in Fig. 19. Representative cases of false positives and misrecognition due to ambiguously shaped or severely distorted symbols are shown in Figs. 17 and 19(k).

Unlike symbol recognizers, published works on sketch recognizers are often evaluated on different datasets not publicly available. Hence no detailed comparison can be drawn here and we can only state that our overall accuracy of above 92% is at least numerically on par with published results on similar engineering domains [41,63,12]. Yet more importantly, our approach can be used on image-based diagrams from different engineering domains, while previous works in sketch understanding assumed online features from tablet PCs or domain-specific heuristics. Enabling useful engineering computations on widely available, image-based diagrams from multiple domains is one of the key contributions of our CNN-based approach.

6. Application scenarios

6.1. Building engineering models from diagrams

In this application, we describe the automated conversion from static engineering diagrams to functional engineering models. The resulting models can then be analyzed through numerical simulations and visualized using animations or time series plots. We will focus on diagrams of control systems and mechanical vibratory systems, which are the domains selected for performance evaluation in Section 5.2.

Both domains have been solved by prior works [41,12] utilizing domain-specific features and online or stroke-level features from tablet PC inputs. In contrast, our approach has the advantage of general applicability: by working with both domains, we demonstrate the domain independence of our approach; by working with tablet PC drawings as well as digitally captured diagrams created on physical media such as paper and white boards, we show the independence from online features available only from tablet PCs.

For the control system diagrams, the CNN recognizer is used in conjunction with the CCA module for localization, because the symbols and connectors here are drawn without overlaps. MATLAB Simulink is used as the post-processor to build the recognized engineering model and simulate its dynamic behavior. Exemplary recognition results are shown in Fig. 18.

For the diagrams of rectilinear mechanical vibratory systems, the sliding windows approach of localization is used. By applying

[1] We believe that the difference between the accuracy with mechanical vibration diagrams and that with Simulink diagrams is due to a few user-introduced distortions that are not covered by the training samples. Examples include body masses of elongated or circular shape, or dampers that visually resemble capacitors or body masses.

[2] Many false positive detections produced by the CNN have low likelihood output. Therefore they are merged and eliminated by correct symbol detections with high likelihood and become transparent to the user. The 10 cases here refer to false positives that are not eliminated by the nearby or overlapping symbol detections that are correct.

