

From Sketch to Layout: Using Abstract Descriptions and Visual Properties to
Generate Page Layouts

by

Russell Lorance Greenlee

Bachelor of Industrial Design
Pratt Institute
Brooklyn, New York
1981

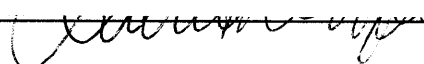
Submitted to the Media Arts and Sciences Section in Partial Fulfillment of the
Requirements of the Degree


MASTER OF SCIENCE IN VISUAL STUDIES
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September, 1988

© Massachusetts Institute of Technology, 1988, All right reserved

Signature of the author _____

Russell Lorance Greenlee
Media Arts and Sciences
August 5, 1988

Certified by _____

Muriel Cooper
Professor of Visual Studies
Thesis Supervisor

Accepted by _____

Stephen Benton
Chairman
Departmental Committee for Graduate Students

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

NOV 03 1988

ROLO
LIBRARIES

From Sketch to Layout: Using Abstract Descriptions and Visual Properties to Generate Page Layouts

by

Russell Lorance Greenlee

Submitted to the Media Arts and Sciences Section on August 5, 1988 in partial fulfillment of the requirements for the degree of Master of Science in Visual Studies.

Abstract

The knowledge used for solving page layout problems can be separated into two kinds, *visual knowledge*, and *content knowledge*. *Content knowledge* is concerned with the structure of the information on the page, while *visual knowledge* is concerned with the appearance of the information and of the page as a whole. Separating these two kinds of knowledge greatly facilitates the development of graphically adept systems.

The DAIS (Do As I Sketch) computer aided page layout system starts with a design grid, a set of layout elements (images or text), and a freehand *concept sketch* drawn by the user. The *concept sketch* shows the approximate locations and visual weights of the principal groups of layout elements. It is independent of the grid, and the number, sizes and shapes of the layout elements being used for a particular problem. This makes the *concept sketch* a very flexible means of defining a class of layout solutions. Potential layout solutions are generated by a controlled search of the design space defined by a small, extendable set of visual properties and relations. Their similarity to the *concept sketch* is evaluated using a grouping hierarchy representation, a matching procedure, and a static evaluation function.

This work was made possible by the generous support of IBM, NYNEX, and Hewlett Packard.

Thesis Supervisor: Muriel Cooper
Title: Professor of Visual Studies

ACKNOWLEDGEMENTS

I would like to thank the following people and organizations for their help and support:

My wife, Karli Hansen, for emotional support and encouragement, and for putting up with my long hours.

My parents and grandparents, for introducing me to the fascinating worlds of music, art, and science at an early age, and for all kinds of support throughout my educational career.

Muriel Cooper, for the opportunity to participate in the stimulating atmosphere at the Visible Language Workshop, and for helping me to focus my ideas and presentations.

Henry Lieberman, Ron MacNeil, Fanya Montalvo, and Patrick Purcell, for much useful criticism and feedback.

Suguru Ishizaki, Sylvain Morgaine, and David Small, for intellectual companionship, and for covering for me while I wrote this thesis.

Bob Sabiston, for the use of his Badwindows graphical interface software.

John Flight, for help with the *concept sketch* parsing software.

The Media Lab, for the chance to work at what must be one the most exciting places in the world.

IBM, and NYNEX, for generous financial support.

Hewlett Packard, for generous hardware and system software support.

TABLE OF CONTENTS

Abstract.....	2
ACKNOWLEDGEMENTS	3
LIST OF FIGURES.....	5
1. INTRODUCTION	6
Motivation	6
Graphic Design Knowledge	7
The Goal of the Research	12
Using DAIS.....	13
The Need for Abstraction.....	16
2. PROJECT DESCRIPTION.....	21
What it Does	21
Visual Knowledge	22
How DAIS Works.....	24
3. THE GROUP HIERARCHY.....	32
Parsing the Sketch.....	32
Grouping Hierarchies	34
Matching Abstraction Hierarchies	39
Comparing Concept and Layout Objects.....	40
Matching Compound Objects.....	41
4. THE LAYOUT GENERATOR.....	43
Plans (Sequencing).....	43
Detail Generators	44
Generator Filters	44
Search Control Strategies	45
5. RELATED WORK.....	50
6. CONCLUSION.....	54
Future Work.....	55
BIBLIOGRAPHY.....	57

LIST OF FIGURES

Figure 1. A page layout system with content knowledge and visual knowledge.....	10
Figure 2. The concept sketch.....	13
Figure 3. A car ad and its template.....	17
Figure 4. Graph illustrating the trade-offs between abstraction level and degree of flexibility for various graphical representations.....	18
Figure 5. Screen dumps showing a typical DAIS page layout problem.....	19
Figure 6. More final layouts from the problem in figure 5.....	20
Figure 7. The main modules of the page layout system.....	25
Figure 8. Abstraction hierarchy levels and their corresponding views.....	30
Figure 9. Parsing non-overlapping rectangles from a sketch.....	33
Figure 10. Approximation of negative space between rectangles.....	34
Figure 11. The most attracting elements on the page form a group.....	35
Figure 12. Grouping algorithm, step 1.....	36
Figure 13. Grouping algorithm, step 2.....	36
Figure 14. Grouping algorithm, steps 3 and 4.....	37
Figure 15. Grouping algorithm, step 5.....	37
Figure 16. The sensitivity problem.....	38
Figure 17. Matching nodes must have the same number of children.....	39
Figure 18. Matching elements must be similar in position and weight.....	40
Figure 19. A layout rectangle cannot be positioned so that it overlaps a concept rectangle by less than 50% of the layout rectangle's area.....	48

1. INTRODUCTION

This thesis is an investigation into computable models of basic *visual knowledge*, and how those models can be used to support computer aided page layout. This knowledge is tested in the DAIS (Do As I Sketch) page layout system. DAIS produces layout solutions from an abstract *concept sketch*. The rest of this chapter discusses the nature of graphic design knowledge and presents the goal of our research. Chapter 2, Project Description, gives an overview of what DAIS does and how it works. The algorithm for creating group hierarchies based on attraction is presented in chapter 3, The Group Hierarchy. The details of the layout generator are presented in chapter 4, The Layout Generator. Related work is reviewed in Chapter 5. Chapter 6, Conclusion, presents ideas for improving and extending DAIS.

Motivation

Automatic page layout is an important research topic for two practical reasons. First, there are many repetitive, routine page layout applications in the traditional print world today that lend themselves to automation. It would be desirable to automate the page layout tasks of publications that have a well defined, rigid format and that must be published in serial editions. Examples of repetitive, rigidly formatted publications are newspapers and catalogs. Second, there soon will be many applications in desktop and electronic publishing that will benefit greatly from automatic page layout capability. Access to encyclopedic quantities of information (both text and images) is becoming more affordable. But raw information is not useful

unless it can be formatted and presented to the user in a way that makes visual sense. In general the makers of such applications will not know the format of the display devices (hard or soft copy), or the structure of the information in advance. Automatic layout provides the capability to visually structure and organize text and images on the fly for electronic display or for hard copy.

Automatic page layout is a good domain in which to test graphic design knowledge representations. Page layout tasks range in difficulty from very easy to very hard to do automatically. For example, fitting a piece of text into a fixed size rectangle is relatively easy, while doing a fashion magazine layout that conveys a sophisticated, upscale image is very difficult to do automatically. This means that the power of graphic design theories and knowledge representations can be assessed according to the difficulty of the page layout problems they can solve.

Graphic Design Knowledge

The function of page layout and graphic design, is to effectively convey information or communicate. We are concerned with two basic kinds of knowledge that are used to solve page layout problems: *visual knowledge*, and *content knowledge*. *Content knowledge* is concerned with how the structure of information affects its arrangement on the page. The principle of keeping illustrations on the same page as the text that refers to them is a simple example of *content knowledge*. *Visual knowledge* is concerned with

the appearance of objects on the page and of the page as a whole. Principles of symmetry, balance, and color are all examples of *visual knowledge*. *Content knowledge* suggests semantic relationships that should be communicated in the layout, without suggesting how this communication is to occur. *Visual knowledge* provides the tools for communicating semantic relationships by creating and manipulating visual relationships between objects.

At first glance, the use of these two kinds of knowledge in page layout seems to be inextricably intertwined. It is reasonable to ask how these two kinds of knowledge might be separated, and what benefits we might derive from such a separation.

That the separability of *visual knowledge* and *content knowledge* is an established fact is not our claim. But we do believe that efforts to do so will be largely successful. This is already happening on a small scale. Document formatters that use tagged types (such as paragraph, heading, and footnote) and style specifications, such as SCRIBE [5.8], separate the format of a document from its content. Beach and Stone [2.1] carry this idea into the world of graphic illustration. Graphical style sheets store format information such as line weight, fill color, and font, separately from the content of the drawing. We test the separability idea by implementing a computer aided page layout system that separates the two kinds of knowledge. We do not claim that humans have separate *visual* and *content knowledge*, only that this separation is useful in a computer implementation.

