EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2007)
M. van de Panne, E. Saund (Editors)

A Combinatorial Approach to Multi-Domain Sketch Recognition

A. Hall[1], C. Pomm[2] and P. Widmayer[2]

[1]Departement EECS, UC Berkeley, USA
[2]Institute of Theoretical Computer Science, ETH Zurich, Switzerland

Abstract

In this paper we propose a combinatorial model for sketch recognition. Two fundamental problems, the evaluation of individual symbols and the interpretation of a complete sketch scene possibly containing several symbols, are expressed as combinatorial optimization problems. We settle the computational complexity of the combinatorial problems and present a branch and bound algorithm for computing optimal symbol confidences. To handle sketch scenes in practice we propose a modest restriction of drawing freedom and present an algorithm which only needs to compute a polynomial number of symbol confidences.

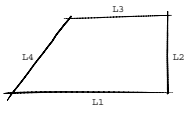
Categories and Subject Descriptors (according to ACM CCS): I.7.5 [Document and Text Processing]: Graphics Recognition and Interpretation

1. Introduction

The history of pen-based computer systems begins in the 60s when Ivan Sutherland presented SketchPad [Sut63]. Today the idea has materialized in devices with different form factors. The most prominent example is probably the Tablet PC. But they all have in common that pen-interaction should serve for better usability. For example, the computer should recognize handwritten text, sketches, diagrams, etc. This yields a real added value compared to traditional pen and paper interaction. In this paper we focus on the problem of recognizing multi-domain sketch scenes of hand-drawn symbols.

There has been a growing interest in this area, partly due to the now readily available pen-based hardware. Recently, several research groups have attempted to tackle the sketch recognition problem by following a similar basic line of approach [Alv04, HD03, HD05, SD05, SPRN02]: a *sketch domain* (e.g., user interfaces) is described by a set of *symbols* which may occur (e.g., button, text field, window). Each symbol can be seen as a template with placeholders for user drawn strokes. The placeholders are referred to as components and are usually associated with a certain geometric type, such as line, circle, or arc. Constraints describe the relationships between components, e.g., perpendicular, parallel, of equal length. The generic approach for multi-domain sketch recognition lends itself to the use of a language for describing symbols. For this purpose we developed the symbol description language SDL on the basis of existing languages. SDL makes moderate use of XML tags for better readability. As an example Figure 1 shows a possible SDL description for a perpendicular trapezoid.

The sketch recognition problem can be stated abstractly as follows. The input is given by a sketch domain (i.e., in our case a set of symbols described in SDL) and a sketch scene which consists of a set of strokes drawn by the user. A good interpretation is sought for such a sketch scene, i.e., a selection of symbols and an assignment of the strokes to these symbols. The obvious goal is that this interpretation reflects as closely as possible the symbols which the user intended to draw.




```
<symbol name="PTrapezoid">
   <components>
       L1, L2, L3, L4: Line
   </components>
   <constraints>
       Parallel(L1,L3)
       Perpendicular(L1,L2)
       PolyLineClosed(L1,L2,L3,L4)
   </constraints>
</symbol>
```

Figure 1: *Symbol* PTrapezoid*: Drawn example with components specified and related SDL description.*

Sketch-Based Interfaces and Modeling 2007, Riverside, CA, August 02-03, 2007.
© 2007 ACM 978-1-59593-913-5/07/0008 $5.00

Note that it is common to preprocess the user's pen input by *segmenting* it into strokes with associated geometric types (line, circle, arc, etc.). Applying this preprocessing is quite helpful, since it simplifies the problem without restricting the user. For example, a user may draw the four components for the symbol *PTrapezoid* in Figure 1 with one single pen movement, or with two movements, or with even more movements. The segmentation process will always provide four strokes representing the four components. In this paper we assume that the strokes have been segmented appropriately and focus on the sketch recognition problem.

From this common starting point various different directions have been taken in the literature. For instance, sketch recognition is considered in [SPRN02] as a statistical parsing problem or in [Alv04] as a probabilistic inference problem based on dynamically constructed Bayesian networks. We propose to study sketch recognition as a combinatorial optimization problem. To the best of our knowledge this is a novelty and at first sight it may seem somewhat surprising, since recognition problems are usually tackled by methods developed in the areas of machine learning, computer vision, or artificial intelligence. But in fact, since finding an assignment of strokes to symbols is a purely combinatorial problem, the described setting lends itself to this approach. In order to consider it as an optimization problem we only need to define an appropriate objective function. This objective function must be able to assess the quality of any possible assignment in form of a *confidence value*. We derive such an objective function, motivating it heuristically.

In previous work often the descriptions of the solution approach, the actually considered search space and the objective function have been intermingled and not clearly separated. An explicitly stated and comparatively simple objective function within a concise model as ours has its merits. For one, the actual goal becomes more transparent. As an immediate consequence, it is easier to reason about the chances of success of various possible search methods. Moreover, this approach enables the quantitative comparison of the outcomes of completely different methods on the same scene.

