Adaptive Recognition of Implicit Structures in Human-Organized Layouts

Takeo Igarashi Satoshi Matsuoka
Dept. of Information Engineering, Univ. of Tokyo
takeo,matsu@ipl.t.u-tokyo.ac.jp

Toshiyuki Masui
SHARP Corporation
masui@bird.shpcsl.sharp.co.jp

Abstract

Card-handling using hypertext editor can be a powerful methodology for generation of ideas or understanding of complex problems. To support such activity, recognizing implicit structure in the arrangement of cards would be useful. But, because the structures to be recognized are by nature ambiguous and highly dependent on user-specific perception, it is difficult for conventional rule-based spatial parsing algorithm to achieve this task. We propose techniques for building spatial parser suitable for finding such ambiguous structures based on the mechanics of human perception. Moreover, our parser is adaptively customized to reflect a particular user's preferences through an interactive suggestion process, supported by application of a genetic algorithm.

1 Introduction

Card handling editors[17][9][4] help the users to arrange small cards on a display to support intelligent information management tasks. The cards themselves contain textual or graphical information, and their spatial relationships represent the semantic relationships among individual information. Applications include idea generation, personal information management, and virtual blackboard for group discussions. Layouts that users author in these editors are implicitly structured to reflect their corresponding organization of information: for example, in Figure 1a and in Figure 1b, users perceive three groups in the card distribution, implicating the division of the given information into three categories. In addition, in Figure 1a, users also see that the upper two groups are interrelated more closely than the one at the bottom. These structures are 'implicit' in that there are no explicit visual cue nor rigid layout constraints that explicitly determine their structures. Rather, they are determined by user's notions pertaining to 'loose' layout rules, such as geometric clustering etc., whose criteria differ among the different users. Thus, it would be highly beneficial if the card-handling editor had a parser that understood such user notions of implicit structures, recognizing them interactively during the editing process; for example, users will be saved from repetitive manipulations of the components of an implicit group. Moreover, the parser might discover the structure that the user might have missed.

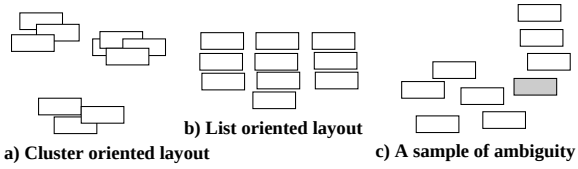


Figure 1: Samples of spatial layouts

However, there are several difficulties in the recognition of such implicit structures. First, these structures are ambiguous and idiosyncratic. For example, in Figure 1c, the membership of the gray card cannot be clearly decided to be the cluster on the left or the vertical list on the right. To find the optimal answer for such ambiguous cases, the parser must consider the surrounding context and evaluate the viability of multiple candidate structures.

Second, user's interpretation of spatial structure vary widely among the users[11]. One may prefer dense list-oriented layouts while another may prefer sparse cluster-oriented layouts, or even their mixture in hierarchically arranged diagrams. Thus, the parser must be able to adapt to the preferences of each user. Finally, the parser should be self-adaptive, since it would be difficult for the end-users to directly 'tweak' the recognizer in an intuitive way.

In this paper, we propose a visual parsing strategy for finding implicit structures in spatial layouts and describe our prototype system based on the strategy. We believe that our system can considerably increase the usability of card handling editors as well as other applications.

Our proposal consists of two parts: 1) the **Link Model** is a parsing strategy based on the observations of the human visual system. It recognizes implicit structures by calculating the strength of connections between objects. 2) *Automated parameter tuning* mechanism based on *Genetic Algorithm* facilitates the parser to learn from user's editing examples to adapt to his individual preferences. The prototype card-handling editor is shown to successfully recognize implicit structures in various layouts, and adapt to users via background processing.

2 Related Work

We owe much to Shipman *et al.*'s seminal works on recognition of structures in spatial hypertexts[16][10]. They investigated spatial layouts that appear in computational and non-computational environments, and developed a visual parser of implicit structures found in such spatial layouts. Our target diagrams are of similar class, and share much of their motivations. There are some key differences, however, and we discuss them in detail in Section 8.2

Research on visual parsing of visual languages[6], also deal with spatial layouts of objects, where many rule based parsing algorithms have been proposed including [3], [5], and [7]. Our work is different in that our intention is to extract emerging structure from free card layouts, whereas they intend to recover the structure of visual languages that was constructed based on a given grammar, which usually embody little ambiguity, and do not differ greatly among different users. Saund's perceptually supported sketch editor[15] is closer to our work—to allow users to access to the visually apparent structures in diagrams—although their target is pen-based sketches.

In addition, our work is relevant to cluster analysis[1]. But we can not directly use existing clustering algorithms, because they only consider distances between objects and ignore the relationships to their surrounding contexts. Ad-hoc extension of current clustering theories, such as mere extension of definition of 'distance', would not be sufficient to describe complex interactions between the links (described later).

3 How Do Humans Perceive Object Structures?

We first emphasize that we want the parser to recognize "what the human visual system perceives" from diagrams[15]. The majority of existing visual processing algorithms discover hidden but concrete structures that exist in the real world, such as constructing 3D models in 3D-vision recognition. There are little concrete structures in human organized layouts; instead, the structures emerge and exist only in the user's mind. Therefore, our approach is to recognize structures "as the humans do", considering the visual and perceptual models of human understanding of geometrical layouts.