Many visual properties and relations are clearly independent of content. Size, shape, and color, are such properties, and bigger than, darker than, and above are content free visual relations. Theories of color and composition have been developed in the design world [3.1, 3.2, 3.3, 3.5, 3.7]. More formal attempts to identify useful visual properties and relations are based on simple visual tests [4.1]. All of these are good sources of content free *visual knowledge*.

By keeping *visual knowledge* separate from *content knowledge*, to the extent that that is possible, we benefit in two ways. First, the *visual knowledge* that is used to solve problems in one application domain can be re-used in other domains, precisely because it is content free. Suppose we have knowledge about how to identify and group objects that are similar in size shape or color. This knowledge can be used in a drawing application to automatically group objects for the user. The same knowledge could be used in a page layout system to visually group layout elements with related content.

The second benefit is that implementing and maintaining both *content knowledge* and *visual knowledge* becomes easier. In the page layout example above we would have a content rule that says, "If a group of layout elements have similar content, then make them visually similar". How visual similarity between the elements is created is left up to the visually knowledgeable part of the system. If the *visual knowledge* and the *content knowledge* were not kept separate we could not write such a simple rule. Instead we would have to write a series of rules of the form, "If a group of

layout elements have similar content, then try to make them all similar in color", "If a group of layout elements have similar content, then try to make them all similar in shape", and so on. Now if we need to change our rule about similar content, we are forced to modify the whole series of rules.

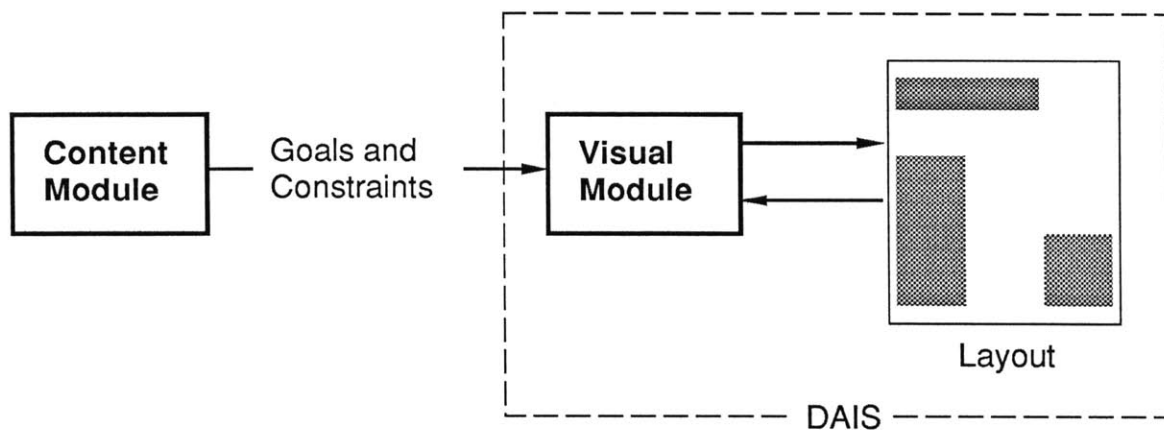


Figure 1. A page layout system with content knowledge and visual knowledge. DAIS is an experimental visual module.

The focus of this research is on *visual knowledge*, and its application to the page layout problem. DAIS is a prototype of a visually knowledgeable page layout system. It is not intended to be a fully functional page layout system. Instead its implementation will test our ideas about the separability of *visual* and *content knowledge*. Until *content knowledge* can be added to DAIS, we do not expect it will be a system of immediate use to graphic designers. A visually knowledgeable system would serve as basis for a more comprehensive page layout system that also includes *content knowledge*. In this comprehensive system, the *visual knowledge* resides in the visual

module, and the *content knowledge* resides in the content module (figure 1). Only the visual module has control of the layout. The content module affects the layout by communicating constraints and goals to the visual module.

The comprehensive page layout system we envision might work in the following way. The visual module would include a selection of techniques for raising the visual importance of an object on the page as part of its knowledge base. These might include changing the color of the object so that it is more eye catching, making the object larger, or placing the object in a prominent position on the page. Suppose the content knowledge module specifies that the visual importance of an object should be raised. The visual module changes the object's color to red. But the designer feels that red should not be used in this layout, so the designer overrides the system. The system responds by making the object larger. This satisfies both the designer's requirement and the content module's requirement.

The flexible interaction between the designer and the various parts of the system is greatly facilitated by the separation of *content* and *visual knowledge*. The requirements of the content module are satisfied because the visual importance of the object is high. Particular visual attributes of the object are irrelevant to the content module so long as the visual importance of the object is high. The designer is happy because the system can raise the visual importance of the object without changing its color to red. The separation of *visual* and *content knowledge* allows the system to be flexible and graphically adept.

The Goal of the Research

Our goal is to show that *visual knowledge* from the design theory literature can be identified and used effectively. While most of this design theory is not formal in the sense of being computable, it is possible to define computable models of the simpler of these representations. We believe that the power and flexibility that these models give us makes this well worth doing.

The goal of this research is to investigate *visual knowledge* representations that will support human/computer communication of flexible design concepts for page layouts. Our investigation takes place in three areas. The first is to identify likely visual properties and relations in the design theory literature. The second is to develop computable representations of these properties and relations, and the third is to test the power and utility of these representations by using them to solve visual page layout problems. The page layout problems are similar to the kind of abstract design problems that might be assigned to a first year student in form and composition class. A DAIS page layout problem can be stated as a design composition assignment: "Using value only (no color), arrange the five selected rectangles so that they form groups whose visual weights and locations are similar to those of the three elements in this concept sketch."

We demonstrate the effectiveness of our acquired *visual knowledge* by using it in the implementation of a page layout system. The goal of the DAIS system is to use a set of layout objects, a design grid, and a rough sketch of the

desired page, to produce a final layout having the same visual characteristics or structure as the sketch. In order to do this we will need to be able to interpret the user sketch, good graphical representations, and a good visual abstraction mechanism.

Using DAIS

We have chosen a page layout application where content is minimally important and form is maximally important. DAIS does layouts for one page photo essays.

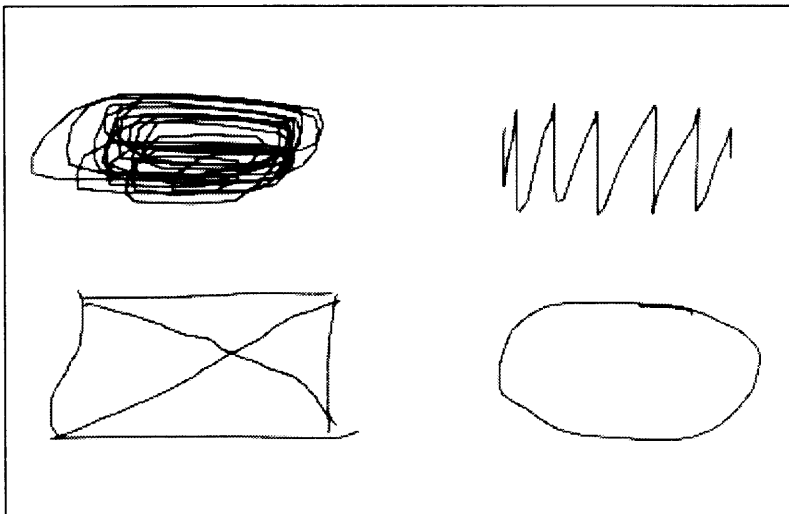


Figure 2. The concept sketch work area showing different kinds of sketching.

The DAIS user sets up a page layout problem by choosing a grid, selecting the layout elements, and drawing the *concept sketch*. The system uses a 6 column by 6 horizontal division grid (for double page spreads) as a

default. The grid defines the size and shape of the page, and restricts the shapes and locations of the layout elements. Layout elements (photographs in this case) are selected from a library of images available to the system. The system must use all of the selected images in the layouts it generates. The *concept sketch* is drawn freehand in a work space whose shape is proportional to that of the grid. The user scribbles rough rectangles to show the approximate locations and visual weights of the principal groups of layout elements (figure 2).

The system then produces a series of layout proposals using the selected images and the *concept sketch*. It is up to the system to place the images at grid positions so that groups of images correspond well with the principal groups shown in the *concept sketch* (figures 5 and 6, at the end of this chapter). The *concept sketch* is very flexible. The same *concept sketch* can be used with different grids, different sizes and shapes of layout elements, and different numbers of layout elements (provided there are at least as many layout elements as there are groups shown in the *concept sketch*).

The process of designing is partly one of making choices [3.8]. At each decision point there are alternatives. Each alternative leads to new choices, and each choice creates new alternatives. These choices and alternatives define a design space. The page layout problem can be characterized as a design space exploration problem. The graphic designer explores this space for a solution that satisfies the requirements of that problem (legibility, editorial content, rhythm, and so on) by making choices and choosing alternatives. Because this design space is very large and very complicated, it is impractical

for the designer to systematically explore all possibilities within the space. Instead the designer relies on past experience with similar problems, judgement, and intuition to make design choices that have a good chance of leading to an acceptable solution.

DAIS is a computer aided page layout system that helps the designer with this exploration process. While human designers are much better than computers at applying past experience and making design choices that are likely to lead to good solutions, computers are much better than humans at systematically exploring a portion of the design space. DAIS combines the strengths of human and computer by providing the ability to systematically explore the portions of the design space that are specified by the designer.