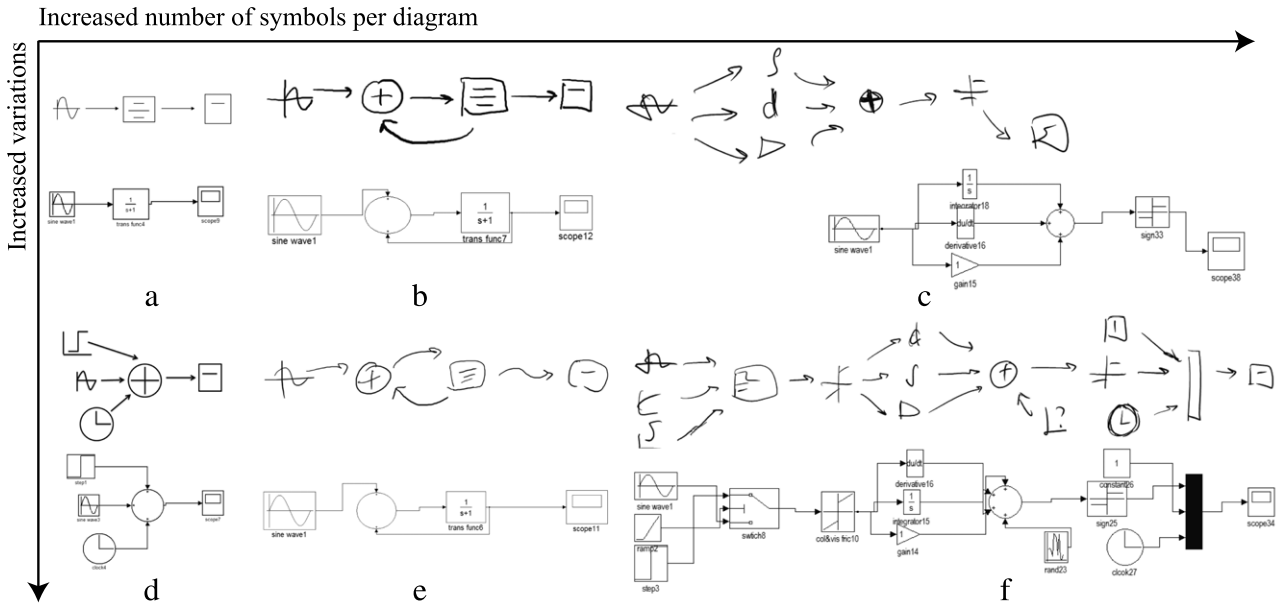


Fig. 18. Recognized control system block diagrams and their corresponding Simulink models. From left to right, the diagrams include more components. From top to bottom, they show more shape variations (a) and (d): Open-loop control system diagrams created with the aid of drawing tools. (b) and (e): Hand-sketched closed-loop control system. (c) and (f): Hand-sketched PID control system.

