Computers & Graphics 32 (2008) 500–510

Contents lists available at ScienceDirect

Computers & Graphics

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

Sketch recognition in interspersed drawings using time-based graphical models

T.M. Sezgin [a,*], R. Davis [b]

[a] College of Engineering, Koç University, Sarıyer, Istanbul 34450, Turkey
[b] Massachusetts Institute of Technology, CSAIL, Cambridge, MA 02139, USA

ARTICLE INFO

Article history:
Received 15 December 2007
Received in revised form
5 April 2008
Accepted 15 May 2008

Keywords:
Temporal sketch recognition
Dynamic Bayesian networks
User interfaces

ABSTRACT

Sketching is a natural mode of interaction used in a variety of settings. With the increasing availability of pen-based computers, sketch recognition has gained attention as an enabling technology for natural pen-based interfaces. Previous work in sketch recognition has shown that in certain domains the stroke orderings used when drawing objects contain temporal patterns that can aid recognition. So far, systems that use temporal information for recognition have assumed that objects are drawn one at a time. This paper shows how this assumption can be relaxed to permit temporal interspersing of strokes from different objects. We describe a statistical framework based on dynamic Bayesian networks that explicitly models the fact that objects can be drawn interspersed. We present recognition results for hand-drawn electronic circuit diagrams, showing that handling interspersed drawing provides a significant increase in accuracy.

1. Introduction

The activity of sketching is typically (and unconsciously) rather stylized in the sense that people sketch in predictable ways. For example, people typically draw enclosing objects first and use a left-to-right stroke ordering when drawing symmetric objects. There is psychological evidence attributing such ordering phenomenon to motor convenience, part salience, hierarchy, geometric constraints, planning and anchoring [1,2].

The existence of ordering patterns during drawing is significant from a recognition perspective, because, as has been demonstrated in a variety of domains, stroke orderings can be used to aid recognition [3–6]. All previous systems, however, make certain assumptions that limit the complexity of the inputs they can accommodate. One system, for example, assumes the scene contains only one object, drawn in a single stroke [4]. Other systems allow recognition in scenes with multiple objects with the restriction that objects or complete object components are drawn using a single stroke [3]. Another approach allows scenes with multiple objects and objects consisting of multiple strokes, but assumes that no objects share strokes [5]. A more recent framework allows stroke sharing under certain conditions and shows how common stroke orderings as well as object orderings can be used for recognition [6].

Even so, one key assumption that all these systems make about free-hand drawing is that people complete each object before moving on to draw the next. Yet real-world data shows this to be untrue. For example, our analysis of free-hand analog electronic circuit diagrams collected from electrical engineers shows it is not uncommon for people to start drawing a new object before completing the current one. This drawing behavior, which we call *interspersed drawing*, occurs in other domains as well [7]. The ability to deal with interspersed drawing is recognized as a major task that sketch recognizers should support [7]. This paper is focused on this issue, and shows how stroke ordering information can be used for sketch recognition in presence of interspersed drawing. Additional key features of our recognition framework include its ability to learn various kinds of temporal patterns from data, the ability to handle multi-stroke objects and multi-object strokes, and support for continuous observable features.

We formally define the sketch recognition task and describe the interspersed drawing phenomena. In Section 3, we describe a recognition framework based on dynamic Bayesian networks (DBNs) that models online sketching as a stochastic process employing specialized constructs called switching parents. Section 4 reports evaluation results showing that a significant percentage of misrecognitions in interspersed drawings can be avoided by explicitly modeling interspersed drawing behavior. We conclude with a broader discussion of the related work and point out possible future directions.

* Corresponding author. +44 7748 727917; fax: +44 1223 334678.
E-mail addresses: mts33@cl.cam.ac.uk, mtsezgin@csail.mit.edu (T.M. Sezgin), davis@csail.mit.edu (R. Davis).

0097-8493/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2008.05.008

2. Problem definition

Informally, the goal of sketch recognition is to segment digital ink drawn by the user, and then classify it by labeling it as one or more of the objects in the domain. We focus here on the domain of hand-drawn electronic circuit diagrams, but our recognition algorithm is not specific to this domain. Objects in this domain are wires, resistors, capacitors, npn-transistors and batteries.

2.1. Terminology

We adopt the terminology and notation used in [6]. A *sketch* $\mathscr{S} = S_1, S_2, \ldots, S_N$ is defined as a sequence of strokes captured using a digitizer, preserving the drawing order. A *stroke* is a set of time-ordered points sampled between pen-down and pen-up events. Each stroke is broken into geometric objects (e.g., lines, arcs) called *primitives* as part of the preprocessing of the sketch.[1] Let $\mathscr{P} = P_{1:T} = P_1, P_2, \ldots, P_T$ be the sequence of time-ordered primitives obtained from sketch $\mathscr{S}$, and $\mathscr{O} = O_1, O_2, \ldots, O_T$ be the sequence of observations (feature vectors) obtained from the primitives.