The human computer interaction style used by DAIS is modelled on relation between a senior designer and a design assistant. The senior designer does a rough freehand sketch showing the spatial arrangement of the page, and the design assistant works out detailed solutions based on the sketch. The role of the senior designer is played by the user, who creates *concept sketches*, while the role of the assistant is played by the computer. The computer explores the portion of the design space defined by the *concept sketch*. The *concept sketch* gives the designer control over the general layout of the page. The designer selects the most suitable layout solutions from those generate by the system.

Sketching provides a quick and powerful means of communicating visual ideas. Designers are already adept at sketching, so a page layout system that can accept sketching as input is easy and natural for designers to use.

The size of the design space is reduced to manageable proportions in two ways. First the use of a grid restricts the possible locations, sizes and shapes of the layout elements. Second, the possible locations for design elements is further restricted by the groupings shown in the *concept sketch*.

The Need for Abstraction

A page layout system that only has representations at the detail level forces the user to specify everything about the layout, in effect forcing the user to design the layout rather than specify the kind of layout. Pixel based representations are an extreme example of detail level representation. The user can only specify the color or value of pixels or groups of pixels. The user cannot make even simple specifications such as "the square at position x, y should be bigger". Object based representations allow the user to work at a level of abstraction that is independent of the pixel level by providing representations for geometrical objects. The user of such a system can work with these objects without having to be concerned with how they are rendered on a CRT, or printer.

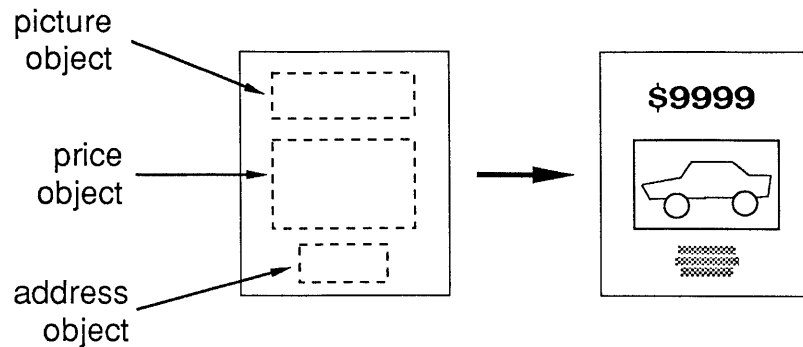
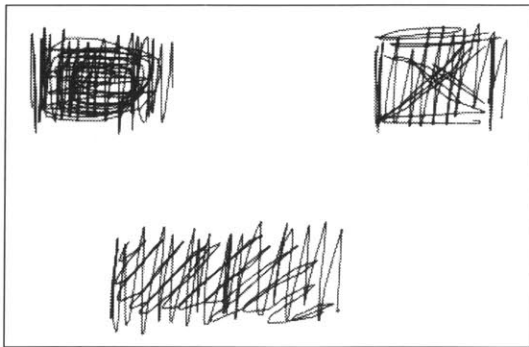


Figure 3. A car ad and its template.

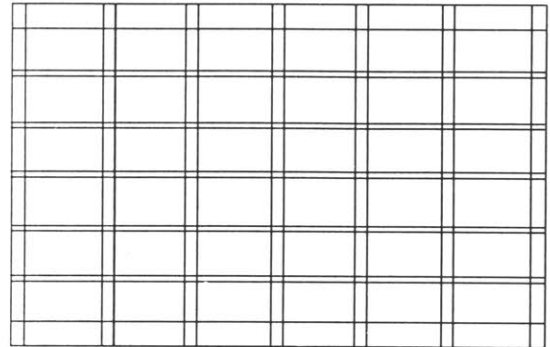
Template representations allow a further level of abstraction by allowing variables to stand for particular objects. The user specifies the attributes (location, size, shape, and so on) of an object slot, without having to specify the particular object that will be used. Variations on the design embodied by the template can be produced by filling the slots in the template with different objects. For example, a car ad for a newspaper might have several basic parts, a price, a picture, and an address (figure 3). A template for car ads would specify the locations for each of the parts. A specific ad is created by filling the the values of the price, picture, and address slots.

A variation on the template representation is the graphical style sheet proposed by Beach and Stone [2.1]. Instead of holding the object attributes constant and letting the objects vary, the style sheet holds the objects constant and varies the attributes.

DAIS *concept sketches* are more flexible than templates in two important ways. First, the elements of the *concept sketch* represent one or more layout elements, and the correspondence between the concept elements



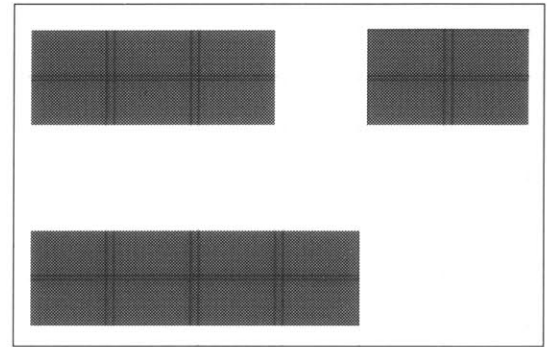
Concept Sketch



Grid



Layout Elements



Intermediate Layout

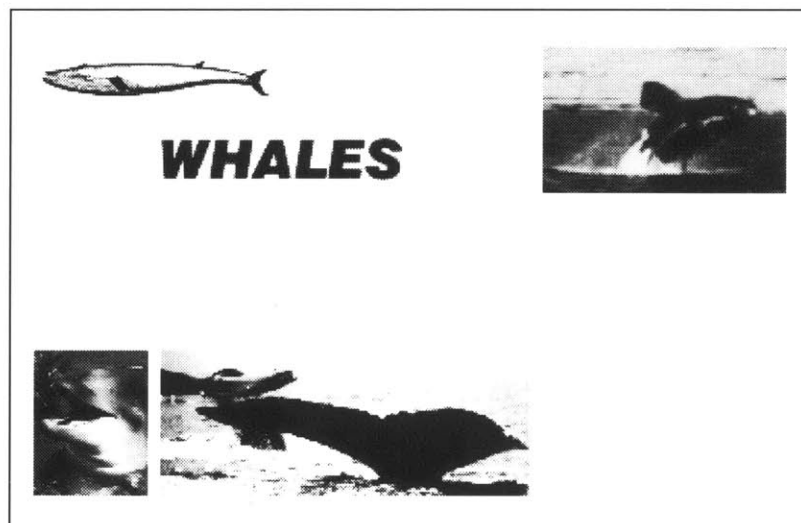


Figure 5. Screen dumps showing a typical DAIS page layout problem.

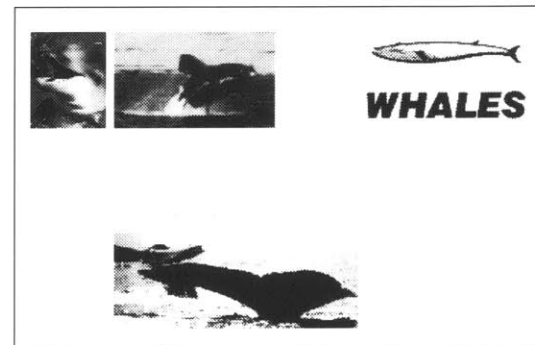
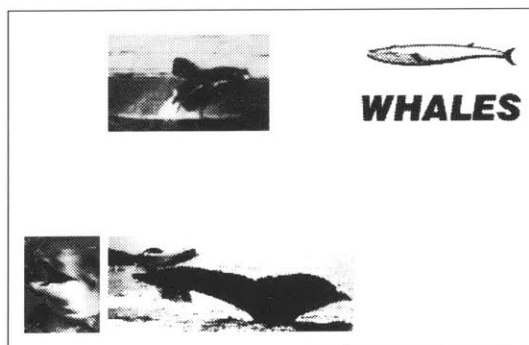
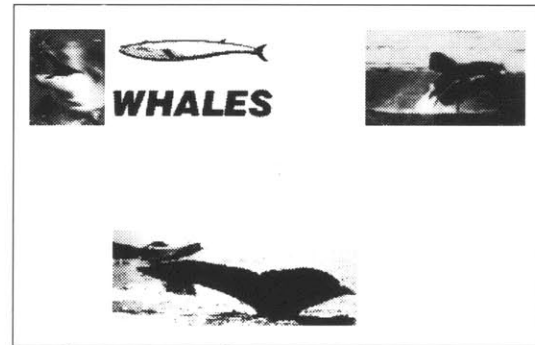
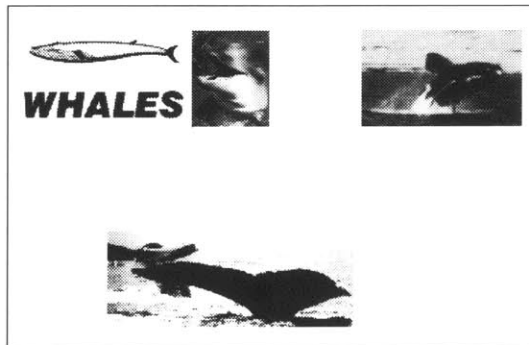
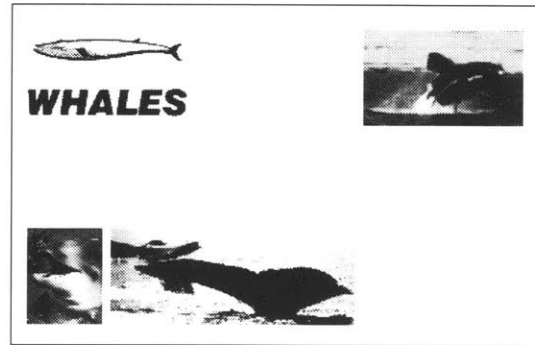
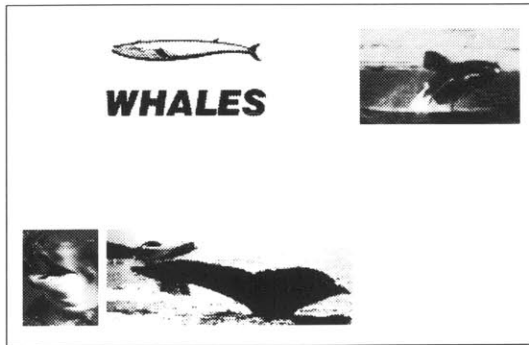


Figure 6. More final layouts from the problem in figure 5, previous page.