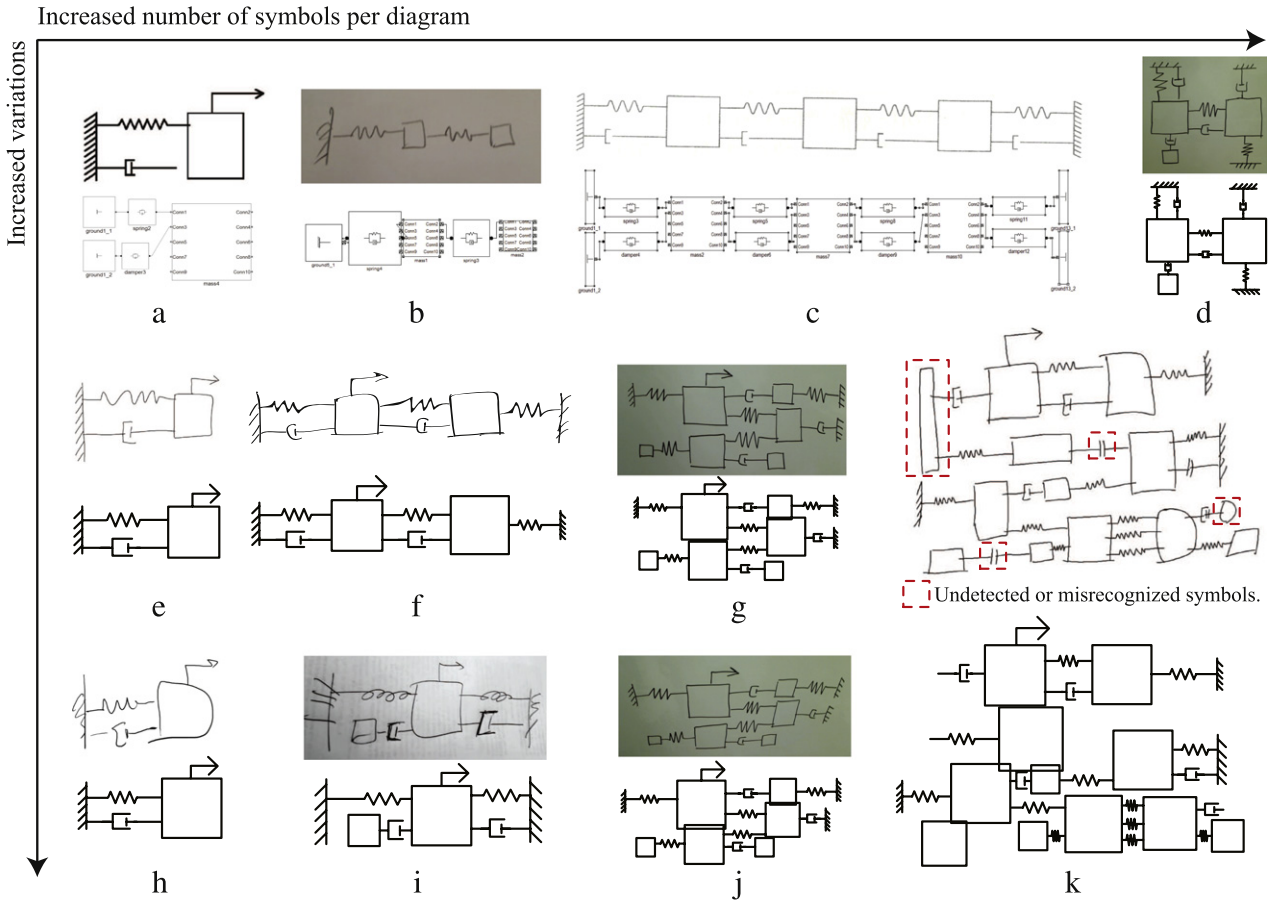


Fig. 19. Recognized vibratory system diagrams and their corresponding SimMechanics models. From left to right, the diagrams show more components. From top to bottom, they show more shape variations. (a), (e) and (k) are screen shots from electronic documents. (b), (d), (g) and (j) are captured from drawings made on paper or on the white board. Note that (g) is (j) subject to foreshortening and camera distortions. (c) and (i) are scanned from printed media. (f) and (h) are sketched with a tablet PC. Each sub-figure shows the input at the top and the recognition result below. For visual clarity, the corresponding models (d) to (k) are displayed as computer generated drawings, rather than the SimMechanics models. (k) is a case not fully recognized. The red bounding boxes with dashed borders indicate misclassified symbols.

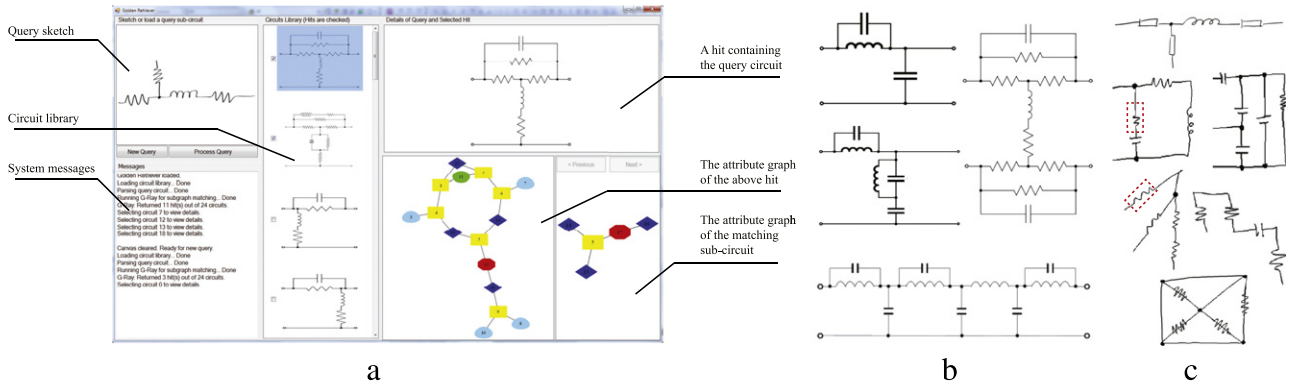


Fig. 20. (a) User interface of the RLC circuit retriever. (b) Examples of RLC circuits drawn with software tools and recognized for retrieval. (c) Examples of sub-circuit query items. The bottom three are extracted from the circuits collected in a previous study [63]. The red bounding boxes with dashed borders indicate misclassified symbols. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the sliding window to the rotated versions of an input diagram that are rotated at the increment of 90°, the CNN is able to recognize 2D rectilinear vibratory systems. MATLAB SimMechanics serves as the post-processing module that simulates the dynamic behavior of the model. Exemplary recognition results, displayed in SimMechanics style or in a simplified form, are shown in Fig. 19.

6.2. Digitization and retrieval of engineering diagrams

In this scenario, we demonstrate an application that performs content-based information retrieval of RLC resonant circuit diagrams. This application, whose interface is shown in Fig. 20(a), recognizes circuits using the CNN and the sliding window. The circuits to be recognized include those drawn with software tools and stored in a database, and those sketched freehand and used as the query terms. Both database and query term are converted to attributed graphs and an attributed graph matching algorithm [64] is used for retrieval.

Five symbol categories are defined following the engineering convention, including resistor, inductor, capacitor, junction and terminal. 24 diagrams of various formally drawn RLC circuits are downloaded from the internet, among which 448 symbols were labeled. 6 diagrams containing 82 symbols are used to train the recognizer. In addition to the above circuits drawn using diagramming software, we have also collected 34 hand-sketched sub-circuits as the query items. 8 of them are screen snapshots from the electronic report of a previous study [63], slightly modified to conform to the symbol definition here. The rest of the query items are drawn by users on a tablet PC. 4 snapshots and 6 hand-sketched diagrams are used for training and the rest are hold out for the performance evaluation. The evaluation shows that all of the formally drawn circuits are correctly recognized, while 22 out of 24 (91%) hand-sketched query circuits are correctly recognized. Fig. 20(b) and (c) shows examples of those circuits.