We use *segmentation* to refer to the task of grouping together primitives constituting the same object. Given a set of classes $\mathscr{C} = \{C_1, C_2, \ldots, C_n\}$, *classification* refers to the task of determining which object each group of primitives represents (e.g., a stick-figure). Segmentation produces K groups $G = G_1, G_2, \ldots, G_K$, and classification gives us the labels for the groups $L = L_1, L_2, \ldots, L_K$, $L_i \in \mathscr{C}$. Each group is defined by the indices of the primitives included in the group $G_i = \rho_1, \rho_2, \ldots, \rho_m$ sorted in ascending order.

We define *sketch recognition* as the segmentation and classification of a sketch. A simplifying assumption in most sketch recognition systems is that a stroke can be part of only one object. Our definition of segmentation in terms of grouping primitives is more general than a definition based on grouping strokes. By defining segmentation as grouping primitives, we prohibit primitives from being shared across objects, but allow a stroke (which can be composed of multiple primitives) to be part of multiple objects (e.g., using a single stroke to draw a resistor and the wires on either side of it).

We use *interspersing* to mean to the situation where the user starts drawing one object but draws one or more other objects before the first is completed. For example, Fig. 1 shows a circuit fragment in which two wires (#3 and #6) are interspersed with the transistor.

More formally, suppose we have two objects $\mathscr{A}$ and $\mathscr{B}$. Assume the primitive indices for the proper grouping of primitives forming $\mathscr{A}$ and $\mathscr{B}$ are $G_{\mathscr{A}} = \rho_1, \rho_2, \ldots, \rho_m$ and $G_{\mathscr{B}} = \rho'_1, \rho'_2, \ldots, \rho'_n$. We say that $\mathscr{A}$ is interspersed with $\mathscr{B}$ if $\rho_1 < \rho'_i < \rho_m$ for $1 \leqslant i \leqslant n$ and $|G_{\mathscr{A}}| + |G_{\mathscr{B}}| = \rho_m - \rho_1 + 1$. The model we present is in fact able to handle a more general case of interspersing where $\mathscr{A}$ is interspersed with multiple objects.

2.2. Desired features of a model

The main feature of our model is its ability to handle interspersed drawing behavior. However, we also support five features identified as important in previous work.

2.2.1. Learning stroke-level and object-level patterns

Stroke orderings used in the course of drawing individual objects naturally contain certain patterns. For example, when drawing arrows, one frequently seen temporal pattern is a long line (the shaft) followed by two shorter lines (parts of the arrow head). These are called *stroke-level patterns* because they capture the probability of seeing a particular sequence of *strokes* with certain properties when sketching an object [6].

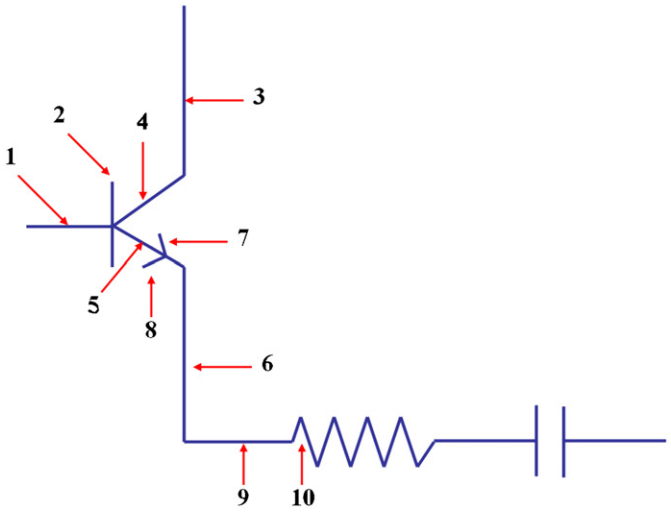

Fig. 1. A diagram illustrating interspersing: The user draws two other objects (wires made from primitives #3 and #6) over the course of drawing the transistor (primitives #2, #4, #5, #7, #8). Numbers indicate the primitive drawing order.

Another kind of temporal pattern present in online sketches is an *object-level pattern* that captures the probability of seeing a certain sequence of objects being drawn [6]. For example, when people draw box-connector diagrams (e.g., organizational charts, linked lists), boxes are typically drawn before connectors.

Our system learns both stroke-level and object-level temporal patterns of a domain from examples and uses them in recognition.

2.2.2. Handling multi-stroke objects and variations in encoding length

Users should be able to draw freely. For example, they should be able to draw a square using one, two, three or four strokes, or draw a resistor with different numbers of humps (thus generating encodings of the input with different numbers of observations). We achieve this by explicitly modeling whether the user has finished drawing an object.

2.2.3. Support for multiple drawing orders

We should be able to accommodate multiple drawing orders instead of just one. For example, it should be possible to draw a square starting with either horizontal or vertical lines. Furthermore, the system ought to learn about the user, learning for example whether the user uses one drawing order more frequently than others. This requires training and classification methods that can use such information. We achieve this by adopting a probabilistic machine learning framework where parameters are estimated to capture the statistics of the training data.

2.2.4. Probabilistic matching score

We would like the result of matching an observation sequence against a model to be a continuous value reflecting the likelihood of using that particular drawing order for drawing that object. This is required if we are to have a mathematically sound framework for combining the outputs of multiple matching operations for scenes with multiple objects such that, among plausible interpretations, those corresponding to more frequently used orders

[1] Because our domain does not have objects with curves, we work only with line segments. However, our model is general, and supports features computed from any kind of primitive.