According to Gestalt psychologists, what makes the human visual systems to perceive objects as collections are characteristics (called *Pragnanz*) such as proximity, similarity, regularity, and so forth[14]. For example, in figure 2a, a human sees three groups of boxes because the boxes in a group are located in mutual proximity and are distant from the members of other groups. This can be said to be the effect of proximity. On the other hand, in figure 2b, a group of four boxes aligned in a column may be recognized in which the deciding factor is regularity that are distinct from other objects. In the following discussions, we call the collections derived from proximity *clusters* and ones from regularity *lists*.

We next focus on the bottom-up nature of human perception. There, objects exist first and structures emerge from the arrangement of objects, not vice-versa. For example, in Figure 2b, one does not initially assume the existence of lists and search the layout for list structures (in a sense this is the rule-based parsers' scheme). Instead, one first perceives regularity in a local region of a layout and then the list structure is conceived from the characteristics of the regularity.

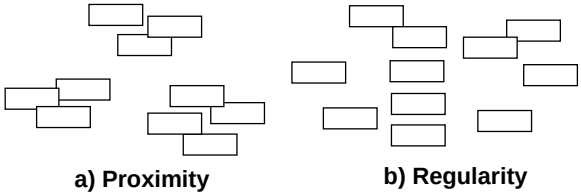
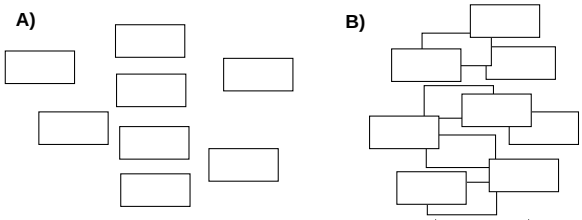


Figure 2: Characteristics of human perception



Identical arrangements exist in both **A)** and **B)**. One can see that it is impossible to decide the structure without considering the surrounding environment.

Figure 3: The need for "Looking Around"

An inevitable problem in any recognition is ambiguity among multiple possibilities of interpretations. For instance, in Figure 1c, one may interpret the gray object to be a member of the vertical list to the right or that of the cluster to the left. We regard the process of selecting the most appropriate interpretation as the result of *interactions* among interpretations. In Figure 1c, the two interpretations conflict, and the stronger one survives. As to which is stronger depends on the surrounding context (see below) and user preference. For this purpose, we must allow fuzzy interpretations such as "these objects seem to constitute lists rather than clusters," rather than distinct true/false assertion in traditional visual parsing algorithms.

Finally, human beings perceive objects depending not only on the local arrangements of objects but also on surrounding context of the objects. For instance, the four middle boxes are arranged identically in both Figure 3a and Figure 3b, but humans usually perceive them to be independent lists in Figure 3a while perceive only a single cluster of objects in Figure 3b. This observation indicates that parser must *look around* the surrounding contexts to recognize the appropriate structures.

4 Link Model—a visual parsing strategy

Based on these observations, we propose an alternative framework for the recognition of implicit structures called the **Link Model**. The general formulation of the **Link Model** is as follows. First, links are created between the adjacent objects. Second, the strengths of links are calculated through interactions among links. The strengths express how clearly the link's node objects appear to be connected to the user. Finally, the stronger links are selected and the objects connected by the selected links are grouped together.

By using the links, we simulate the bottom up parsing in the human visual system. Each structure emerges as the result of interactions among the links, allowing the parser to find structures even in somewhat disordered layouts. Then, we simulate the effect of regularity and proximity by strengthening links if the node objects are lined regularly or located near each other.

The problems of ambiguity and surrounding context are solved by the interactions among the links. Links represent interpretations such as "these objects should be included in a collection," and interactions among the links correspond to the interactions among the interpretations. Fuzziness is introduced by using a very simple fuzzy logic for the parameters of links. *Looking around* the surrounding context is achieved by the interactions and by selecting stronger links in comparison to the surrounding links.

4.1 Link Creation Process

To realize "to connect adjacent objects by links", we adopted the following strategy:

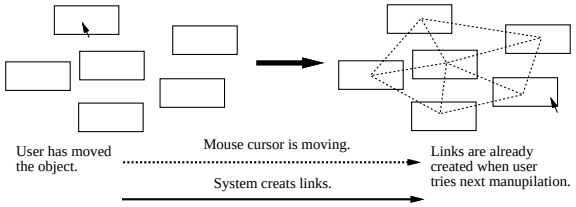
(1) Sort the objects by upper-left x-coords.

(2) List the candidate proximity objects for each object comparing x-coords of objects.

(3) For each object (which we call the *parent* object), the parser checks the objects in the list created in (2) and creates a link between the parent object and nearest object in each region. Eight regions (up, up-right, right, right-down...) cover the all the surrounding directions around the parent.

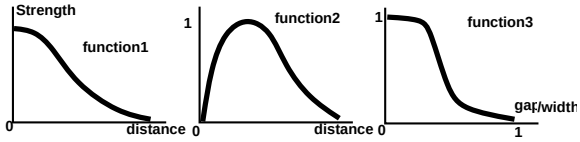
With this strategy, we can avoid wasteful calculations of checking all the combinations of objects. Experiments have shown that this strategy creates perceptually acceptable links(Figure 13). The resulting diagram is similar to Delaunay diagram (dual transformation of Voronoi diagram[13]), but our strategy is different in that our algorithm is based on the distances between rectangles whereas Delaunay diagram is based on the angles.