In the domain of electric circuits, we observe that the bounding box aspect ratios of junctions and resistors differ significantly. The former is approximately 1:1 whereas the latter is usually 2:1 or greater. If we force the CNN's input size to conform to one value, then the detected bounding boxes for the categories with a significantly different aspect ratio would not be tight and might include pixels belonging to neighboring connectors or symbols. As a result, those pixels will be erroneously masked as part of the symbol and cause errors during connectivity analysis.

To prevent this from happening, two CNNs with different input aspect ratios are trained. The first, responsible for the detection of resistor, inductor and capacitor, has an input window with the aspect ratio of 2:1. The second, responsible for the detection of junction and terminal, has an input window of 1:1 aspect ratio. The dual recognizer treatment works well without leaving undetected connectors or symbols. We also note that here the use of two CNNs does not break the generality of the proposed approach, although only one CNN is seen in previous application scenarios. The reason is that the need for multiple CNNs can be statistically, rather than manually, determined from the distribution of the aspect ratios of user-labeled bounding boxes in the training dataset. Multi-modal distributions, for example, would necessitate multiple CNNs.

After training, our circuit retrieval application is able to take a hand-sketched circuit as the query term and find, from the stored diagram database, one or more matching circuits that contain the query term as sub-circuits.

This application scenario enables engineering diagram retrieval based on the higher-level, semantic information conveyed by the pixels, which is more precise and intuitive compared to the alternatives such as template matching of the lower-level, pixel patterns of the query and indexed images, [65], or text-based search of the diagram annotations using textual queries. In addition, this application scenario shows that our approach is applicable to work with hand-sketched images as well as images drawn with software support.

6.3. Intelligent editing

Here we demonstrate the utility of diagram recognition in facilitating diagram editing. Given an input diagram to be edited, our editing interface first recognizes the symbols using the CNN and a localization module applicable to the domain, and then uses the recognized symbols to simplify editing operations.

If the users wish to edit a symbol in a pixel-based image, they need to apply the editing tool to all pixels belonging to that symbol, usually by moving the pointing device. Alternatively, with the symbols recognized, the users could select and edit only a convenient subset of pixels belonging to that symbol. Our interface will propagate the editing operations to the unselected pixels that also belong to the symbol of interest. Fig. 21 shows an example of this scenario applied to the domain of floor plans.[3] The user is able to delete the bed symbol by crossing it out with the eraser tip of a digitizer or a mouse cursor. Without diagram recognition and intelligent editing, the same editing trajectory would only erase part of the bed symbol. The user would then have to move the eraser over all the pixels of the bed symbol to achieve the same

[3] This domain targets floor plans generated using Microsoft Visio and captured using screenshots. The CNN recognizer is trained using 1 example per symbol category for the five symbol categories shown in Fig. 21. The CCA localization module is used. Because the test diagram is drawn using the same software tool as the training example, there is no recognition error. We note that this scenario is not particularly challenging to recognize, but nonetheless demonstrates the potential supportive value of diagram recognition in diagram editing.

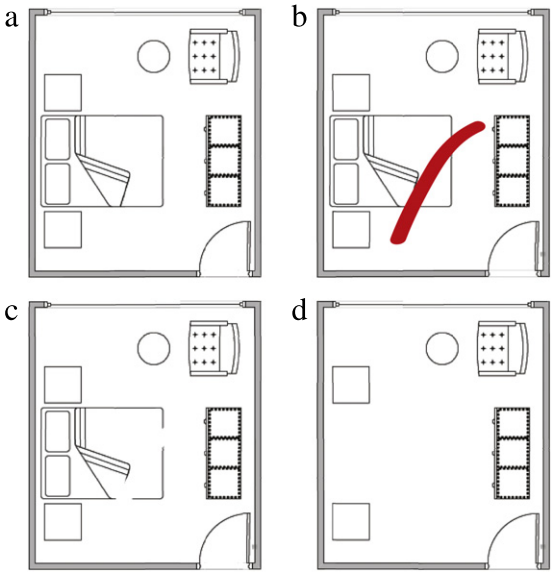


Fig. 21. Example of deleting a bed symbol with the intelligent editing interface. (a) The input floor plan to be edited. (b) The user moves the eraser along the red trajectory. (c) Without semantic support, only pixels on the eraser trajectory is deleted. The user would have to apply the eraser on all pixels belonging to the bed symbol, with care that pixels of the two night stands are not erased accidentally. (d) With semantic-awareness, the intelligent interface would erase all the pixels of the bed symbol when the cursor movement of (b) is performed.

result, with care not to erase pixels belonging to other adjacent symbols.

