Pattern Recognition 42 (2009) 3215 – 3223

Contents lists available at ScienceDirect

Pattern Recognition

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

On-line hand-drawn electric circuit diagram recognition using 2D dynamic programming

Guihuan Feng[a], Christian Viard-Gaudin[b], Zhengxing Sun[a,*]

[a]State Key Laboratory for Novel Software Technology, Nanjing University, 210093 Nanjing, China
[b]IRCCyN/UMR CNRS 6597, Ecole Polytechnique de l'Université de Nantes, France

ARTICLE INFO

Article history:
Received 8 August 2008
Received in revised form 15 January 2009
Accepted 26 January 2009

Keywords:
On-line sketch recognition
Electric circuit diagram
Dynamic programming

ABSTRACT

In order to facilitate sketch recognition, most online existing works assume that people will not start to draw a new symbol before the current one has been finished. We propose in this paper a method that relaxes this constraint. The proposed methodology relies on a two-dimensional dynamic programming (2D-DP) technique allowing symbol hypothesis generation, which can correctly segment and recognize interspersed symbols. In addition, as discriminative classifiers usually have limited capability to reject outliers, some domain specific knowledge is included to circumvent those errors due to untrained patterns corresponding to erroneous segmentation hypotheses. With a point-level measurement, the experiment shows that the proposed novel approach is able to achieve an accuracy of more than 90 percent.

1. Introduction

Sketches are widely used in engineering and architecture fields, especially for the early design phases [1]. This is mainly due to the fact that a sketch is a convenient tool to catch rough ideas, so that the designers can focus more on the critical issues rather than on the intricate details [2]. The problem is that although it seems so quick and intuitive for humans to recognize sketches, it is really a great challenge for the computer [3].

A difficult task in sketch recognition is to have a good balance between the drawing freedom and the complexity of recognition. Generally, the more freely a system can endure, the more difficult sketch recognition will be. Consequently, for the sake of simplicity, most of the existing online recognition techniques are based on the assumption that people will not start to draw a new symbol before the current one has been finished. Obviously this is not always the case. One of the greatest advantages of sketch-based interface is that it provides a natural and free interaction platform. Therefore, it is a significant attempt to try solving the situation with interspersed symbols.

Like speech or text recognition, sketch recognition itself is domain dependent [3]. Domain knowledge to some extent can help recognition. Sketch recognition focuses on the localization and recognition of its constitutional components; the problem is that although isolated symbol recognition has been studied for many years, it still suffers in correctly rejecting outliers. Consequently, recognition based only on symbolic similarity is prone to errors. In this paper, we include contextual constraints to help to solve this problem. Here, constraints refer to the connectivity requirement of symbols, and we introduce a tolerant connectivity evaluation strategy.

This contribution is an extension of the work introduced in [4]. The dataset used for validating the proposed method has been extended from 10 to 15 subjects, with a total number of 130 sketches instead of 87. Furthermore, we have reformulated the problem statement and introduced the solution from a theoretical point of view. Additional experiments have also been conducted to assess the sustainability of a cost function combining pattern recognition information with soft contextual cost, which is a key point of the proposed framework.

The remainder of the paper is organized as follows. Section 2 provides a review of related works. Section 3 formulates sketch recognition as a dynamic programming problem. The details of our approach are presented in Section 4, followed by the experimental results in Section 5. Finally, conclusion and proposed future works are drawn in Section 6.

2. Related works

As mentioned before, many efforts have been done concerning the recognition of isolated symbols that are segmented explicitly by pausing [5] or by switching between different input modes [6]. For example, Rubine [7] proposed an 11-dimensional feature vector to describe a single stroke. Works in [2,8] are analogous, except that stroke segmentation is included, so they can recognize symbols made of multiple strokes. More research on isolated symbol recognition can be found in [9].

* Corresponding author. Tel./fax: +86 25 83686099.
E-mail address: szx@nju.edu.cn (Z. Sun).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.01.031

In order to make sketch-based interaction more free and natural, researchers are working on the automatic parsing and recognition of continuous streams of strokes. Sezgin and Davis [10] made full use of different people's drawing styles to improve both efficiency and performance; Costagliola and Deufemia [11] proposed a left right (LR) based sketch parsing strategy; Sim-U-Sketch [12,13] is a sketch-based interface that depends on a hierarchical "mark-group-recognize" sketch understanding architecture; Gennari et al. [14] employed ink density and stroke characteristics to enumerate candidate symbols; Alvarado and Davis [15] developed a parsing approach based on dynamically constructed Bayesian networks. However, although all these works aim at developing automatic recognition techniques, few of them have addressed the problem of dealing with interspersed symbols. Most of the existing researches are based on the assumption that symbols will not be drawn temporally overlapping each other, which is not always the case. Although Sezgin and Davis [16] extended their approach to recognize interspersed symbols, it still exists a high dependency on the drawing order of strokes. Also, as it concerns only temporal patterns, no spatial or geometric constraint is incorporated. Thus, it is difficult to distinguish symbols that have the same constituent elements but with different structures, such as two horizontal lines and a vertical capacitor.