In our current implementation, this link creation process is invoked after every movement of objects in the editor, whereas the following recognition processes are called only when the implicit structure needs to be parsed by invoking *click selection*(see Section 6). By adopting this two-phased strategy, system can utilize the spare time between user's manipulations for adaptation, and the parsing delay after the click selection is significantly reduced(Figure 4).



System utilizes the mouse moving time to create links.

Figure 4: Mouse moving and Link creation.



Strength = function1 (distance)
List level = function2 (distance) X function3 (gap / width)

Figure 5: Calculations of strength and list level.

4.2 Strength Calculation Process

Initially, primary calculations are done for each link: they are *strength* value that indicates how close its node objects are and *list-level* value that refers to what extent the link can be considered as a list. The strength value is calculated as a function of the distance between the node objects and the list-level value is calculated as a function of the distance and the x-/y-gaps. These functions are initially defined as shown in Figure 5, but will be 'tuned' by the adaptation process in Section 5. In addition to these two values, we assign a *link type* to each link. Link type is defined as **vertical list** or **horizontal list** if the list-level value is not zero, and defined as **cluster** otherwise. We call the links with **list** type *list-links* and those with **cluster** type *cluster-links*. Lists are also categorized to be horizontal or vertical.

After the primary strength calculation, interactions between the links occur. The basic policy in designing the interaction algorithm is as follows.

- To strengthen the list-links to detect lists in the layouts as in Figure 2b.
- However, links should not be overemphasized as being list-links if the list is not distinct in a cluster, in order to avoid erroneous recognition as shown in Figure 6.

The interaction process proceeds as follows:

(1) In the first interaction process, each link's list-level value is reduced if it is inadequate to consider the link to be the a list-link, in order to prevent erroneous grouping as shown in Figure 6. Figure 7 shows the inadequate cases: in both cases, the thick links are initially considered to be list-links due to the spatial relation of their

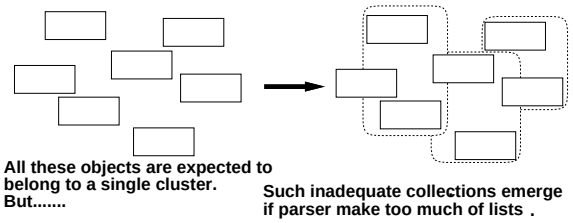


Figure 6: Example of an inadequate parse.

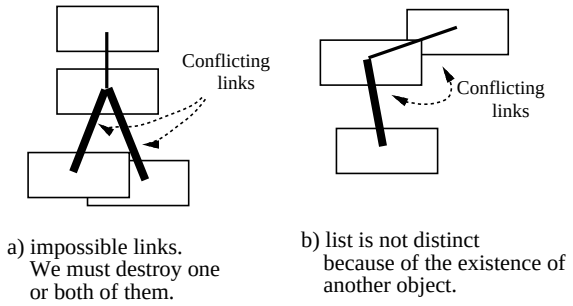


Figure 7: Example of inadequate list-links

node objects, but should not be perceived as list-links because of the interference of surrounding objects. As a result, the algorithm reduces the list-level value of these links by a parameterized factor.

(2) In the second interaction process, each link's strength value is re-calculated through *repression* and *reinforcement*. The primary purpose of this process is to detect the regularity in the arrangement as shown in Figure 2b. Actual definitions are as follows:

- The definition of final strength value for a cluster-link:

```
strength
× (1 + (cluster-repression-rate
        × interference))
```

- The definition of final strength value for a list-link:

```
strength
× (1 + list-reinforce-rate
       × list-level
       × (1 + interference-rate
              × interference))
```

Where `interference` is obtained in the process of the primary checking of adjacent links. The definition of `interference` is somewhat complicated to go over in detail, but the main scheme is shown in Figure 8: `interference` is given minus values by repressions by adjacent links of different types and is given positive values by reinforcements by adjacent links of the same type.

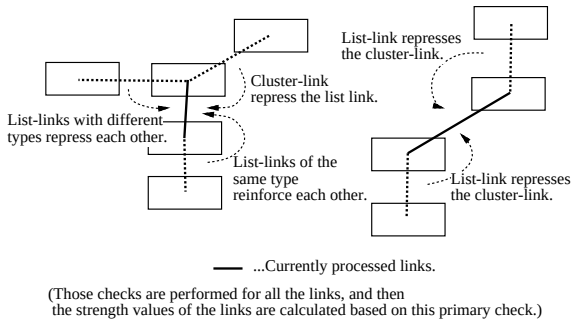


Figure 8: Interactions between links

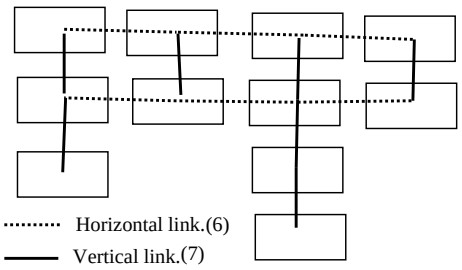


Figure 9: The supplemental procedure: enumerate the horizontal and vertical links and decides which is a substructure of the other.