This is similar in spirit to ScanScribe [66]. However, our application differs from ScanScribe in that ours finds semantically meaningful structures (i.e., symbols) using the high-level, domain-specific visual information, while ScanScribe is based on the low-level, domain-independent Gestalt Laws of Grouping.

7. Discussions

7.1. Addressing the challenges posed in Section 2

7.1.1. Invariance properties and recognition accuracy

Due to its network architecture and training, the CNN has built-in and learned invariance to shape distortions inherent in freehand sketches and distortions induced during image acquisition. It yields accuracy on par with the state-of-the-art sketch recognition systems. The overall system is able to extract the intended engineering models from input diagrams for the computations of interest, regardless of how the diagram is drawn (computer-aided versus freehand) or where the diagram is drawn (digital media such as tablet PCs versus physical drawing media such as paper).

7.1.2. Trainability

Because of the trainable nature of this approach, the recognition accuracy can be improved by interactive user feedback such as confirmations and corrections. Also, to target a new engineering domain, the user may simply provide new training samples (i.e., labeled prototypical diagrams in the target domain) and retrain the recognizer, rather than manually tuning the recognizer to the new domain. And we see no reason why other engineering domains cannot be tackled using our approach. For example, bond graphs, petri nets or state machines are drawn in a way similar to the Simulink block diagrams, that is, with symbols and connectors disconnected at the pixel level. Therefore those domain can also be recognized using the CNN combined with CCA. Digital circuits, UML diagrams, flowcharts and family trees, similar to the RLC circuits and mechanical vibratory diagrams, can be recognized using the CNN with the sliding window.

The procedure to expand the training set automates the generation of additional symbol and non-symbol samples, relieving users the burden of manually drawing and labeling many additional training sketches.

7.1.3. Computational complexity and processing time

Currently, the processing time of the sliding window is longer than that of the CCA. With the sliding window, the number of CNN evaluations is on the order of $O(wh)$, where w and h are the width and height of the input image and wh is the number of pixels. With the CCA, the number of CNN evaluations is on the order of $O(n_{comp})$, where n_{comp} is the number of connected components, namely the symbols and the connectors. CCA itself has the time complexity of $O(wh)$ [56], but we observe that it does not add a significant overhead: With a 1024×768 input diagram, the running time with the sliding windows is approximately 9 s on a 2.26 GHz CPU. The running time with the CCA is below 1 s when the number of symbols are below 40.

In both cases, a fairly interactive conversion from diagrams to models is achieved. There is still room to improve the processing speed by incorporating parallel processing and processor-optimized convolution codes. Formal user studies are still needed to assess the impact of the processing speed on the user-perceived effectiveness of the system and the user performance in design tasks involving diagrams.

7.2. Future extensions

In this research, we have limited the scope of the features of the recognizer. Several potentially useful features are yet to be incorporated. Therefore, we consider the following three major extensions in the future.

First, the Convolutional Neural Network recognizer can be enhanced by exploiting context. Currently the CNN recognition is based entirely on the image-based patterns inside a region of interest, either within a sliding window or a connected component. It is not informed by the context, namely, the symbols detected outside the current region of interest and the domain knowledge that defines valid relative locations between the symbols. This is the reason why the lower-left pointing arrow in control system diagrams is misclassified as a Ramp block, as seen in Fig. 16. Despite its Ramp-like appearance, it is located between other symbols, without an arrow pointing to itself. Such contextual cues could have been integrated into the recognizer and help to distinguish the arrow from a ramp. To this end, it may be desirable to incorporate into the recognition pipeline a contextual classifier such as Conditional Random Fields [67].