2. PROJECT DESCRIPTION

This chapter covers the implementation of the DAIS page layout system. The first section describes the operation of the system. The second section outlines the visual properties used by the system. The third section describes how DAIS works.

What it Does

DAIS generates page layouts from an abstract *concept sketch*, using a grid and a set of layout elements. The abstract concept is a freehand sketch, done by the designer, that shows the principal groups of layout elements. The user controls the visual composition (spatial arrangement) of the layout by means of a rough, freehand sketch (figure 5, at the end of the previous chapter). It is up to the system to find how to group the layout elements, and arrange them on the grid so that the main visual elements in the resulting layout have the same spatial arrangement as the elements of the sketch.

The *concept sketch* is more abstract than the generated layouts in two important ways. First, a visual element shown in the sketch can represent a single layout element, or a group of layout elements. This makes the *concept sketch* independent of the number of layout elements used for a particular problem, provided there are at least as many layout elements as there are sketch elements. Second, the *concept sketch* is independent of any particular grid. When doing the *concept sketch*, the designer need not be concerned with the size and shape of the layout elements, the number of layout elements, or the grid that is being used for a particular layout problem.

In order to keep the layout task from being overly complex, the type of layout that DAIS can generate is restricted in several ways. Layout elements are not allowed to overlap other elements. Layout elements must be rectangular. Other, more complicated shapes, like L shapes are not allowed. All edges of layout elements must lie on grid boundaries. Floating elements are not allowed. Many layouts used in magazine articles, and books adhere to these restrictions.

Visual Knowledge

A small number of visual properties and relations were selected from the art and design theory literature [3.2, 3.3, 3.7]. They are visual weight, balance, and attraction. Their selection was based on their usefulness for dealing with composition (spatial arrangement), and on their suitability for computational modelling.

Arnheim defines visual weight this way: "Physically and kinesthetically, weight is the effect of gravitational attraction. Visually, weight is the dynamic power inherent in an object by virtue of its conspicuousness, size, shape, location, etc." Two techniques were used to find a computable model for visual weight (and other properties and relations). First, we simplified the definition as much as possible. Second we defined visual properties and relations by analogy to physical ones. The idea that visual properties parallel physical properties is contained in Arnheim's definition where he compares visual weight to physical weight. For visual weight we

have simplified Arnheim's definition considerably. In the physical world an object's weight is a function of its volume and its density. In the two dimensional visual world we are only concerned with the visible attributes of the object, so volume corresponds to the visible portion of the object (its area), and density corresponds to its contrast with the background. We define visual weight as a function of the object's area and its contrast with the background.

Arnheim defines balance as "The dynamic state in which the forces constituting a visual configuration compensate for one another." We define the balance point of a group of objects by direct analogy to physics. The weight of an object is its visual weight, and its location is the position of its center point.

Attraction controls how objects group visually. The attraction between a pair of objects is based on similarity of size, shape, and texture, and by their proximity to each other [3.3]. We have defined attraction based on location only.

Each of these visual property models was informally tested on a small group of subjects. The results indicate that people see these properties in a consistent way. All of our test subjects saw large rectangles of a single value, as being visually heavier than small rectangles, for example. These visual properties interact with each other, so it is a challenge to design a visual test that isolates a single property. Montalvo [4.2] has done interesting work in this area using Bongard problems (a kind of visual puzzle). Our informal

testing indicates that further formal research into computational models of visual properties would be worth pursuing.

How DAIS Works

The system uses a generate and test strategy to produce page layout proposals based on a rough sketch supplied by the user. First, the layout generator produces a layout by searching the design space. This search is guided and controlled by the *concept sketch* and the grid. Once a layout has been produced by the layout generator, it is compared to the *concept sketch*. This is done by the concept to layout matcher. The matcher returns a score that is a measure of how well the layout corresponds to the *concept sketch*. If the matching score for the layout is not good enough (exceeds a threshold), the layout is rejected, otherwise it is presented to the user. Then the generator picks up where it left off to produce the next layout. This process continues until there are no more layouts to produce (the design space has been completely searched), or the user interrupts the system.

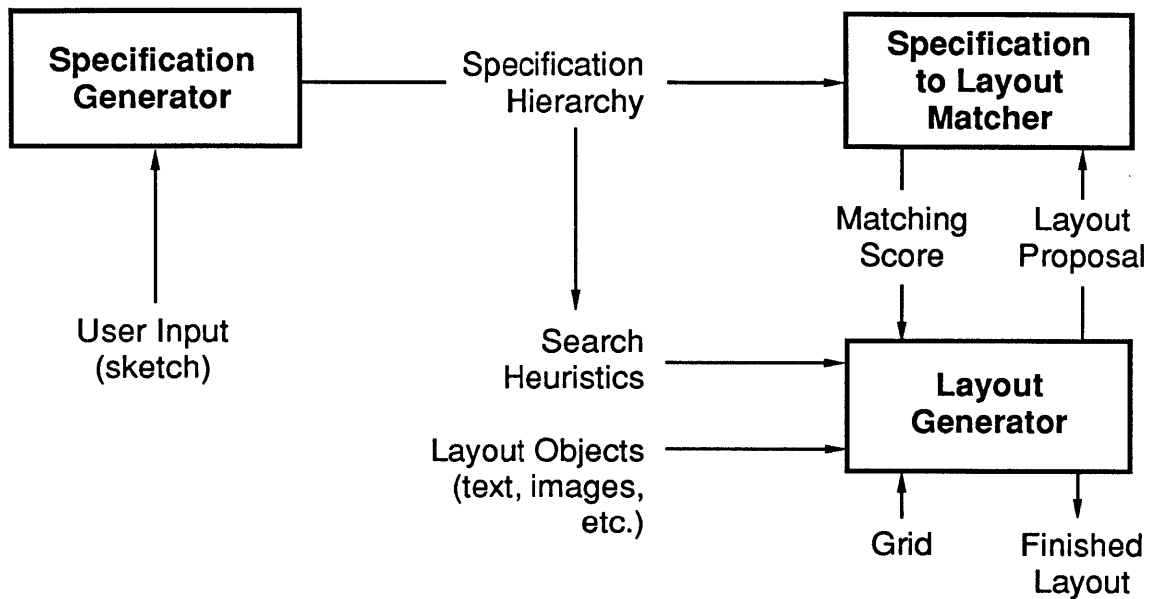


Figure 7. The main modules of the page layout system.

The DAIS page layout system has three parts: the concept generator, the layout generator, and the concept to layout matcher. The concept generator is responsible for parsing the sketch and setting up the concept hierarchy. The layout generator is responsible for producing layout alternatives that satisfy the matching criteria. The concept to layout matcher is responsible for setting up the layout hierarchy and matching it to the concept hierarchy (figure 7). The operation of these modules is described in the following sections.

From Sketch to Layout

The DAIS layout generator uses search to explore page layout design spaces. The search is controlled in two ways. The overall search strategy is one of successive refinement. This means that the generator first produces intermediate layout solutions that are then further refined until a proposal

can be produced (figure 5, at the end of the introduction chapter). The second control strategy uses visual properties (sizes, shapes, and positions) of the rough sketch to guide the placement of layout elements. The combination of these two control strategies greatly limits the amount of search necessary to find layout proposals.

Page Layout Search Spaces

The size of page layout search spaces is a function of the number of positions allowed by the grid, the number of objects in the arrangement, and the size and shapes of the objects in the layout. For example, suppose we have a 6 x 6 position grid, and we want to position six rectangles that each occupy one grid rectangle, with no overlaps. The first rectangle can be placed in any one of 36 positions, leading to 36 alternative partial layouts. In each of those alternative layouts, the second rectangle can be placed in any of 35 possible positions, leading to 36 * 35 alternatives, and so on. The number of possible layouts is

$$\text{total} = 36 * 35 * 34 * 33 * 32 * 31 = 1.4 * 10^9.$$

We can generalize from this example.

$$\text{total} = p! / (p - n)!, \text{ where } p \text{ is the number of grid positions, and } n \text{ is the number of } 1 \times 1 \text{ objects.}$$

If we increase the size of each of the rectangles in the previous example to occupy 3 grid rectangles we decrease the number of possible layouts by an order of magnitude.

total = 36 * 33 * 30 * 27 * 24 * 21 = 4.8 * 10⁸.