4.3 Link Selection Process

After all the strength values are calculated, the stronger links are selected and the objects connected to these selected links are grouped together. The judgment whether the link is selected or not is done by comparing strength of links connected to each object. If certain links are considerably weaker than the other links connected to the object, the weaker links are eliminated. If there is no distinct difference of strength values among the links connected to an object, the decision is delayed. Such links will be eliminated if one of them is marked as weak at some other node. Here, the cutoff value is parameterized subject to adaptation.

In addition, there is a supplementary procedure at the end of the parsing process. This procedure decides which lists are primal in combined list collections, horizontal or vertical. In an arrangement like Figure 9, the algorithm cannot recognize that the diagram is a horizontal list consisting of vertical lists. Currently, we simply count the numbers of links belonging to each list type, and the type that has more links is selected as primal. For example, in Figure 9, there are 7 vertical list links vs 6 horizontal list links, so the system selects vertical list links as primal and the horizontal links as secondary.

Throughout these procedures, there are numerous fuzzy parameters that influences the outcome of the algorithms, such as the functions shown in Figure 5, control functions in the interactions, and cutoff value of the weak links in the selection, etc. These param-

eters are optimized for each user by the automatic tuning mechanism described in the next section.

5 Parameter Tuning with Interactive Genetic Algorithms

5.1 The Need for Adaptation

Adaptation in user interface is helpful in interacting efficiently with multiple users with wide-range of preferences. In card handling editors, a user may prefer list oriented layout whereas another may prefer stack oriented layout, etc. Moreover, the difficulty in the parsing of layouts is that different users can perceive different structure in an identical layout. Therefore adaptation is not only helpful but essential in finding desirable structures for each user.

Another requirement is self-adaptation. It is almost impossible to adapt parsing algorithms to each user by 'tweaking' the parameters by hand, because the algorithm is too complicated, embodying numerous interdependent parameters. Therefore, the parser should learn from user's manipulations and adapt itself automatically. Such automatic learning will free developers from the annoying burden of repetitive tuning (In fact, it is already very difficult for us to tune parameters by hand) and enables the on-line customization according to 'implicit' preferences of each user.

5.2 Interactive Adaptation and GA

As mentioned earlier, our parsing algorithm embodies numerous (currently thirty) parameters that decide the behavior of the algorithm, and adaptation is achieved by tuning these parameters. Adaptation is performed interactively. We constructed the tuning process as a background process that works independently from the main editor process and constantly receives the record of user's manipulations and returns the refined parameter set.

We used genetic algorithm(GA)[2] to achieve automatic parameter tuning instead of other numerical search algorithms for the following reasons: first, the search space of our problem contains many local solutions and it is difficult for heuristic algorithms like hill-climbing to find the best solution. Second, genetic algorithms are suitable for constant modifications of evaluation functions. In our system, the evaluation function changes constantly at every user's manipulation. Other search algorithms maintain only one solution and cannot respond well to such frequent changes of evaluation functions. Genetic algorithms can find the modified solution relatively fast because they maintain many chromosomes that cover the solution space.

5.3 Implementation of GA-based Parameter Tuning Mechanism

5.3.1 General Architecture

Before we describe the actual tuning process, we present the general architecture of the system as illustrated in Figure 10. If a user finds the recognized structure to be inadequate, he enters the modification mode. There, links are visually displayed and he can connect or disconnect the objects by clicking on the links (Figure 13). Upon leaving the modification mode, the result is passed to the tuning process as a new training sample. The tuning process then tries to find a temporary solution for the sample. It is 'temporal' in that it only satisfies the new sample and doesn't necessarily satisfy the entire set of training samples; nevertheless it is necessary to facilitate quick interactive response. The truly optimal solution for the entire sample set will be searched for later in general search routine. When a temporary solution is found, or a better solution is found in the general search, the new parameter set is passed to the editor and the editor immediately adopts the new parameter set.

5.3.2 Details of Tuning Algorithm

During the tuning process, the parameter sets are represented as chromosomes. The system maintains sixty chromosomes, each of which has a score that indicates how well the parser recognizes desired structures in the samples with the given parameter set of the chromosomes. The definition of the evaluation function is as follows.

```
1 - 0.5
  × ( number-of-missed-links
    + number-of-wrongly-selected-links)
  / number-of-desired-links
```

The system searches for the optimal chromosomes by creating new chromosomes through mutation and crossover as is with standard genetic algorithms. The difference is that our tuning process continues on repeatedly as user interaction proceeds.

The tuning process has three stages that executes in turn during user interactions. The general search routine searches for the optimal solution for the entire set of training samples. If an optimal solution is found, the maintenance routine starts, and the system prepares for the next sample by increasing the variety of chromosomes. If the system receives a new sample, the general search and maintenance routines stop, and the sub-search routine starts, which searches for the temporary solution as we have described. General search routine starts again after a temporary solution is found.

In practice, various deliberate strategies based on close observations of the problem domain are required to make these search algorithms work efficiently. Due to space limitations, we intend to describe the salient details of our search algorithms in another paper.