In the second place, we have chosen to assume that connectors are non-intersecting solid lines and to not deal with dotted or dashed lines. With this, each connector will be a single connected component if CCA is performed. If this assumption does not hold, CCA will merge multiple connectors into one single component or break down one single connector into multiple components. As a result, the subsequent connectivity analysis performed on each connected component will no longer be valid. Techniques to recognize dotted or dashed lines [68,69] need to be incorporated in the future.

In the third place, additional training samples are synthetically generated using image distortions that are randomly drawn from a prescribed set of distortions. This is simple and empirically effective. Perhaps a data-driven and potentially more effective way of generating training samples is to utilize the distortions that are inferred from existing training datasets, similar to [70].

8. Concluding remarks

In this paper, we present a visual recognition approach for network-like engineering diagrams. This approach leverages a

Convolutional Neural Network as a symbol recognizer and two localization methods applicable to a wide spectrum of network-like engineering diagrams. Evaluations in different application scenarios with different types of diagrams demonstrate the effectiveness of our approach.

With our approach, a computer will be able to recognize the engineering model conveyed by diagrammatical images. The recognition of the engineering information would enable computations of engineering interests on the image-based inputs and this leads to useful applications such as interactive, sketch-based modeling and simulation, digitization and retrieval of hard-copy diagrams and semantic-aware editing of diagrams. We hope the proposed diagram recognition approach, its applications and its future extensions could add computational support to engineering diagrams whose supportive values in engineering problem-solving is known in [5].

Acknowledgements

We would like to thank the anonymous reviewers for their invaluable comments and suggestions. This research was supported by the National Science Foundation CAREER Award #0846730.