Our contributions and outline. In Section 2 we give a brief overview of related work. This is followed by the definition of our combinatorial model in Section 3. The model is an abstract, mathematical description which forms the basis of a flexible multi-domain sketch recognition system. Two combinatorial problems arise: the optimal assignment of strokes to a single symbol, and the optimal assignment of strokes from a complete sketch scene to possibly many symbols. We discuss the complexity of these problems by identifying which variants are NP-hard and which are polynomially solvable. Furthermore, we propose algorithms for the most general versions (see Sections 4 and **??**). We solve the first problem called MAXSYMBCONF optimally with a branch and bound algorithm which in the worst case has exponential running time, but performs well in practice. Under a commonly made restriction of drawing freedom (strokes of one symbol must be drawn consecutively) we are able to give an optimal algorithm for MAXSCENECONF which only needs to invoke the algorithm for MAXSYMBCONF a polynomial number of times. The algorithms have been implemented and tested within a framework for sketch recognition called *SketchWork*. The easy to use and powerful symbol description language SDL allows the realization of sketch recognition applications in a variety of domains. A preliminary experimental analysis with promising results is presented in Section 6. Due to space limitations we omit proofs or discussions, e.g., on the combination of confidence values or on speeding-up techniques for branch and bound. All these details can be found in the PhD thesis [Pom06].

2. Related Work

Many sketch-based systems allow only gestures for simplicity. Such one-stroke symbols can be classified with the approach in [Rub91]. Rubine extracts features from training samples of a gesture and derives a statistics-based recognizer. He uses 13 features, e.g. the distance between first and last sampling point or the length of the bounding box diagonal. The approach is not suitable for multi-stroke symbols as they are not appropriately characterized by these features.

Segmentation is a well studied problem to normalize the input strokes for approaches which deal with multi-stroke symbols. Powerful approaches have been presented in the literature, see, e.g., [HSN04, ZSDL06]. Although we are aware that the quality of the segmentation process can heavily influence the usability of a sketch recognition system we assume that the strokes have been segmented appropriately and focus on the actual recognition step.

Multi-domain approaches generally make use of symbol description languages to enable a flexible specification of sketch domains. Pasternak [Pas94] proposes a description language to recognize CAD-drawings. Shilman et al. [SPRN02] use a small set of relations for statistical parsing of sketched scenes. Hammond and Davis [HD03, HD05] develop LADDER as a dedicated language for sketching. Our description language SDL was based on the primitives and constraints of LADDER.

Graph-based approaches are standard in the pattern recognition community. Naturally, these methods apply also for recognizing hand-drawn sketches. The idea is to encode geometry, topology or other features of a sketch scene with a graph. Then, recognizing a symbol means solving a subgraph-graph isomorphism problem. In [MF02, LMV01] such approaches are presented. See also [LMV01] for a good overview of other graph matching methods, e.g. a linear programming approach [AD93]. [SPRN02] considers sketch recognition as a statistical parsing problem. Hierarchical symbols may be defined via a context free grammar. The basic shapes are one-stroke symbols which are recognized

with a method described in [Rub91]. Only a small set of constraints is supported and the system may not scale very well to scenes with a large number of strokes (> 500). A multi-domain sketch recognition framework including the symbol description language LADDER is described in [HD03]. The recognition itself is built upon the rule-based system *Jess*. Jess implements the so-called *Rete-Algorithm* which requires exponential running time and space in the worst case [For82]. In [HD05] it is stated that the system may slow down exponentially with an increasing number of shapes to be evaluated by Jess. Alvarado [Alv04] presents a framework for multi-domain sketch recognition which also uses LADDER. Input strokes are used to generate symbol hypotheses within a Bayesian network. The hypotheses with highest probabilities lead to an interpretation of the sketch scene. With a growing number of strokes the system does no longer reply in real-time. In [KS04] a hierarchical parsing approach is proposed. The mark-group-recognize strategy is conducted in three steps: First, certain delimiter patterns, so-called markers, are identified. Second, these markers are used for a spatial analysis of the sketch scene to cluster strokes. Third, each stroke cluster is processed separately and assigned to a symbol. To apply this method a sketch domain must have marker symbols which are easy to identify, e.g., arrows in a graph, or else recognition quality will suffer. In particular, it is not clear how stroke clustering works if we chose a sketch domain without markers.

The work in [SD05] is most closely related to our approach for interpreting a complete sketch scene as it makes use of the order in which a user draws the strokes of a symbol. A study has shown that many people draw a given symbol always in exactly the same way. This behavior is used to train Hidden Markov Models for recognizing sketches. The approach is polynomial in time. In contrast to our approach, a training phase is needed before the system can be used. Sharon and van de Panne [SvdP06] present a model to recognize sketches which only contain a single symbol. Motivated within a probabilistic framework they derive an objective function to perform a maximum likelihood search by branch and bound. In our model, we do not use a probabilistic background. Instead, we heuristically motivate our objective function for symbol evaluation and design it without biases towards symbols with few components or few constraints (see 3.3). This is important when considering symbol evaluation as a subproblem in scenes with many symbols.

Multi-domain sketch recognition approaches like above are difficult to compare in their performance because there is so far no standardized test corpus of annotated sketches as for pattern recognition; for a first attempt see [OAD04].