In general the size of the unrestricted search spaces for page layout problems is overwhelming.

It is worth noting that the use of the grid greatly decreases the size of the search spaces in the above examples. This is true because the grid severely limits the allowable positions for layout objects. Without the grid we could conceivably place layout elements in all perceptibly different positions on the page resulting in the number of layout possibilities increasing by many orders of magnitude.

If we restrict the number of positions allowed for each layout object, the number of layout alternatives decreases dramatically.

total = p^k , where p is the number of allowed positions, and k is the number of layout elements.

Allowing three positions for each of 6 elements results in only 729 layout alternatives.

The DAIS layout generator chooses positions for layout elements in relation to sketch elements. The number of positions allowed for each element varies from element to element and from problem to problem, but is always a small fraction of the total number of possible positions. In many instances there will be no allowable positions for an element. This has the desirable effect of pruning less promising branches from the search tree.

Comparing Layouts to the Sketch

The DAIS tester must be able to compare and evaluate the proposals produced by the generator, which are detailed and specific, to the sketch supplied by the user, which is general and abstract. This comparison of the detailed to the abstract is accomplished through the use of the grouping hierarchy. The grouping hierarchy allows the tester to ignore unwanted detail when comparing an abstract *concept sketch* to a detailed layout.

All layouts, pages, and objects are represented by group hierarchies. In the group hierarchy, an object is either a primitive object, or it is a compound object. A compound object is a group of objects. The grouping of objects is based on an attraction function. The attraction function determines the similarity of two objects according to some visual property of the objects, such as size or shape. We are using grouping by proximity which is a crude measure of the amount of negative space between two objects. Some other potentially useful ways of grouping objects are grouping by various alignments (vertical, or horizontal, by edges or centers), by similarity of proportion, size, or shape, and even by balance about some common axis. Whatever property we use to group objects, we can represent a layout or a page as a grouping hierarchy.

We will be looking at two types of group hierarchies. The first is the hierarchy that is used to represent the user supplied concept hierarchy. This serves as an abstract goal for the layout generator. The second type is the layout hierarchy actually produced by the layout generator. Both the concept

hierarchy and the layout hierarchy are syntactically identical, but are used in different ways by the layout system.

The matching of the concept and layout hierarchies happens after a layout proposal has been generated. This means that we can apply any needed application constraints or *content knowledge* in the layout generation phase. The matcher is only concerned with evaluating what is produced, not how it is produced.

The concept hierarchy is constructed from input (in the form of a rough sketch) from the user. The purpose of the concept hierarchy is to capture and convey the overall visual structure of the page. That is, the user specifies in a general way, without reference to the particular elements or constraints of the layout what he or she would like the system to produce, and the system attempts to produce a complete layout, using the particular elements and constraints of that problem which embodies the user's specification. The user concept conveys the following information: the number of major visual elements on the page and their grouping by proximity, the approximate position of the major visual elements, and the approximate visual weight of the major visual elements.

One of the primary characteristics of the concept hierarchy that distinguishes it from a layout hierarchy is that it is intended to represent an abstract class of layouts. The concept hierarchy is the "theme" from which the layout generator is to produce "variations". Therefore it is by definition more abstract or at least no more detailed than a layout hierarchy. A layout

hierarchy is constructed for each layout proposal produced by the layout generator. It conveys the full detail of a specific layout. Thus the concept hierarchy defines a class of layouts, and the layout hierarchy defines a particular layout.

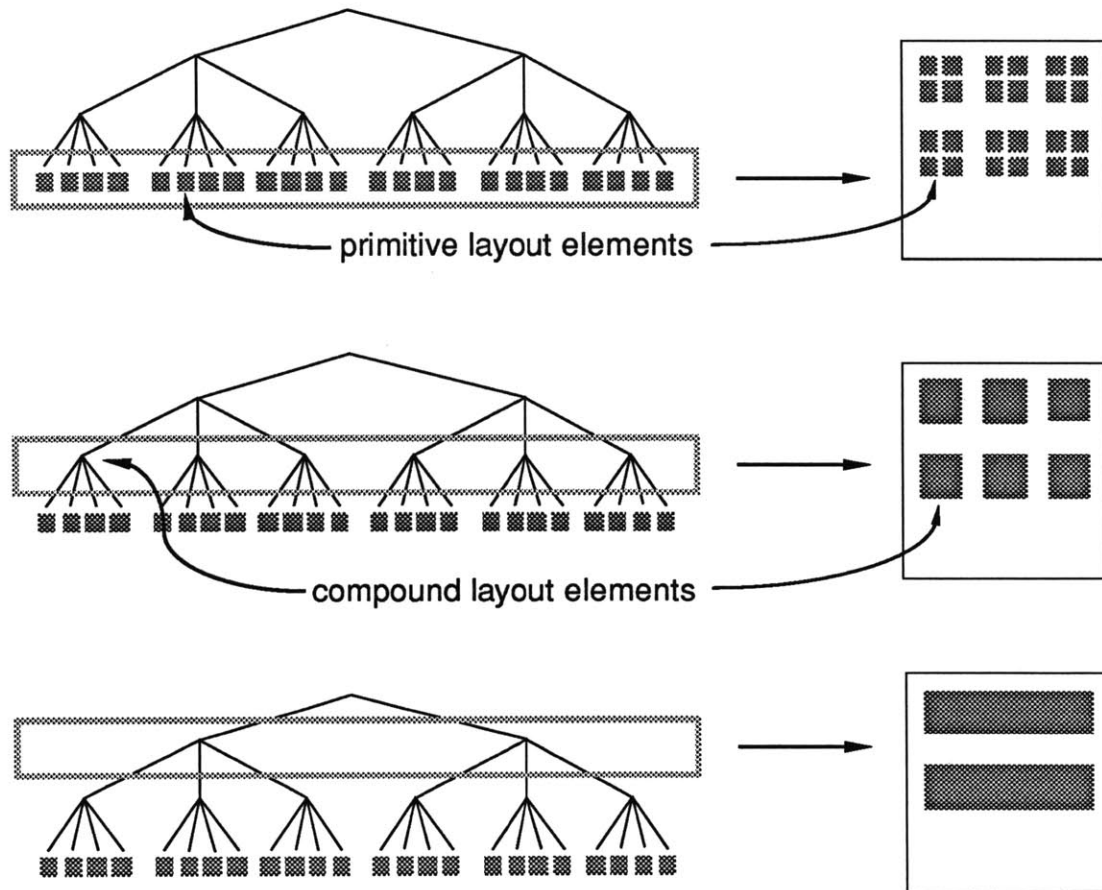


Figure 8. Abstraction hierarchy levels and their corresponding views.

The group hierarchy representation gives us a computable way of comparing any pair of objects or layouts. Because of the way the group hierarchies are defined, they give us a method for viewing objects at different

levels of abstraction. A cut near the top of the hierarchy gives us an abstract view, whereas a cut near the bottom of the hierarchy gives us a more detailed view of the object (figure 8).

It is this abstraction mechanism that allows us to use the overall visual structure conveyed by the *concept sketch* as an abstract concept for our layout system. The user provides the top levels of the grouping hierarchy, and the layout system "fills in" the lower levels using the actual elements of the particular problem. This allows the user to do the kind of task that humans do best (context sensitive conceptualization using common sense), and lets the layout system do the kind of tasks that computers do best (systematically exploring large numbers of possibilities).

3. THE GROUP HIERARCHY

The group hierarchy is the most important data structure used by DAIS. It provides multiple views of a layout or concept at different levels of abstraction. This makes it possible to compare the *concept sketch*, which is an abstract view of a class of layouts, to a final layout.

Parsing the Sketch

The *concept sketch* is created interactively by the user with a tablet and pen. The basic element of the sketch is a stroke. A stroke is defined as the path traced by the pen while the button is down. Points along the stroke path are sampled using the tablet and stored in an array. Strokes are represented on the CRT as a series of straight anti-aliased line segments connecting the sample points as the user draws. A sketch is a list of all the strokes drawn by the user during a sketch session.

The concept hierarchy is described in terms of rectangular areas on the page. These rectangular areas represent the main visual elements of the layout. It is the task of the sketch parser to analyze the list of strokes that describe the *concept sketch*, and return a list of non-overlapping rectangles suitable for use by the grouping algorithm. The DAIS user communicates the layout to the system using a formal visual language. The medium of this language is the freehand sketch. Engineering and architectural drawings are an example of formal visual languages. Fred Lakin has identified and implemented parsers for several of these languages [4.1]. The syntax of the visual language used by the sketch parser is designed to be simple to

implement and as flexible as possible. Any kind of stroke (curved, squiggly, or straight) or set of overlapping strokes is acceptable to the parser. Any stroke or group of strokes is interpreted as a rectangle. There are not syntax errors in this visual language. Therefore there can be no "wrong" input. This allows concept elements to be drawn in any style that suits the user.