6 Prototype system

Our prototype system is written entirely in Scheme-Tk. Currently, the user can only manipulate the arrangement of cards on the display and is not allowed to write informations on the cards, but it is sufficient for testing the capabilities of our parser. The recognized structure is used to move, rearrange, and delete cards as a collection. Recognition process is called at user's request of performing these activities, and shows the recognized structure to user. If the result is not what the user intended, he can modify the recognition visually, and the modification is used to refine its param-

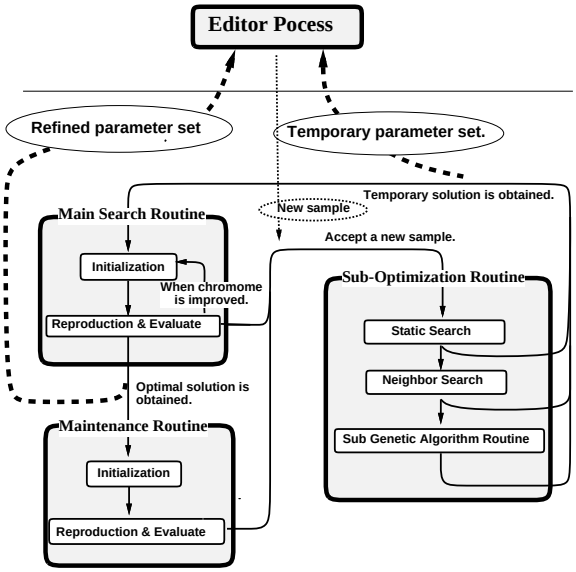


Figure 10: Overview of the automated parameter tuning mechanism.

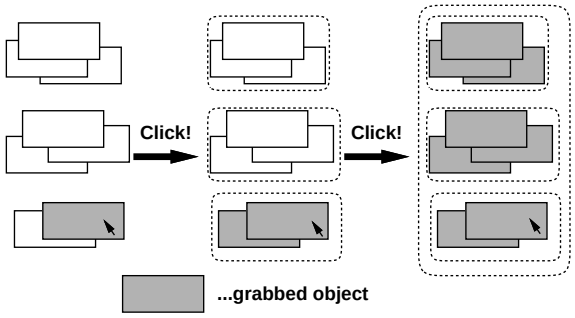


Figure 11: Description of hierarchical click selection.

eters of the parser. Parameter tuning process runs as background task and searches for the best parameter set while the user continues to use the editor.

In order to navigate through a hierarchy of recognized implicit structures, we employed multiple click selections as seen in text editors, where a single click selects a single object pointed by the mouse cursor, and double click selects the primary 'collection' that contains the object. Each successive click selection selects the next level of collection in the hierarchy. Figure 11 shows the example of such hierarchical click selections. Currently clicks are made on button 1 for selection and dragging, while button 2 invokes a menu bar, where the user can select several commands that work on collections: rearrangement, deletion, and modification(See Section 5.3.1).

We show brief examples using our system. Figure 12 shows a diagram created in our editor and the recognition that occurs with a triple click selection of button 1, plus a menu bar. Figure 13 shows the dragging of the recognized collection, and the modification mode, which appears when the user selects the modification command. All the links created in the link creation process are shown, and the thick links are the selected links. Users can toggle the selected links by clicking on them.

7 Experimental Results

Figures 14-16 show some examples of interactive recognition by our prototype. These layouts are recognized by the initial parameter set before adaptation. This shows that the perceptually desirable structures are parsed correctly by our **Link Model** for these simple examples with less ambiguity. Parsing time is a few seconds on a SparcStation 20, which allows the interactive use of the recognition mechanism.

In Figure 17 we illustrate the effect of parameter tuning. In this example, the parser fails to recognize the list structure, which is correctly recognized after proper adaptation. The temporary solution of adaptation is found in about thirty seconds and the global solution is found in about forty minutes. In this case, the training samples consist of 7 example layouts like the ones in Figure 12. Generally, temporary solutions are found in about several tens of seconds, but it currently takes orders of an hour to find global solutions for several training samples in our current implementation, which is based on the Scheme-Tk interpreter.

8 Discussions

8.1 Improving Adaptation Speed

As we have described, our parameter tuning process can find an optimal solution for a small number of samples within practical time with a combination of local and global GA-based optimization and background processing during user interaction. However, it is not fast enough to find a more general solution for a significant number of samples (hundreds or thousands), which may be required to represent users' preferences properly in all conceivable situations. Therefore, we must substantially improve the speed performance of our tuning mechanism. Employing compilers instead of Scheme-Tk will dramatically improve performance, but algorithmic improvement is also deemed essential.

8.2 Comparison with Related Work

Our *link model* is advantageous over conventional rule-based algorithms for recognizing implicit structures in that our parser can deal with ambiguous layouts more effectively. For example, finding lists in layouts like Figure 3a but *not* finding them in Figure 3b, would either be impossible or very difficult. It is also subject to easier adaptation, because one need not discretely add new rules, but rather can resort to parameter tuning which is a better-known optimization technique.