3. The Combinatorial Model

3.1. Strokes, Primitives, and Constraints

Pen-based systems use *sampling points* and *strokes* to represent the user's pen input. Sampling points discretize the user's pen movement. A stroke consists of a sequence of sampling points and represents the input from a single pen-down, pen-move and pen-up action of the user. Formally, a *sampling point* $\mathbf{a}$ is a triple (x,y,t) with real coordinates $x,y \in [0,1]$ and time component $t \in [0,\infty)$. A *stroke* $\mathbf{q}$ is a tuple $(\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_n)$ of $|\mathbf{q}| = n \in \mathbb{N}$ sampling points with $t_1 < t_2 < \cdots < t_n$. By STROKES we denote the set of all possible strokes.

In the introduction we have shown how a symbol is described by means of basic geometric elements and their relations to each other. These basic elements of a symbol are called *components*. Components may be chosen from a set of available *primitives*. The relations between components of a symbol are called *component constraints* and may be chosen from a set of available *constraints*. For a set of useful constraints see [HD03, Pom06]. For our model we need not fix which primitives and constraints we want to use in practice. All we need is an evaluation function for every primitive and every constraint. We use these functions to evaluate input strokes with respect to a given symbol and compute an overall confidence value. We have decided to choose confidence values from $[0,1]$ with value 1 as best possible evaluation.

A *primitive* p is determined by a function α_p which assigns to every stroke $\mathbf{q} \in$ STROKES a value from $[0,1]$. The function α_p is called *evaluation function* of p. By PRIMITIVES we denote the set of all possible primitives. It is up to a good system design to choose reasonable evaluation functions for the desired primitives. We only assume α_p to be efficiently computable. Typically, the function α_p measures the approximation error e between an idealized primitive shape and a given stroke.

The constraints of a symbol s serve to describe the relations between the components of s. For example, if a rectangle is to be defined then the pairs of opposite sides must be parallel. Thus, `Parallel` is a helpful constraint with two arguments. In general the number of arguments may range from 1, e.g. to determine an absolute length of a component, to m, e.g. to define a polyline. Let ρ be a m-ary function which assigns a value from $[0,1]$ to every m-tuple of strokes $(\mathbf{q}_1,\ldots,\mathbf{q}_m)$. The function ρ is called *constraint*. By CONSTR$[m]$ we denote the set $\{\rho : \text{STROKES}^m \to [0,1]\}$ of all possible constraints which accept an m-tuple of strokes as input. Again, we assume the constraints to be efficiently computable functions. Usually the functions evaluate basic geometric properties of the given strokes such that efficiency is not a problem.

3.2. Symbols

A symbol consists of all essential elements for evaluating a set of strokes which are distributed to its components. A *symbol* $s = \langle \mathbf{P},\mathbf{R}\rangle = \langle (p_1,\ldots,p_k),(r_1,\ldots,r_t)\rangle$ consists of $|s| = k \in \mathbb{N}$ *components* and $t \in \mathbb{N}_0$ *component constraints*. Component $i \in \{1,\ldots,k\}$ is of primitive type p_i. Each com-

ponent constraint r_i, $i \in \{1,\ldots,t\}$, is associated with a constraint ρ_i of cardinality m_i and a tuple $\mathbf{J_i} = (j_1,\ldots,j_{m_i}) \in \{1,\ldots,k\}^{m_i}$. We define r_i as follows: $r_i : \text{STROKES}^k \to [0,1]$, $r_i(\mathbf{q}_1,\ldots,\mathbf{q}_k) \mapsto \rho_i(\mathbf{q}_{j_1},\ldots,\mathbf{q}_{j_{m_i}})$. SYMBOLS is the set of all possible symbols.

3.3. Combining the Confidence Values

The basic idea of our generic approach is to measure properties of drawn objects and translate these measurements into an evaluation system. Our emphasis is to generate one confidence value from all measured symbol properties. Reasoning about a *scoring rule* which combines several confidence values, fuzzy logic and multi-criteria decision making suggest that an appropriate rule for our problem is the geometric mean – see, e.g., [FW00] for a discussion of different approaches. Alternatives like the arithmetic mean, the median, or probabilistically motivated approaches do not match our needs properly. In particular, outliers (e.g., one constraint with very low score) strongly influence the total confidence as compared to other types of means, which is desirable. In probabilistic approaches the score is usually computed as the product of all individual component and constraint scores. This introduces a bias towards symbols with few components and few constraints, which is avoided with the geometric mean. Further, we realize that a good balance of evaluating the components and evaluating the constraints is important. Hence, we determine scores from components and constraints separately and then compute from these scores the final confidence value.

3.3.1. Symbol Confidence

Given k user drawn strokes we want to evaluate the confidence of an assignment of these strokes to the components of a symbol s, with $|s| = k$. We describe such an assignment by a permutation $\pi \in S_k$, where S_k denotes the set of all permutations of $(1,\ldots,k)$.

Definition. *Let $s = \langle \mathbf{P}, \mathbf{R} \rangle$ be a symbol with k components. For a given set of strokes $Q = \{\mathbf{q}_1,\ldots,\mathbf{q}_k\}$ and for a given* assignment *$\pi \in S_k$ of strokes to components the expression*

$$\sqrt{\left[\prod_{i=1}^{k} \alpha_{p_i}(\mathbf{q}_{\pi(i)})\right]^{\frac{1}{k}} \cdot \left[\prod_{i=1}^{t} r_i(\mathbf{q}_{\pi(1)},\ldots,\mathbf{q}_{\pi(k)})\right]^{\frac{1}{t}}}$$

denotes the symbol confidence $\text{SymbConf}(s,Q,\pi)$.