References

[1] Ullman DG, Wood S, Craig D. The importance of drawing in the mechanical design process. Computers and Graphics 1990;14(2):263–74.
[2] Shpitalni M, Lipson H. Classification of sketch strokes and corner detection using conic sections and adaptive clustering. ASME Journal of Mechanical Design 1995;119:131–5.
[3] Landay JA, Myers BA. Sketching interfaces: toward more human interface design. IEEE Computer 2001;34(3):56–64.
[4] Yang MC. Concept generation and sketching: correlations with design outcome. In: ASME design engineering technical conferences and design theory and methodology conference. 2003.
[5] Schutze M, Sachse P, Romer A. Support value of sketching in the design process. Research in Engineering Design 2003;14:89–97.
[6] Song S, Agogino AM. Insights on designers' sketching activities in new product design teams. In: ASME design engineering technical conferences and computers and information in engineering conference. 2004.
[7] Chusilp P, Jin Y. Impact of mental iteration on concept generation. Journal of Mechanical Design 2006;128(1):14–25.
[8] Yang MC, Cham JG. An analysis of sketching skill and its role in early stage engineering design. Journal of Mechanical Design 2007;129(5):476–82.
[9] Silva Rd, Bischel DT, Lee W, Peterson EJ, Calfee RC, Stahovich TF. Kirchhoff's pen: a pen-based circuit analysis tutor. In: Eurographics workshop on sketch-based interfaces and modeling. 2007.
[10] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 1998;86(11):2278–324.
[11] Haralick RM, Shapiro LG. Computer and robot vision, vol. I. Addison-Wesley; 1992.
[12] Kara LB, Gennari L, Stahovich TF. A sketch-based tool for analyzing vibratory mechanical systems. Journal of Mechanical Design 2008;130(10):101101.
[13] Fahn CS, Wang JF, Lee JY. A topology-based component extractor for understanding electronic circuit diagrams. Computer Vision, Graphics, and Image Processing 1988;44:119–38.
[14] Okazaki A, Knodo T, Mori K, Tsunekawa S, Kawamoto E. An automatic circuit diagram reader with loop-structure-based symbol recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 1988;10(3):331–41.
[15] Lin X, Shimotsuji S, Minoh M. Efficient diagram understanding with characteristic pattern detection. Computer Vision, Graphics, and Image Processing 1985;30:84–106.
[16] Yu Y, Samal A, Seth SC. A system for recognizing a large class of engineering drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence 1997;19(8):868–90.
[17] Futrelle RP, Kakadiaris IA, Alexander J, Carriero CM, Nikolakis N, Futrelle JM. Understanding diagrams in technical documents. IEEE Computer 1992;25(7):75–8.
[18] Futrelle RP, Nikolakis N. Efficient analysis of complex diagrams using constraint-based parsing. In: International conference on document analysis and recognition (ICDAR). 1995. p. 782–90.
[19] Messmer BT, Bunke H. Automatic learning and recognition of graphical symbols in engineering drawings. In: Goos G, Hartmanis J, van Leeuwen J, editors. Graphics recognition methods and applications, vol. 1072. Springer; 1996. p. 123–34.
[20] Liu W. Example-driven graphics recognition. In: Proceedings of the SSPR2002 (structural, syntactic, and statistical pattern recognition. LNCS, vol. 2396. 2002.
[21] Yan L, Liu W. Engineering drawings recognition using a case-based approach. In: International conference on document analysis and recognition. 2003. p. 190–4.
[22] Yang S. Symbol recognition via statistical integration of pixel-level constraint histograms: a new descriptor. IEEE Transactions on Pattern Analysis and Machine Intelligence 2005;27(2):278–81.
[23] Zhang W, Wenyin L, Zhang K. Symbol recognition with kernel density matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 2006;28(12):2020–4.
[24] Barrat S, Tabbone S, Nourrissier P. A Bayesian classifier for symbol recognition. In: Seventh IAPR international workshop on graphics recognition. 2007.
[25] Luqman MM, Brouard T, Ramel J-Y. Graphic symbol recognition using graph based signature and Bayesian network classifier. In: International conference on document analysis and recognition (ICDAR). 2009.
[26] Escalera S, Fornés A, Pujol O, Radeva P, Snchez G, Llads J. Blurred shape model for binary and grey-level symbol recognition. Pattern Recognition Letters 2009;30(15):1424–33.
[27] Barrat S, Tabbone S. A Bayesian network for combining descriptors: application to symbol recognition. International Journal on Document Analysis and Recognition 2010;13(1):65–75.
[28] Nagy G. Twenty years of document image analysis in PAMI. IEEE Transactions on Pattern Analysis and Machine Intelligence 2000;22(1):38–62.
[29] Cordella L, Vento M. Symbol recognition in documents: a collection of techniques?. International Journal on Document Analysis and Recognition 2000;3(2):73–88.
[30] Ozer OF, Ozun O, Tuzel CO, Atalay V, Cetin AE. Vision-based single-stroke character recognition for wearable computing. IEEE Intelligent Systems and Applications 2001;16(3):33–7.
[31] Rubine D. Specifying gestures by example. Computer Graphics 1991;25:329–37.
[32] Yasuda H, Takahashi K, Matsumoto T. A discrete hmm for online handwriting recognition. International Journal of Pattern Recognition and Artificial Intelligence 2000;14(5):675–88.
[33] Hall A, Pomm C, Widmayer P. A combinatorial approach to multi-domain sketch recognition. In: SBIM '07: proceedings of the 4th eurographics workshop on sketch-based interfaces and modeling. New York (NY, USA): ACM; 2007. p. 7–14.
[34] LaViola J, Zeleznik R. Mathpad2: A system for the creation and exploration of mathematical sketches. In: Proceedings of SIGGRAPH, vol. 23. 2004. p. 432–40.
[35] Murugappan S, Ramani K. FEAsy: A sketch-based interface integrating structural analysis in early design. In: Proceedings of the ASME international design engineering technical conferences and computers and information in engineering conference 2009, vol. 2. 2009. p. 743–52.
[36] Hammond T, Davis R. LADDER, a sketching language for user interface developers. Computer and Graphics 2005;29(4):518–32.
[37] Alvarado C, Davis R. Dynamically constructed Bayes nets for multi-domain sketch understanding. International joint conference on artificial intelligence. 2005.
[38] Alvarado C, Davis R. SketchREAD: a multi-domain sketch recognition engine. In: UIST '04: proceedings of the 17th annual ACM symposium on user interface software and technology. New York (NY, USA): ACM; 2004. p. 23–32.
[39] Sezgin TM, Davis R. Sketch recognition in interspersed drawings using time-based graphical models. Computers and Graphics 2008;32(5):500–10.
[40] Cowans PJ, Szummer M. A graphical model for simultaneous partitioning and labeling. In: AI and statistics. 2005.
[41] Kara LB, Stahovich TF. Hierarchical parsing and recognition of hand-sketched diagrams. User interface software technology. 2004.
[42] Saund E. Finding perceptually closed paths in sketches and drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence 2003;(25):475–91.
[43] Notowidigdo M, Miller RC. Off-line sketch interpretation. In: AAAI fall symposium series 2004: making pen-based interaction intelligent and natural. 2004.
[44] Kara LB, Stahovich TF. An image-based, trainable symbol recognizer for hand-drawn sketches. Computers and Graphics 2005;29(4):501–17.
[45] Pu J, Gur D. Automated freehand sketch segmentation using radial basis functions. Computer-Aided Design 2009;41(12):857–64.
[46] Ouyang TY, Davis R. A visual approach to sketched symbol recognition. In: Proceedings of the 21st international joint conferences on artificial intelligence (IJCAI-2009). 2009.
[47] Garcia C, Delakis M. Convolutional face finder: a neural architecture for fast and robust face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 2004;26(11):1408–23.
[48] Bishop CM. Neural networks for pattern recognition. USA: Oxford University Press; 1996.
[49] Bouvrie J. Notes on convolutional neural networks. MIT CBCL tech report. 2006. p. 38–44.
[50] Huang F-J, LeCun Y. Large-scale learning with svm and convolutional nets for generic object categorization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE Press; 2006.
[51] Osadchy M, LeCun Y, Miller M. Synergistic face detection and pose estimation with energy-based models. Journal of Machine Learning Research 2007;8:1197–215.
[52] Simard PY, Steinkraus D, Platt JC. Best practice for convolutional neural networks applied to visual document analysis. In: International conference on document analysis and recogntion. 2003. p. 958–962.