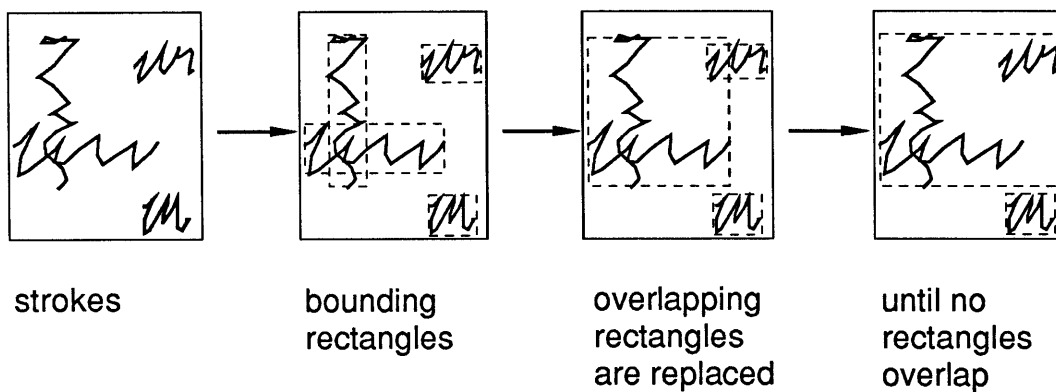


Figure 9. Parsing non-overlapping rectangles from a sketch.

The parser first finds the bounding rectangle of each stroke in the sketch and puts them in a list. Any group of mutually overlapping rectangles in the list are removed and replaced by one rectangle that is the bounding rectangle of the group it replaces. This process of finding all groups of mutually overlapping rectangles and replacing them with single rectangles is repeated until none of the rectangles in the list overlap (figure 9).

Grouping Hierarchies

Objects are grouped by visual attraction. Two objects are visually attracted to each other if they are close to each other, or similar in size and shape or color. For each of these visual properties we can define an attraction function that is the difference in that property's value for the two objects.

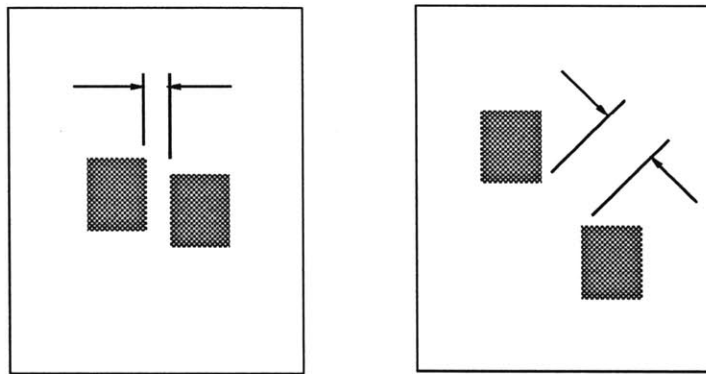


Figure 10. Approximation of negative space between rectangles.

DAIS defines the attraction function based on the distance between the nearest edges or corners of a pair rectangles (figure 10). This distance function is intended to approximate the amount of negative space between the rectangles.

The attraction based on distance is defined as follows:

$$A = 1.0 / \text{dist}^2, \text{ where dist is defined as above.}$$

This definition results in groupings that correspond well with the way humans group layouts.

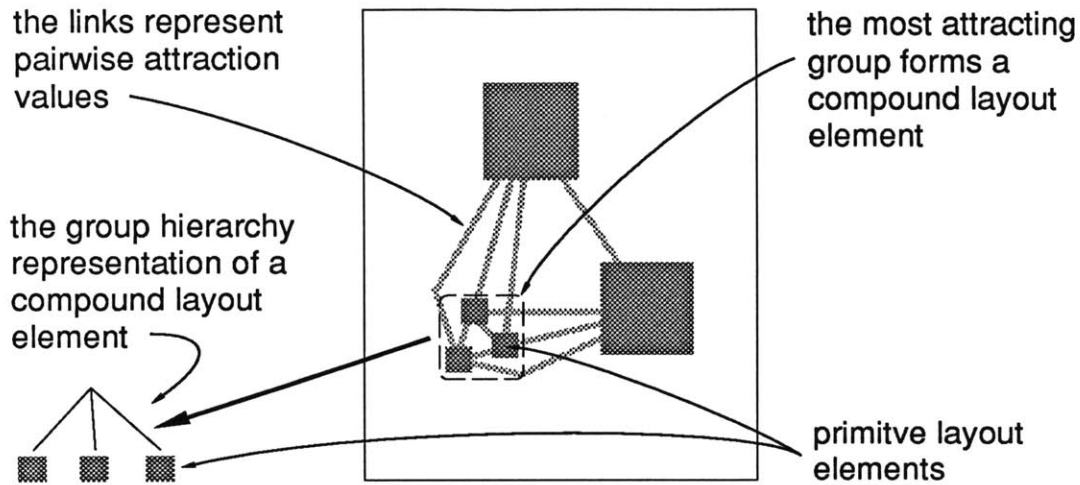


Figure 11. The most attracting elements on the page form a group.

Once we have a way of computing the attraction between a pair of objects, we need to be able to find groups of strongly attracting objects. This can be done by first computing the attraction between all pairs of objects, and then grouping pairs that are more strongly attracted to each other than to any of the other objects on the page. The overall strategy for doing this is to find the most strongly attracting groups on the page and treat those groups as one object (figure 11). The process is repeated recursively until no more strongly attracting groups are found, creating a hierarchy based on grouping by attraction from the bottom up.

The grouping algorithm follows:

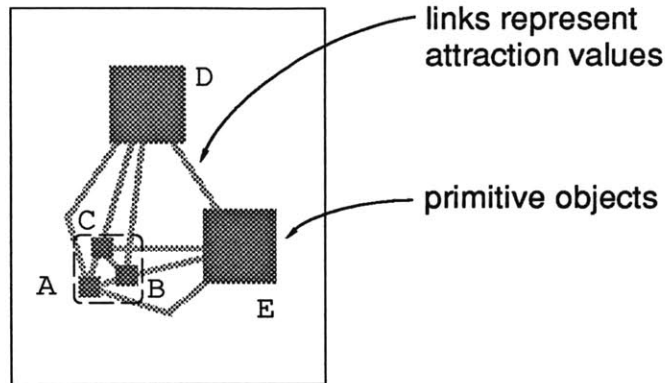


Figure 12.

1) Compute the attraction between every pair of objects on the page (figure 12).

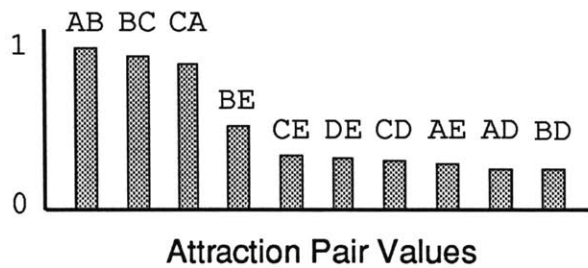


Figure 13.

2) Sort the values of the attraction pairs from greatest to least amount of attraction (figure 13).

3) Make a list of the differences between adjacent pairs in the sorted pairs list from step two (figure 14, below).

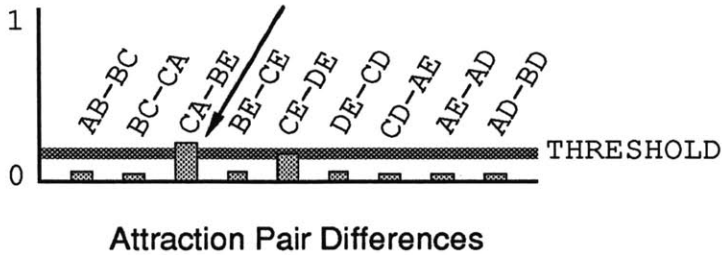


Figure 14.

4) The first difference in the list that exceeds a threshold divides the pairs list into strongly attracting and less strongly attracting pairs (figures 14 and 15). The threshold is determined experimentally. If no difference exceeds the threshold, stop. The group hierarchy is complete.

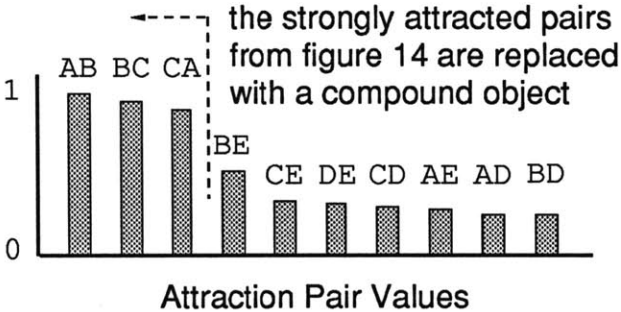


Figure 15.

- 5) Replace each group of mutually attracting objects (corresponding to the strongly attracting pairs from step four) with a compound object (figure 15).
- 6) Repeat steps 1 through 6.