From this definition we can derive a combinatorial problem where one aims to find the best possible confidence value for a given symbol and a set of given strokes.

Problem MAXSYMBCONF. *Given a symbol $s = \langle \mathbf{P}, \mathbf{R} \rangle$ with k components and a set Q of k strokes. Find an assignment $\pi \in S_k$, such that* $\text{SymbConf}(s,Q,\pi)$ *is maximized.* $\text{MAX}_{\text{Symb}}(s,Q)$ *denotes the maximum value.*

3.3.2. Scene Confidence

The interpretation of a sketch scene is based on the computation of symbol confidences. We regard the *scene confidence* of a sketch scene as a combination of maximum symbol confidences of possible symbols. A *sketch domain S* is a set of symbols, i.e. $S = \{s_1,\ldots,s_z\} \subset \text{SYMBOLS}$, $z \in \mathbb{N}$.

Definition. *Given a sketch domain S and a set of strokes Q. Further, given a* partition *$\mathcal{P} = Q_1,\ldots,Q_\ell$ of Q, i.e. $Q = Q_1 \dot\cup \ldots \dot\cup Q_\ell$, and a tuple of symbols $\mathbf{S} = (s_1,\ldots,s_\ell) \in S^\ell$. Let $|s_i| = |Q_i|$ hold for all symbols s_i with $i \in \{1,\ldots,\ell\}$. The* scene confidence *is defined as*

$$\text{SceneConf}(Q,\mathcal{P},\mathbf{S}) = \left[\prod_{i=1}^{\ell} \text{MAX}_{\text{Symb}}(s_i,Q_i)\right]^{\frac{1}{\ell}}.$$

The set Q is denoted as sketch scene*. The partition and the related tuple of symbols $\langle \mathcal{P}, \mathbf{S} \rangle$ is denoted as* interpretation *of the sketch scene.*

Based on the scene confidence we may formulate the interpretation of a sketch scene as follows.

Problem MAXSCENECONF. *Given a sketch domain S and a set of strokes Q. Find an interpretation $\langle \mathcal{P}, \mathbf{S} \rangle$ of Q, such that the scene confidence* $\text{SceneConf}(Q,\mathcal{P},\mathbf{S})$ *is maximized. The maximum value is denoted by* $\text{MAX}_{\text{Scene}}(Q)$.

4. Evaluation of Symbols

4.1. Complexity

We proved that MAXSYMBCONF is NP-hard and even inapproximable if $\mathsf{P} \neq \mathsf{NP}$, already for binary constraints. The basic idea is to reduce the well known NP-hard Hamiltonian circuit problem. For unary constraints there is a polynomial time algorithm based on weighted bipartite matching. But symbol descriptions with only unary constraints are very limited. A reasonable expressiveness of symbol descriptions is only assured if we use at least binary constraints.

Theorem 1 MAXSYMBCONF *is NP-hard and not approximable if $\mathsf{P} \neq \mathsf{NP}$, already for binary constraints.*

Theorem 2 MAXSYMBCONF *can be solved in polynomial time, if all constraints have only one argument.*

4.2. A Branch and Bound Algorithm

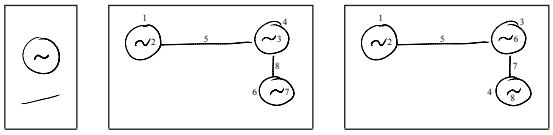
As the computation of the maximum symbol confidence is an NP-hard problem, we may not expect a polynomial-time algorithm. But for practical computation there are many better approaches than simply going through all possible assignments of strokes and components. In the following we propose a branch and bound algorithm which finds an optimal solution for MAXSYMBCONF. The general idea is to recursively construct partial solutions by assigning one component after the other. For each component we iterate over

the remaining strokes. We try assigning each of these strokes to the current component, evaluate the obtained partial solution, and if we cannot immediately discard this partial solution we recur.

Description of partial solutions. Let $\pi \in S_k$ be a permutation of k elements; π is a possible solution of MAXSYMBCONF. The set of all permutations which agree from the first up to the m-th position with permutation π is called a *partial solution* and denoted by $\pi|_m$. For the special case $m = 0$ we have $\pi|_0 = S_k$.

Branching. Let $\pi|_m$ be a partial solution (the first m positions are fixed). The BRANCH method partitions $\pi|_m$ into $k-m$ sets, say, $\tau_1|_{m+1}, \ldots, \tau_{k-m}|_{m+1}$. The $\tau_j|_{m+1}$ agree on the first m positions with $\pi|_m$ and are pairwise distinct on the $(m+1)$-th position. This position contains one of the remaining values $\{1,\ldots,k\} \setminus \{\pi(1),\ldots,\pi(m)\}$. In the corresponding search tree all permutations with all k positions fixed are leaves. An inner node v corresponds to the partial solution $\pi|_m$ if the nodes $v_{m-1},\ldots,v_0$ on the path from v to the root correspond to the partial solutions $\pi|_{m-1}, \ldots, \pi|_0$. If we traverse the tree and, e.g., visit a node on level 3, then the corresponding permutation has fixed its first 3 positions.