[53] Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Journal of Physiology 1962;160:106–54.
[54] Shapiro L, Stockman G. Computer vision. Prentice Hall; 2002.
[55] Haralick RM, Shapiro LG. Computer and robot vision. Boston (MA, USA): Addison-Wesley Longman Publishing Co., Inc; 1992.
[56] Suzuki K, Horiba I, Sugie N. Linear-time connected-component labeling based on sequential local operations. Computer Vision and Image Understanding 2003;89(1):1–23.
[57] Lampert CH, Blaschko MB, Hofmann T. Efficient subwindow search: a branch and bound framework for object localization. IEEE Transactions on Pattern Analysis and Machine Intelligence 2009;31:2129–42.
[58] Lehmann A, Leibe B, Gool LV. Fast prism: Branch and bound hough transform for object class detection. International Journal of Computer Vision. http://dx.doi.org/10.1007/s11263-010-0342-x.
[59] Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science 2000;290(5500):2323–6.
[60] Kara LB, Shimada K. Sketch-based 3D shape creation for industrial styling design. IEEE Computer Graphics and Applications 2007;27(1):60–71.
[61] Bishop CM. Pattern recognition and machine learning. Berlin: Springer; 2006.
[62] Hse H, Newton AR. Sketched symbol recognition using zernike moments. In: Proceedings of the 17th international conference on pattern recognition, vol. 1. 2004. p. 367–370.
[63] Gennari L, Kara LB, Stahovich TF. Combining geometry and domain knowledge to interpret hand-drawn diagrams. Computers and Graphics 2005;29(4):547–62.
[64] Tong H, Gallagher B, Faloutsos C, Eliassi-Rad T. Fast best-effort pattern match in large attributed graphs. In: Knowledge discovery and data mining. 2007.
[65] Pu J, Ramani K. On visual similarity based 2d drawing retrieval. Computer-Aided Design 2006;38(3):249–59.
[66] Saund E, Mahoney J. Scanscribe: perceptually supported diagram image editing. In: Diagrammatic representation and inference. Springer; 2004. p. 428–32.
[67] Quattoni A, Collins M, Darrell T. Conditional random fields for object recognition. In: Neural information processing systems. MIT Press; 2004. p. 1097–104.
[68] Dori D, Wenyin L, Peleg M. How to win a dashed line detection contest. In: Goos G, Hartmanis J, van Leeuwen J, editors. Graphics recognition methods and applications, vol. 1072. Springer; 1996. p. 286–300.
[69] Agam G, Luo H, Dinstein I. Morphological approach for dashed lines detection. In: Goos G, Hartmanis J, van Leeuwen J, editors. Graphics recognition methods and applications, vol. 1072. Springer; 1996. p. 92–105.
[70] Learned-Miller E. Data driven image models through continuous joint alignment. IEEE Transactions on Pattern Analysis and Machine Intelligence 2006;28(2):236–50.