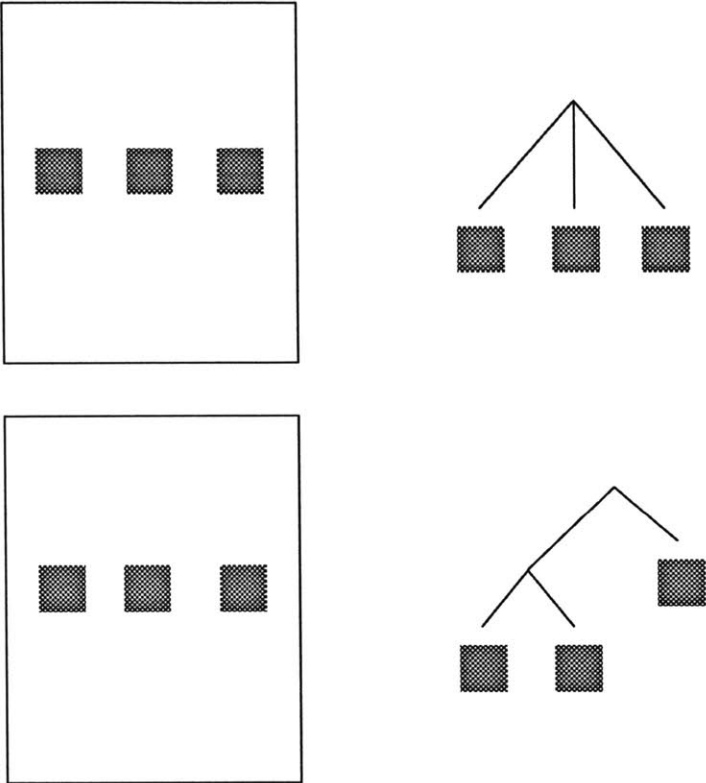


Figure 16. The sensitivity problem. A small change in the middle rectangle position causes an unexpected group to be formed.

A refinement to the grouping algorithm adds a weight constant to each of the attraction pair values found in step 1 of the grouping algorithm. The value of this weight constant is a function of the value of the most attracting pair. The weight constant adjusts the granularity of the attraction function. The purpose of the weight constant is to minimize the algorithm's sensitivity to visually small differences in attractions (not positions) between the most strongly attracting pairs. We do not want visually small changes in the layout to result in big changes in the group hierarchy (figure 16). In the example above, without the weight constant, moving the middle rectangle slightly to

left causes an unexpected regrouping in the hierarchy. We would not expect to see the two rectangles to form a group until they are much closer to each other.

This grouping algorithm is similar to the clustering algorithm described by Pavlidis and Van Wyk [2.2] in their work on automatic beautification of drawings. We extend the grouping idea by the recursive application of the grouping algorithm to build the group hierarchy. The clusters described in [2.2] correspond to the most attracting groups at the lowest level of the group hierarchy. Our grouping algorithm is adaptive. Objects are only grouped in relation to all objects on the page.

Matching Abstraction Hierarchies



Figure 17. Matching nodes must have the same number of children.

In order for a concept to match a layout, the structure of the layout hierarchy must include the whole structure of the concept hierarchy starting from the root. For example, if the root of the concept hierarchy has three children, and the root of the layout hierarchy only has two children, they cannot match (figure 17). The user has specified that the desired layout should

be composed of three main elements, and the layout only has two. The layout hierarchy must be a structural "superset" of the concept hierarchy. A leaf node in the concept hierarchy can match either a compound, or leaf node in the layout hierarchy. This says that a concept element can represent a single layout element, or a group of layout elements.

Comparing Concept and Layout Objects

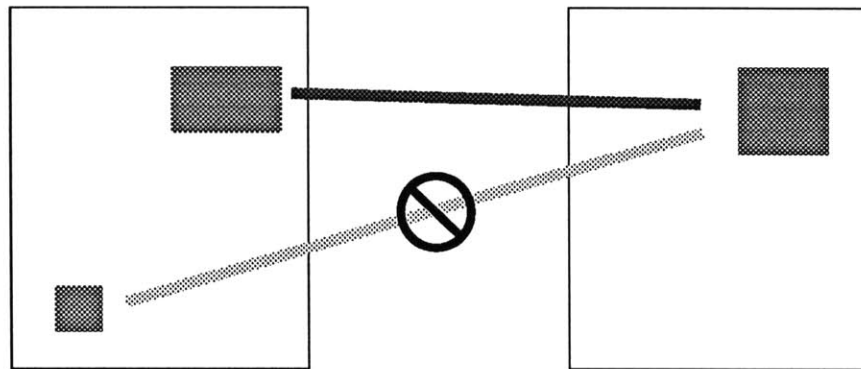


Figure 18. Matching elements must be similar in position and weight.

We know how to match two hierarchies structurally. Now for each node in the concept hierarchy we want to find the node in the layout hierarchy that is its best match. We do not want to match a concept node representing a large object in the upper left corner with a layout node representing a small object in the lower right corner, if there is another layout node representing a large object in the upper left corner (figure 18).

First the system needs to know how to compare primitive objects. In general the objects in our two layouts will not be identical. This requires that

we have a method for computing the similarity between objects according to some visual criteria. At present the system uses the position and the visual weight of the objects as follows:

$s = \text{distance}(\text{obj}_1, \text{obj}_2) + |w_1 - w_2|$, where s is the measure of similarity between obj_1 and obj_2 .

Other visual properties that can be used to evaluate objects are proportion, color, and orientation. It is important to note that this method for computing the similarity score between two objects combines completely different visual properties (apples and oranges). There is no correct way to do this. For example, one user might feel that the position of the objects is the most important factor in determining their similarity, and another user might feel that proportion is more important. While most people would probably agree that position, weight (size and value), and proportion are important properties for determining visual similarity, to some degree the choice of which properties to use and how to weight them is a matter of taste. This means that this similarity function should be accessible to the user, perhaps as part of a system configuration menu.

Matching Compound Objects

Each of the visual properties used to compute the similarity factor for primitive objects (position and weight) can be found for compound objects. The position of a compound object is the center of balance of the objects from which it is composed. The weight of a compound object is the sum of the

weights of the objects from which it is composed. This allows us to compare compound objects in exactly the same way that we compare primitive objects. We can also compare primitive to compound objects, which we need to be able to do in order to match an abstract concept object to a compound layout object.

4. THE LAYOUT GENERATOR

The layout generator uses depth first search and hill climbing to explore the design space. This aggressive search strategy is safe to use because all paths in the search space are of finite length. The depth of the search tree is determined by the number of layout elements that are to be placed. Since a layout is not complete until all elements are positioned, the lengths of all paths to complete layouts is equal to the number of elements. A breadth first search would therefore expand many more nodes than a depth first search before finding layout solutions. For an interactive system such as DAIS, depth first search is preferable. The purpose of the hill climbing heuristics (discussed below) is to increase the likelihood of finding layout solutions early in the search. Local maxima (telephone poles) in the search space are not a fundamental problem since the layout generator will eventually search the entire space defined by the *concept sketch*, finding all acceptable solutions. The trade off we have made by using hill climbing is in favor of a shorter wait for preliminary results, while increasing the chance that a really good solution (in terms of the static evaluation function) might not be found until much later.

The layout generator uses plans, detail generators, and constraints to produce layout proposals.

Plans (Sequencing)

The layout generator has a simple facility for controlling the order in which details are chosen. A very simple plan would be to try all location

alternatives for each object until all possibilities are exhausted. This is a one step plan. A plan is a list of detail generators in the order they are to be applied. Each layout alternative keeps track of which state it is in using a state slot. The generator expands a layout node by applying the detail generator found in the layout's state slot. When a detail generator has produced all alternatives, it puts the next detail generator in the plan, into each alternative layout's state slot. Different search strategies can be easily implemented by creating new plans using different detail generators.

Detail Generators

A small collection of detail generators has been written for the layout generator. A detail generator is a Lisp function that creates new layout alternatives from a partially complete layout. Layout alternatives are created by adding variations on a design detail to the layout. The `set-obj-position` detail generator finds a layout rectangle that has not been placed on the grid, finds all possible positions for that rectangle, and returns new layout alternatives for each possible position. Other detail generators find grid sizes and shapes for a layout rectangle.

Generator Filters

The layout generator has filters that may be applied during the search. A filter is a Lisp procedure that looks at the layout alternatives produced by a detail generator and prunes any that do not satisfy a predicate. For example, a

filter might check layout rectangle positions and prune any alternatives where the largest rectangle is not in the top half of the layout. Filters specialize the behavior of the detail generators by pruning layout alternatives before they can be expanded.

Search Control Strategies

The layout generator uses successive refinement and detailed design knowledge to control the search. Successive refinement is a design approach to complex problems. Rather than attempt to solve a design complex design problem in one step, the idea is to work out an approximate design solution, while ignoring details. Once the approximate solution is known it can be further refined until all design details are specified. This strategy has been used in engineering design expert systems [1.1]. DAIS uses a two step refinement process for generating layouts.

In addition to successive refinement, the layout generator uses detailed design knowledge wherever possible to limit alternatives and to choose the order in which alternatives are expanded. For example, if it is known that an acceptable layout will only have layout rectangles in certain areas of the page, then do not allow rectangles to be placed outside those areas.