Bounding. For a partial solution $\pi|_m$ we have to estimate the value of SymbConf, although only the first m components have strokes assigned to them. We get a simple upper bound by assigning the value 1 to all evaluations which cannot be computed so far. We can do better though, by treating the two factors of SymbConf separately. The first factor $[\prod_{i=1}^{k} \alpha_{p_i}(\mathbf{q}_{\pi(i)})]^{1/k}$ concerns only the evaluation functions α_{p_i} of the primitives. Clearly, we know the correct values for the first m components. For the remaining $k-m$ components we at least know which of the strokes will be assigned to them. Thus, to obtain an upper bound for the first part of SymbConf we may compute on optimal assignment of remaining strokes to remaining components. Since each α_{p_i} takes only one stroke as parameter, this can be done in polynomial time by solving a derived weighted matching problem. For the second factor $[\prod_{i=1}^{t} r_i(\mathbf{q}_{\pi(1)},\ldots,\mathbf{q}_{\pi(k)})]^{1/t}$ of SymbConf the previous approach does not work since the component constraints usually are functions with more than one argument. Anyway, we can compute a bound which is better than the trivial one by bounding each component constraint r_i separately. If the number of unassigned arguments of r_i is less than some constant, we try all possible assignments for the remaining $k-m$ strokes. The largest resulting confidence value is our upper bound. For constraints with too many unassigned positions we can simply use 1 as upper bound. Constraints with no unassigned arguments can be evaluated directly. One can develop heuristics to accelerate the approach, e.g., it can be very helpful to choose a *good* order in which the individual components are assigned. Due to space limitations we refer to [Pom06] for details.



(a) (b) (c)

Figure 2: *(a) Sketch domain. (b),(c) Sketch scenes. The numbering resembles the drawing order. The symbols in (b) were drawn consecutively, but not so in (c).*

5. Interpretation of Sketch Scenes

The computation of maximum symbol confidences is required for finding the maximum scene confidence. Thus Theorem 1 implies that problem MAXSCENECONF is in general NP-hard. But even if MAXSCENECONF is considered independently from MAXSYMBCONF one can show that different versions of the problem are NP-hard.

Theorem 3 MAXSCENECONF *is NP-hard even if the sketch domain consists of only one symbol with a (polynomially computable) ternary constraint.*

Theorem 4 MAXSCENECONF *is NP-hard even if the contained symbols have only binary constraints and the confidences can be computed in polynomial time.*

On the positive side we have the following theorem for a very restricted version of MAXSCENECONF.

Theorem 5 *If the sketch domain only contains symbols with at most two components each,* MAXSCENECONF *can be computed in polynomial time.*

5.1. Consecutively Drawn Symbols

To overcome the problems that are imposed by the complexity of MAXSCENECONF we propose a modest restriction of drawing freedom: *Symbols may only consist of consecutively drawn strokes* (consecutively-drawn property). This means a symbol must have been drawn completely before the user starts to draw another symbol. But all the strokes belonging to one symbol may be drawn in arbitrary order – see Figure 2 for an example. In practice this restriction is not too hard for users but it simplifies the assignment of strokes to symbols dramatically.

Non-Overlapping in Time. If strokes $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k$ are ordered by time, we write $\mathbf{q}_1 \ll_t \mathbf{q}_2 \ll_t \ldots \ll_t \mathbf{q}_k$ meaning that $\mathbf{q}_1$ was drawn before stroke $\mathbf{q}_2$, etc. Our next step is to extend the notion of such non-overlapping strokes from a set of strokes to a partition of strokes.

Definition. *Let Q be a set of strokes of a sketch scene and let $\mathcal{P} = Q_1,\ldots,Q_\ell$ be a partition of Q. The partition $\mathcal{P}$ is called* non-overlapping in time *if there holds for all subsets Q_i and Q_j of $\mathcal{P}$ with $i < j$: $\mathbf{p} \ll_t \mathbf{q}$ for all strokes $\mathbf{p} \in Q_i$ and $\mathbf{q} \in Q_j$.*

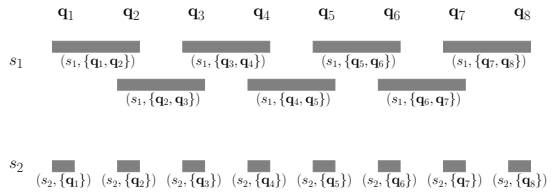
Simplification of MAXSCENECONF**.** Clearly, if we enforce the consecutively-drawn property, we only need to consider partitions that are non-overlapping in time. This yields the modified MAXSCENECONF$^{\text{consec}}$ problem: Given a sketch domain S and a set Q of strokes of a sketch scene. Find an interpretation $\langle \mathcal{P}, \mathbf{S} \rangle$ of Q with a partition $\mathcal{P} = Q_1, \ldots, Q_\ell$ of strokes from Q that is non-overlapping in time such that the scene confidence $\text{SceneConf}(Q, \mathcal{P}, \mathbf{S})$ is maximized. In this simplified version the partitioning is restricted to find *splitting points*, i.e. to find in a set of strokes $\mathbf{q}_1 \ll_t \mathbf{q}_2 \ll_t \ldots \ll_t \mathbf{q}_k$, which is ordered by time, the positions where one subset ends and the other subset starts.