Hammond and Davis proposed a sketching language LADDER [17]. However, their approach can only describe regular shapes without too much detail, and it is highly dependent on the recognition accuracy of the low level primitives. Our system performs well even when symbols are drawn with over-traced strokes. Saund et al. [18] solved the sketch recognition problem from a perceptual perspective. Gestalt theory is introduced, which argues that human performs domain-independent groupings to locate salient objects. However, it is more suitable for the clustering of texts instead of diagrams.

Domain-specific knowledge is essential in designing a robust sketch recognition system, and it has been widely used in image [19] and video [20] understanding, as well as the recognition of handwritten zip codes [21]. A variety of circuit recognition systems [12–15] have utilized connectivity constraints to help to improve performances. However, in most previous works, connectivity constraints are defined as binary heuristic rules, where a threshold is defined to test whether or not the required connectivity is fulfilled. Since sketches are imprecise in nature, binary connectivity evaluation is prone to errors. Again, due to the moderate rejecting capability of most symbol classifiers to the outliers, we believe that by combining connectivity evaluation with parsing strategy it will greatly help to solve such problems.

In this paper, sketch recognition is formulated as a two-dimensional dynamic programming (2D-DP) problem to process the situations with interspersed symbols. Also, a tolerant connectivity function is proposed to improve both the efficiency and the performance.

3. Problem formulation

Dynamic programming is a powerful tool that can be used to solve planning and decision-making problems [22], and it has been applied to a wide variety of problem domains. The definition of a dynamic programming scheme usually requires the following points:

(1) Divide the problem into several stages.
(2) Identify several possible states at each stage.
(3) Make decisions to change the starting state of this stage and continue to its ending state, which will also be the starting state of the next stage.
Note that the decision made on each state will affect the state for the next stage. In each stage, a decision that achieves a reward closer to the maximum or minimum total reward is desirable.
(4) Define a recursive relationship between the value of the decision at the stage and the previously found optima, i.e. the optimal decision function contains itself in its definition

$$f_k^*(s_k) = \underset{u_k}{\text{opt}}\{v_k(s_k, u_k) + f_{k+1}^*(s_{k+1})\} \tag{1}$$

where s_k represents the starting state of the kth stage, u_k represents the decision made in the kth stage, v^k denotes the transition cost from s_k to s_{k+1}, f_k denotes the function for finding the optimal decision, and opt is the optimal value (maximum or minimum) among all of the u_k which optimizes f_k.

To automatically extract symbols from a freeform sketch, this paper proposes the formulation of the problem with the following Dynamic Programming notations:

(1) stages: number of symbols in the sketch, which is unknown and varies in our application;
(2) state at a stage: defined as an ordered pair (S,V), where S denotes the set of segments extracted after preprocessing, and V represents the set of recognized segments, which is a subset of S;
(3) decision: the newly recognized symbol $(V'-V)$, V' being a superset of V, which comes from hypothesis H_i;
(4) the sketch recognition task can be formulated as

$$\left.\begin{array}{c} f(S,\Phi)=0 \\ f(S,V')=f(S,V)+\text{cost}_L(V'-V); \quad V\subset V'\subseteq S \end{array}\right\} \tag{2}$$

An on-line sketch is basically a sequence of strokes, where a stroke is a sequence of points starting with a pen-down and ending with a pen-lift. Before carrying out the recognition stage itself, each stroke is separated into segments that correspond to perceptual graphical primitives. This is done by an over-segmentation preprocessing step, which assumes that each of the resulting segments belongs to at most one symbol. As a consequence, a symbol is a group of segments that corresponds to a specific label in a domain, such as a resistor or a capacitor in an electric circuit diagram. Sketch recognition starts from (S, Φ), where no symbol has been recognized. Each time a group of segments is added, a new symbol is recognized and f is updated with the recognition cost of $(V'-V)$ (details of how such cost is computed will be illustrated in Section 4.3). The different combinations of segments correspond to different sequences of costs, yielding to different f values. The bigger the cost is, the less likely a true symbol it is. Therefore, the aim of sketch recognition is to find the optimal division $\{H_i\}$ that minimizes the final cost [4], as shown below:

$$\arg\min_{\{V'-V\}} f(S,S) \tag{3}$$

4. Proposed approach

4.1. Flow chart

There are two opposite kinds of strategies to perform online sketch recognition. One is called immediate feedback, which means once a stroke is drawn, sketch recognition will start. The advantage of such strategy is that users can view the recognition results in real time. But it will, to some extent, distract the user during the design task [23]. Again, due to the lack of complete drawing context, such methods always need to place constraints to the drawing style of some specific symbol. In this paper, we adopt another strategy, namely lazy feedback, which means recognition starts only after the