Compared to Shimpan et.al.'s work, the main differences stem from the slight differences in the target application. Shipman's algorithm facilitates more distinct *specialists*, each of which specializes in recognizing a certain class of structures, and parses the entire layout in batch, first by stochastically analyzing the

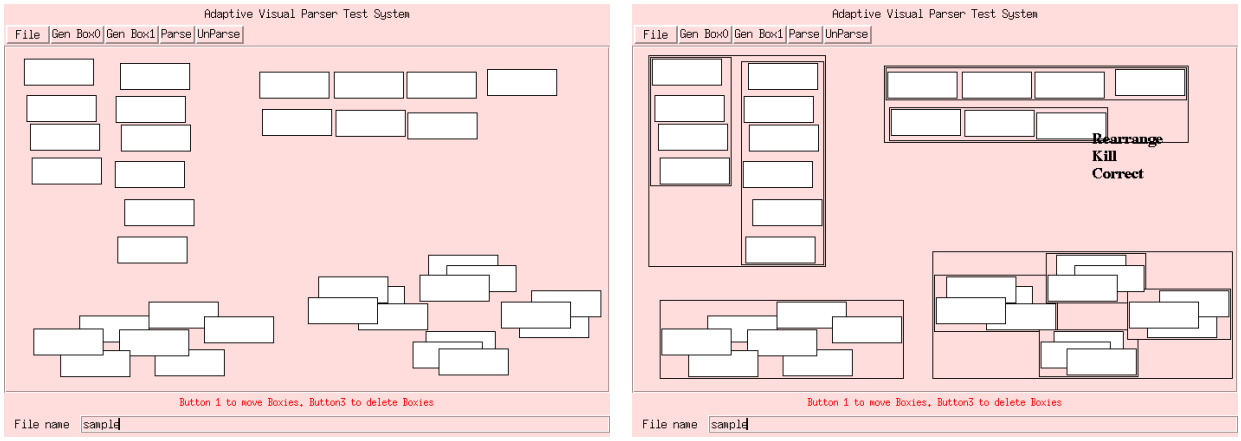


Figure 12: Examples of our prototype: diagram before parsing and the result of recognition.

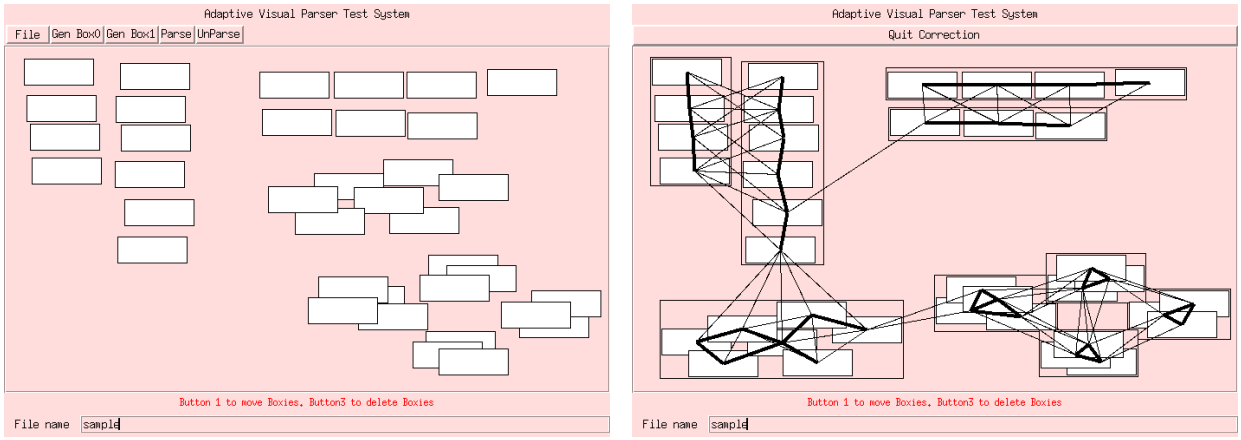


Figure 13: Examples of our prototype: dragging of a collection and modification mode.

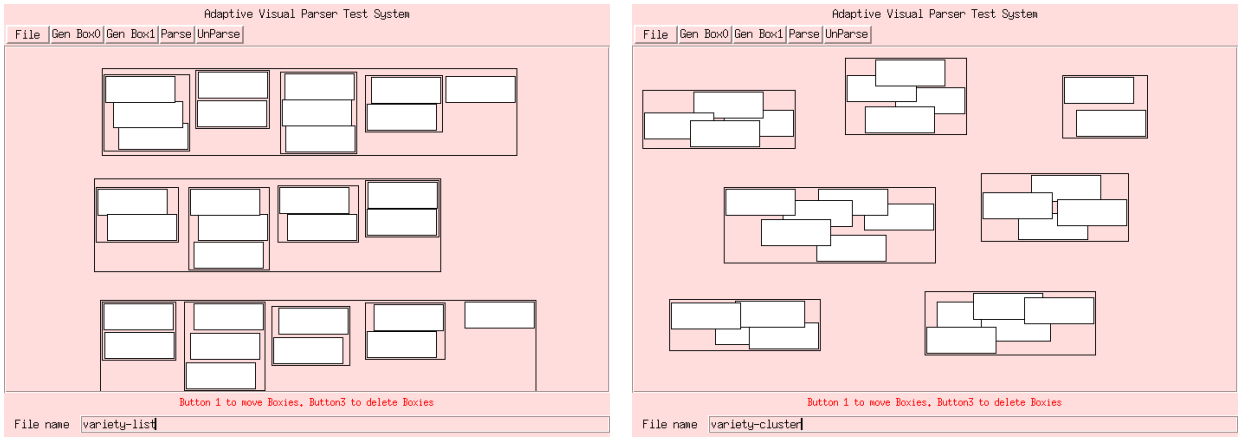


Figure 14: Examples: recognition of lists and clusters.