5.2. A Polynomial Time Algorithm

The consecutively-drawn property allows for an efficient exhaustive search. We consider all *possible* assignments of strokes to available symbols and then find an assignment which maximizes the scene confidence.

Symbol candidates. We denote a set of strokes as a *symbol candidate* for a symbol s_i if these strokes possibly could represent the symbol s_i. In particular, a symbol candidate has to satisfy the consecutively-drawn property, i.e. a symbol candidate may consist only of consecutively drawn strokes. By (s_i, Q_i) we denote a symbol candidate which interprets the strokes $Q_i = \{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$ with symbol s_i. $\text{MAX}_{\text{Symb}}(s_i, Q_i)$ is the natural criterion to compare the confidence of the symbol candidate with other ones.

Example. We want to motivate our approach with an example. Suppose we are given a sketch domain $S = \{s_1, s_2\}$ with $|s_1| = 2$ and $|s_2| = 1$. A sketch scene for this domain is shown in Figure 2(b) where 8 Strokes have been drawn. The following schema depicts symbols, strokes, and possible symbol candidates (gray bars).



Certain symbol candidates exclude each other, e.g. $(s_1, \{\mathbf{q}_1, \mathbf{q}_2\})$ and at the same time $(s_1, \{\mathbf{q}_2, \mathbf{q}_3\})$ as symbol candidates are not possible. Else stroke $\mathbf{q}_2$ would have to be used for more than one symbol.

Scene graph. An algorithmic solution for our approach can be found by coding the above schema into a graph. We call this graph a *scene graph*. In a first step we translate the above schema directly into a graph and leave a more compact version to a second step. Additionally to the vertices which represent symbol candidates the scene graph has a start and an end vertex. The solution for MAXSCENECONF$^{\text{consec}}$ consists of using symbol confidences as edge weights in the scene graph and finding the symbol candidates which contribute to the overall best interpretation of a scene via a shortest path search – see the scene graph in Figure 3 and the corresponding sketch scene in Figure 2(b) for an example.

In general the scene graph $G = (V, E)$ is generated as follows. The set of vertices V consists of a start vertex, an end vertex and vertices for every possible symbol candidate for every symbol. The vertices are directed from *earlier* vertices to *later* vertices, i.e. there is an edge from v_1 to v_2 if the strokes of the symbol candidate belonging to v_1 were drawn earlier than the strokes of the symbol candidate belonging to v_2. The start vertex has only outgoing edges, the end vertex has only incoming edges. Edges from the start vertex connect to all vertices of symbol candidates which use the first stroke. If two symbol candidates which do not exclude each other use subsequently following strokes, e.g. like the candidates $(s_1, \{\mathbf{q}_1, \mathbf{q}_2\})$ and $(s_1, \{\mathbf{q}_3, \mathbf{q}_4\})$, then there is an edge from the vertex of the earlier candidate to the vertex of the latter one. All vertices of symbol candidates which use the last stroke are connected with the end vertex.

Algorithm. A scene graph G is directed and contains no cycles, i.e., G is a *directed acyclic graph* (DAG). On a DAG a shortest path can be computed very efficiently [CLRS01]. But shortest path algorithms minimize the sum of weights of used edges. To be able to use the algorithm, we transform the problem MAXSCENECONF$^{\text{consec}}$ into a minimization problem. Instead of maximizing the objective function $[\prod_{j=1}^{\ell} \text{MAX}_{\text{Symb}}(s_j, Q_j)]^{1/\ell}$ by taking the logarithm, we minimize a transformed objective function $\frac{1}{\ell} \sum_{j=1}^{\ell} (-\log \text{MAX}_{\text{Symb}}(s_j, Q_j))$. The transformed values are from the interval $[0, \infty)$ and assigned to the nodes as weights. Note that we avoid applying the log function on 0-valued symbol confidences by removing the corresponding node from the graph because it will not contribute to a reasonable interpretation of the scene.

More compact modeling. So far we have not concentrated on a representation of the scene graph which is particularly compact. The following version of the scene graph will use only $O(|Q|)$ vertices instead of $O(|S| \cdot |Q|)$ and only $O(|Q| \cdot |S|)$ edges instead of $O(|S| \cdot |Q| \cdot |S|)$. To achieve this we modify the role of vertices and edges. Transformed symbol confidence values and the related symbols are now coded on edges. Vertices represent the points of time between two strokes. Figure 4 shows the more compact version of the recent scene graph shown in Figure 3.

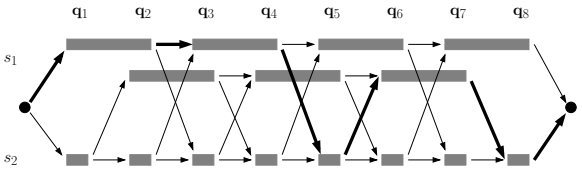


Figure 3: *Scene graph related to sketch from Figure 2(b). The bold path represents a possible interpretation.*

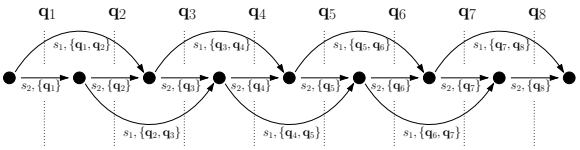


Figure 4: *Compact version of the scene graph for the sketch scene in Figure 2(b).*

For example, the edge with label $s_1, \{q_1, q_2\}$ represents the corresponding symbol candidate. The weight of this edge is the transformed symbol confidence of $s_1, \{q_1, q_2\}$. If there are more symbols which have the same number of components, the symbol candidate is chosen as label which has a maximum symbol confidence among all competing candidates. Now the scene graph consists of $|Q| + 1$ vertices, which is the start vertex v_0 and $|Q|$ further vertices v_i. The edges on the shortest path from v_0 to the end vertex represent an optimal solution of the sketch scene. If there are several optimal paths then an arbitrary path among all possible optimal interpretations is returned.

Theorem 6 *Given a sketch domain S and a sketch scene Q, the scene graph G of the sketch scene is a DAG. The shortest path from the start vertex of G to the end vertex corresponds to an interpretation which maximizes the scene confidence. G contains $O(|Q|)$ vertices and $O(|Q| \cdot |S|)$ edges. Hence the shortest path can be found in time $O(|Q| \cdot |S|)$.*

6. Experiments

We have implemented our model and the related algorithms in *SketchWork*, a framework that provides sketch recognition for pen-based applications. Our symbol description language *SDL* allows to define a broad range of symbols which the system can recognize. For a preliminary experimental analysis we have chosen *graphs* (4 symbols), *geometry* (20 symbols) and *buttons* (16 symbols) as domains – see Figure 5. Several users have contributed sketch scenes drawing the symbols according to the consecutively-drawn property. Additionally, the users were asked to draw each component with a single stroke because segmentation was not the focus of our experiments. In total we had 20 scenes for testing. We have labeled those scenes assigning stroke IDs and

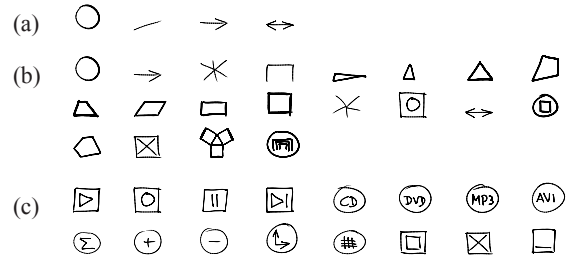


Figure 5: *Domains: (a) graphs, (b) geometry with special symbol* Tri-DoubleAngle *(last symbol), (c) buttons.*

symbol interpretations. We count a group of strokes as recognized if our labels agree with the recognizer's labels. This is a very simple but also strict scoring scheme because it will not account for related symbols that were misclassified, e.g. a square and a rectangle. We conducted our experiments on a Tablet PC with a 1.6 GHz Intel Centrino processor, 768 MB RAM, and Windows XP (Tablet PC Edition 2005). We are aware that our setting yields only a feeling for the systems's performance. For a complete evaluation a more extensive setup is needed, with more users, test-cases, and a comparison with existing approaches. Furthermore, it would be of interest to check the system's sensitivity towards the choice of various evaluation functions and to see how well it performs if segmentation is included as preprocessing.

6.1. Computing Maximum Symbol Confidences

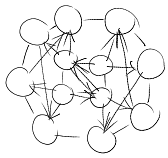
Figure 5 shows all symbols in the three test domains. For all of them the computation of the maximum symbol confidence happened in real-time visiting up to 10.000 nodes in the branch and bound tree (e.g. Square: 0.0012 – 0.0031 s; CheckBox: 0.0028 – 0.0061 s). The only exception was the symbol *Tri-DoubleAngle* (see Figure 5(b)) which was designed to challenge the algorithm. In this case the running time was between 0.41 and 9.76 seconds visiting between 3550 and 72195 nodes. The reason for that is the fact that we used repeatedly the ternary constraint `Opposite-Side(.,.,.)` to describe the symbol.

6.2. Interpretation of Sketch Scenes

Figure 6 lists the results of all 20 sketch scenes and shows three examples. Text labels are defined as one sort of primitives and thus integrate very well into our model. We used the Tablet PC's powerful handwriting recognition to interpret the labels. But we observed that the execution of handwriting recognition is quite expensive (c.f. graphs with/without labels and especially buttons). Focusing on graph scenes we notice an overall good recognition (∅82% for graphs without labels, ∅88% in the other case). Mainly the misinterpretation of vertices or edges as labels decreased the recognition rate. The running times are acceptable for interactive work (without labels: 44 to 158 strokes interpreted within 1.64 to 6.31 s; with labels: 19 to 64 strokes interpreted within 1.97 to 4.13 s), especially, when we take into account that we can adopt a scene graph for dynamic updates. Then, the response time after every stroke is only the time for computing the incremental update, e.g. for scene GL8, roughly $4.138/64 \approx 0.06$ s.