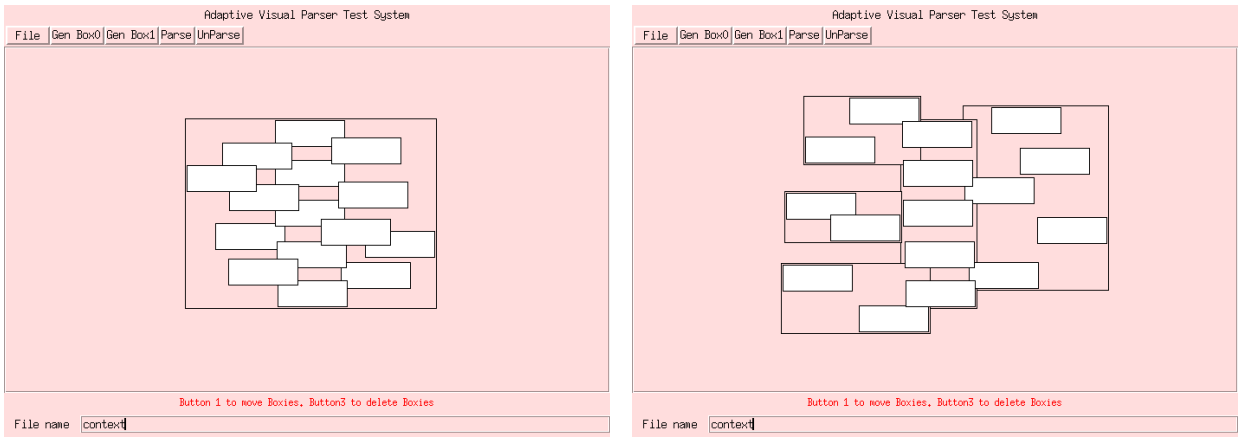


Figure 15: Examples: the effect of surrounding context

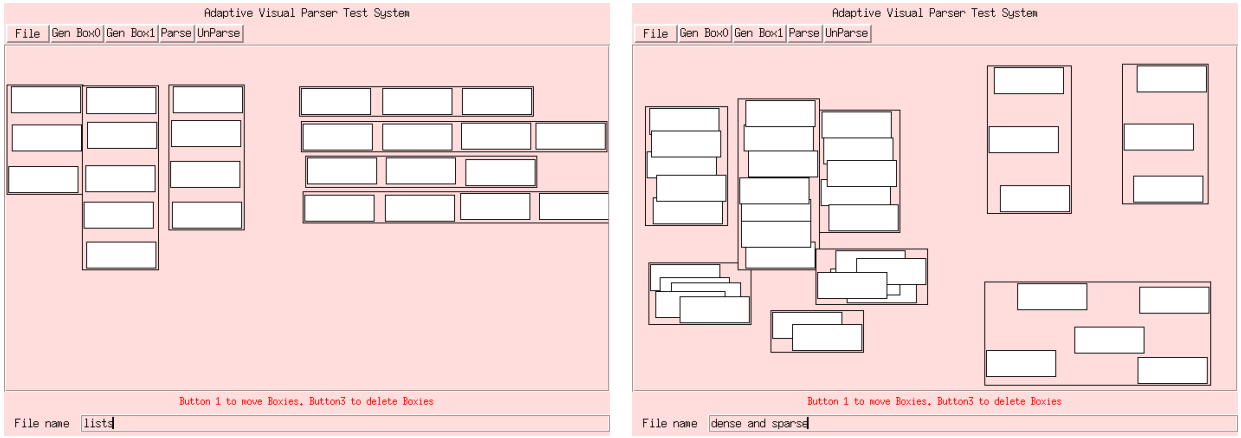


Figure 16: Examples: distinction between lists and a variety of layouts

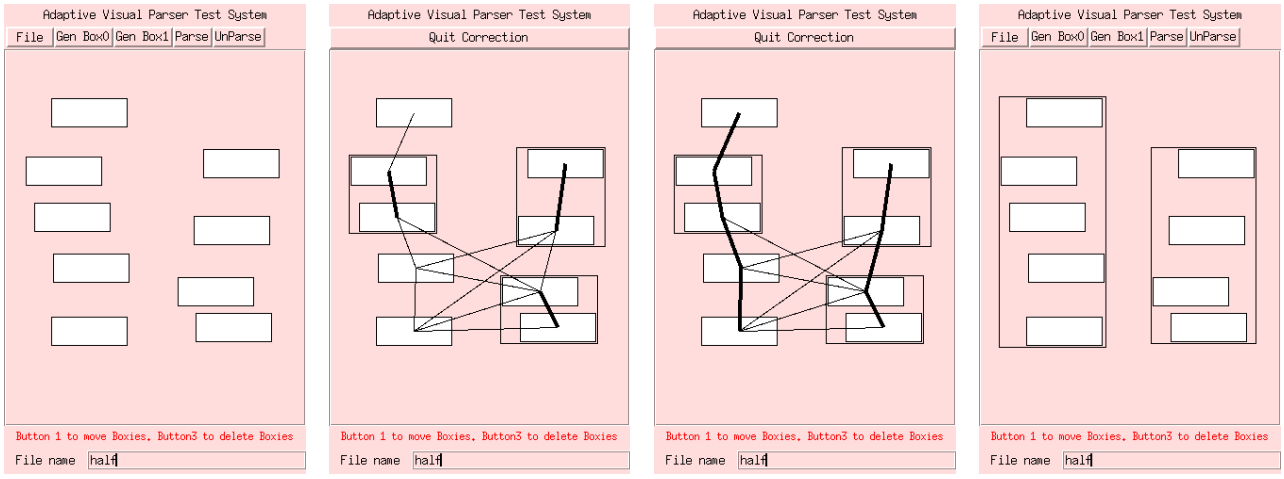


Figure 17: Examples: modification and adaptation.

entire diagram so as to decide the ordering of the application of specialists, and then applying each one in turn. As a result, Shipman's algorithm (1) recognizes more types of structures compared to ours, such as *stacks* and *composites* using predefined specialists, and (2) and is more suitable for diagrams that have already been systematically constructed to structurally represent user's knowledge.