7. Conclusion

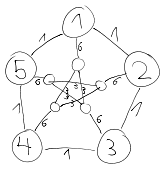
This paper proposes a new approach for recognizing sketch scenes. We presented a combinatorial model, optimization problems and corresponding algorithms to solve the recognition task. We have presented a new idea to both simplify



Scene: G3

Graphs without labels

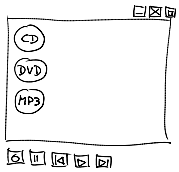
Q	$\|Q\|$	Time in s	Recogn.	in %
G1	44	2.595	5/44	34
G2	73	3.834	31/33	94
G3	70	3.067	36/38	95
G4	59	1.644	26/31	84
G5	158	6.315	69/80	86
G6	90	2.019	90/90	100



Scene: GL2

Graphs with labels

Q	$\|Q\|$	Time in s	Recogn.	in %
GL1	19	1.978	5/9	56
GL2	40	2.424	24/25	96
GL3	24	2.313	13/13	100
GL4	26	3.105	15/15	100
GL5	33	2.655	13/13	100
GL6	21	2.102	6/8	75
GL7	58	2.161	20/21	95
GL8	64	4.139	32/38	84



Scene: B1

Geometry

Q	$\|Q\|$	Time in s	Recogn.	in %
Geo1	80	2.464	18/20	90
Geo2	44	1.076	11/17	65
Geo3	20	0.562	5/7	71
Geo4	82	1.945	15/33	45

Buttons

Q	$\|Q\|$	Time in s	Recogn.	in %
B1	71	18.4 / 57.1	11/12	92
B2	78	15.7 / 86.6	16/18	89

Figure 6: *Examples of collected scenes; tables of running times and recognition results.* Graphs with labels: *Always with handwriting recognition.* Buttons: *First/second running time is without/with handwriting recognition.*

the recognition of sketch scenes and at the same time not restrict the user's drawing freedom too much. A more extensive experimental study is needed to thoroughly assess the practical merits of our approach.

Acknowledgments. We gratefully acknowledge the comments and pointers of careful anonymous reviewers.

References

[AD93] ALMOHAMAD H. A., DUFFUAA S. O.: A linear programming approach for the weighted graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 5 (1993), 522–525.

[Alv04] ALVARADO C.: *Multi-Domain Sketch Understanding*. PhD Thesis, MIT, 2004.

[CLRS01] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C.: *Introduction to Algorithms*, 2. ed. MIT Press, 2001.

[For82] FORGY C. L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence 19*, 1 (1982), 17–37.

[FW00] FAGIN R., WIMMERS E. L.: A formula for incorporating weights into scoring rules. *Theor. Comput. Sci. 239*, 2 (2000), 309–338.

[HD03] HAMMOND T., DAVIS R.: Ladder: A language to describe drawing, display, and editing in sketch recognition. In *IJCAI* (2003).

[HD05] HAMMOND T., DAVIS R.: Ladder, a sketching language for user interface developers. *Computers and Graphics 29* (2005), 518–532.

[HSN04] HSE H., SHILMAN M., NEWTON A. R.: Robust sketched symbol fragmentation using templates. In *IUI '04* (2004), pp. 156–160.

[KS04] KARA L. B., STAHOVICH T. F.: Hierarchical parsing and recognition of hand-sketched diagrams. In *UIST '04* (2004), ACM Press, pp. 13–22.

[LMV01] LLADOS J., MARTI E., VILLANUEVA J.: Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Trans. on Pattern Analysis and Machine Intell. 23*, 10 (2001), 1137–1143.

[MF02] MAHONEY J. V., FROMHERZ M. P. J.: Three main concerns in sketch recognition and an approach to addressing them. In *Proceedings of the AAAI Spring Symp. on Sketch Understanding* (2002), AAAI Press.

[OAD04] OLTMANS M., ALVARADO C., DAVIS R.: Etcha sketches: Lessons learned from collecting sketch data. In *AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural* (2004), AAAI Press.

[Pas94] PASTERNAK B.: Processing imprecise and structural distorted line drawings by an adaptable drawing interpretation kernel. In *IAPR Workshop on Document Analysis Systems* (1994), pp. 349–363.

[Pom06] POMM C.: *Ein kombinatorischer Ansatz zur Erkennung handgezeichneter Skizzen in Stift-basierten Anwendungen*. PhD Thesis, ETH Zurich, 2006.

[Rub91] RUBINE D.: Specifying gestures by example. In *SIGGRAPH '91* (1991), ACM Press, pp. 329–337.

[SD05] SEZGIN T. M., DAVIS R.: Hmm-based efficient sketch recognition. In *IUI '05* (New York, NY, USA, 2005), ACM Press, pp. 281–283.

[SPRN02] SHILMAN M., PASULA H., RUSSELL S., NEWTON R.: Statistical visual language models for ink parsing. In *AAAI Spring Symposium on Sketch Understanding* (2002), AAAI Press.

[Sut63] SUTHERLAND I.: *Sketchpad: A Man Machine Graphical Communication System*. PhD Th., MIT, 1963.

[SvdP06] SHARON D., VAN DE PANNE M.: Constellation models for sketch recognition. In *Proceedings of the 2nd EG Workshop SBIM* (2006), pp. 19–26.

[ZSDL06] ZHANG X., SONG J., DAI G., LYU M.: Extraction of line segments and circular arcs from freehand strokes based on segmental homogeneity features. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 36*, 2 (April 2006), 300–311.