By contrast, our algorithm targets more ambiguous layouts, aiding the users in real-time construction and manipulation of implicit structures. Aside from lists, we handle clusters which are not available in Shipman's system. Clusters can represent rougher categorization compared to more specific structures such as stacks, allowing the flexible use of layouts[8]. Also, because we have a bottom-up algorithm with high locality, we believe that our algorithm runs sufficiently faster compared to theirs, although no implications on speed have been given in their description. Such speed is necessary for direct application to interactive editing, which is rather a side-issue for Shipman's system, where the entire diagram is parsed in batch.

As a future work, it would be interesting to investigate the possibility of combining the benefits of both algorithms. Instead of simple list and cluster links, we could define the relationships between the parses of locally-applied specialists in Shipman's algorithm. The specialists will be parameterized and will mutually affect each other by repression or reinforcement. It is not clear whether such extension would be viable, but if successful, we would have the additional benefit of being able to effectively apply the specialists in different orders in different locations within the same diagram depending on their local layout context.

9 Conclusions

To enhance graphical human computer interactions, it is necessary for computer to understand the implicit user-created structures embodied in visual information. We have developed a parsing algorithm that recognizes the implicit structure in human organized spatial layouts, and applied it to a prototype card handling editor. The main idea is to construct a visual parser based on the observations on human perception mechanism, and furthermore have the parser be self-adapting to user preferences using genetic algorithm. We found that our **Link Model** is effective in recognizing ambiguous structures, and the parameter tuning mechanism achieved adaptation for a small number of samples.

There are many possibilities that lie ahead. In the parsing algorithm, we must extend the algorithms to handle other graphical cues such as the attributes of the objects. In the tuning algorithm, improvement in speed is necessary to realize practical adaptation. Besides these extensions and improvements, urgent work is to validate the utility and the effectiveness of our system with proper user-testing.

References

[1] Anderberg, M.R., "Cluster Analysis for applications", Academic Press, New York, 1973.

[2] Davis,L. "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York, 1991.

[3] Golin,E.J., "Parsing Visual Languages with Picture Layout Grammars", *Journal of Visual Languages and Computing*, no. 2, 1991, pp. 371-393.

[4] Kawai,K. *et al.*, "Preliminary Experiments with a Distributed and Networking Card-handling Tool Named KJ-Editor"(In Japanese), *Journal of Japan Society for Artificial Intelligence*, vol.8, no.5, 1993, pp. 583-592.

[5] Helm,R., Marriott,K., Odersky,M., "Building Visual Language Parsers", *Proc. of CHI'91*, 1991, pp. 105-112.

[6] Lakin,F., "Spatial Parsing for Visual Languages", In Visual Languages," Plenum, 1986, pp. 35-85.

[7] Maeda,T., "Grammar Customization for Adaptive Spatial Parsing"(in Japanese), *WISS'93*, Osaka, Japan, 1993, Kindai-Kagakusha, pp. 257-264.

[8] Mander,R., Salomon,G., Wong,Y.Y. "A 'Pile' Metaphor for supporting Organization of Information." *Proc. of CHI '92*, 1992, pp. 627-634.

[9] Marshall, C.C., Halasz, F.G., Rogers, R.A., Janssen, W.C., Jr. "Aquanet: a hypertext tool to hold your knowledge in place", *Proc. of Hypertext '91*, 1991.

[10] Marshall, C.C, Shipman, F.M. "Searching for the Missing Link: Discovering Implicit Structure in Spatial Hypertext", *Proc. of Hypertext '93*, 1993.

[11] Masui,T., "Graph Object Layout with Interactive Genetic Algorithms", *Proc. of the 1992 IEEE Workshop on Visual Language, IEEE Computer Society Press*, 1992, pp. 74-80.

[12] Montana,D., "Automated Parameter Tuning for Interpretation of Synthetic Images", Davis.L. "Handbook of Genetic Algorithms", Van Norstrand Reinhold, New York, 1991.

[13] Okabe,A., Boots,B., Sugihara,K., Spatial Tessellations—Concepts and Applications of Voronoi Diagrams, John Wiley, London, 1992.

[14] Roth,I., Frisby,J.P., "Perception and Representation : A Cognitive Approach", The Open University, 1986.

[15] Saund,E., Moran,T.P, "A Perceptually Supported Sketch Editor", *Proc.of UIST'94*, 1994, pp. 175-184.

[16] Shipman,F.M., Marshall,C.C., Moran,T.P., "Finding and Using Implicit Structure in Human Organized Spatial Layouts of Information", *Proc.of CHI'95*, 1995, pp. 346-353.

[17] Shipman,F.M., Marshall,C.C., Coombs,J.H, "VIKI: Spatial Hypertext Supporting Emergent Structure", *Proc.of ECHT'94*, 1994.