Successive Refinement

The task of generating layouts from the concept is divided into two sub-tasks. The first task is to map the concept rectangles onto the layout grid.

Each intermediate layout produced by this mapping process is compared with the concept hierarchy, and any that do not match well are discarded (pruned from the search space). The second task is to place the layout alternatives on the grid using the intermediate layout as a guide. This means that a layout element can only be placed in a position where it is completely overlapped by a rectangle in the intermediate layout. Thus we minimize the possibility that positioning a rectangle will change the top level structure of the layout's grouping hierarchy. This two step refinement strategy has the advantage of pruning most of the overall layout types that have bad groupings at an early stage of the search. Before this strategy was adopted, layouts with incorrect groupings (the majority of potential layout solutions generated) were not discovered until the layout was completed and evaluated by the matcher.

Mapping the *concept sketch* to the grid involves finding grid heights, widths, and positions that are close to those of the concept rectangles. The number of intermediate layouts generated depends on how many grid heights, widths, and position alternatives we allow. The more detail alternatives that we allow, the more likely we are to generate intermediate layouts that do not match the specification.

It is desirable to have specifications that are flexible. This means that we want to be able to find layout solutions using layout elements whose total area is somewhat different from that of the concept rectangles. As we decrease the number of intermediate layout alternatives, we decrease the flexibility of the specification. To see this, let us suppose that we have limited ourselves to just one intermediate layout, and that the total area of the rectangles in this

layout is 10 square grid units. If we are trying to do a layout using layout rectangles whose total area is 12 square grid units, we will not succeed in finding a solution because there is not enough room in our intermediate layout. As we increase the number of intermediate layout alternatives, we increase the flexibility of the specification by generating intermediate layout with varying total areas. The number of grid height, width, and position alternatives generated during the mapping process has been carefully chosen to provide flexibility without undue increase in the complexity of the search space.

Detailed Design Knowledge

The detailed design knowledge used by the layout generator can be formulated as a small set of rules:

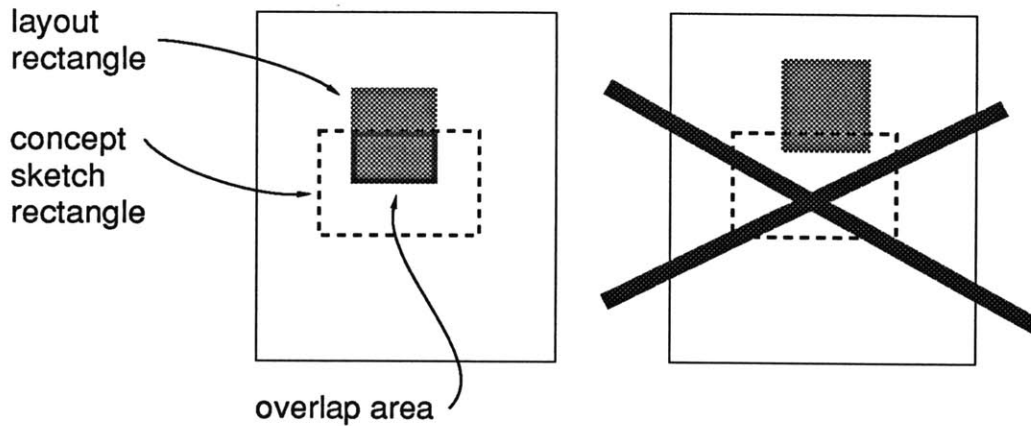


Figure 19. A layout rectangle cannot be positioned so that it overlaps a concept rectangle by less than 50% of the layout rectangle's area.

When positioning a rectangle during the intermediate layout generation phase, only allow positions where the rectangle would be overlapped by more than 50% of its area by a concept rectangle (figure 19). (This rule is embodied as a constraint on the intermediate layout rectangle position generator).

When choosing the partially complete intermediate layout alternative to which the next generator will be applied, choose the one whose most recently placed rectangle has the greatest overlap with a concept rectangle. (This rule changes the depth first search into a hill climbing search. It is built into the intermediate layout rectangle position generator).

If an intermediate layout does not match the concept well, then prune it from the search tree. (This rule is implemented as part of a special detail generator that prepares intermediate layouts to be used as a guide for final layout generation).

If a layout element is positioned where it is not completely overlapped by an intermediate layout rectangle, delete that layout alternative. (This rule is built into the final layout position generator).

5. RELATED WORK

Work related to computer aided page layout is found in several areas, such as diagram understanding, engineering design expert systems, computer typography, and page description languages. Weitzman [1.8] uses *visual expertise* similar to that used by DAIS, in a graphic design system for non-expert users. The system allows the user to design simple graphical displays, while providing design expertise in the form of critiques. It also enforces graphics standards. Unlike DAIS, which produces finished layouts from an abstract *concept sketch*, Weitzman's system critiques an existing design, and offers alternatives only after the user has created the initial specific design. This places most of the design decision making burden on the user.

Mackinlay [1.4] has developed a system for the automatic display of bar charts, scatter plots, and graphs, that treats graphic designs as sentences in a formal graphical language. His system uses a *compositional algebra* to compose different graphic languages, and a generate and test strategy to find a suitable design. The choice of which languages to compose is controlled by *expressiveness criteria*. The selection of a suitable design is governed by *effectiveness criteria*, which are based on perceptual research. This system produces designs that are optimal in terms of the *expressiveness* and *effectiveness criteria*. DAIS makes no assumptions about expressiveness and effectiveness. These are left to the designer.

Plass [5.6] has developed a pagination system that uses a galley of text and a galley of figures and tables. Placement of the figures and tables in the text is controlled by an evaluation function that minimizes the occurrence of

figures on different spreads from their first reference. Plass has shown under what conditions optimal pagination (in terms of the evaluation function) becomes uncomputable. Plass' system is almost entirely content driven, allowing the designer no control over the arrangement of layout elements on the page.

Useful examples of general design theory can be found in the area of engineering design. Brown and Chandrasekaran [1.1], and Mital, Dym, and Morjaria [1.5], have developed expert systems for solving engineering design problems, using design theories that are potentially useful in page layout. Like DAIS both systems use knowledge guided search to explore design spaces. Mital et al use a process of dividing goals into sub-goals to structure the design problem. Brown and Chandrasekaran uses a process of successive refinement much like the one used by DAIS. Like DAIS, both systems use design generators and have some kind of design sequencing. Both systems have more sophisticated backtracking mechanisms than DAIS, which simply backtracks to the previous decision node in the event of a failure. Brown and Chandrasekaran attempt to limit backtracking by handling failures when they occur. Mital et al use dependency directed backtracking.

The DAIS project is part of an ongoing investigation into the representation of graphic design knowledge at the Visible Language Workshop at MIT's Media Laboratory. Previous work at the VLW includes a magazine cover design system [5.2] and a package design system [5.1]. The magazine cover design system uses a grid and adjacency rules to control the placement of images. The package design design system captures a prototype

package design that can be used to generate a family of package designs. Lieberman's spatial allocation ideas outlined in [1.3], inspired the DAIS project.

The *concept sketch* used by DAIS acts like a very flexible template. Beach and Stone [2.1] propose a kind of template called the graphical style sheet which separates the visual style elements from the content of a drawing. The variables in the style sheet are visual properties like line weight, fill color, or font. This approach also separates the visual aspects of a graphic design from the content of the design.

The grouping algorithm used in DAIS is similar in some respects to the clustering algorithm described by Pavlidis [2.2]. The DAIS algorithm creates a hierarchy of objects and compound objects, recursively, while the clustering algorithm only finds clusters at the lowest level of abstraction. Another difference is that the DAIS algorithm is adaptive, ie. it groups objects in relation to all of the objects on the page.

The sketch parsing algorithm used in DAIS is based on ideas proposed by Lakin [4.1]. Lakin points out that there are various sketching, or drawing languages that are well formalized, and hence suitable for computer interpretation. Architectural and engineering drafting, flowcharts, and finite state machine graphs are all examples of formal visual languages. We have created a very straightforward rectangle sketch language for DAIS. There are no syntax errors in this language. Anything that is sketched will be

interpreted as some kind of rectangle. This makes it a very friendly tool for the graphic designer since there is no "wrong" input.

Our ideas about the benefits of separating *visual knowledge* from *content knowledge* were partly inspired by Montalvo [4.2]. Montalvo proposes a general set of visual properties and relations, to be used in a visual knowledge base that can support a wide variety of domain specific applications. These visual properties and relations were derived by studying the response of a group of test subjects to Bongard problems. Bongard problems are a kind of visual puzzle that expose a single visual property or relation. The properties and relations that were consistently found in the Bongard problems by most of the test subjects are used in the final set. Those that were difficult to see are eliminated from the final set. Montalvo's work is important because it proposes a technique for acquiring visual knowledge.

Some of the complex issues involved in typesetting tables are highlighted by Beach [5.3], while Rubenstein [5.9] outlines some of the computational problems encountered in book design. Chamberlin [5.4] proposes classifying page layout systems according to three mutually independent criteria. They are batch versus interactive, text only versus text and images, and procedural versus declarative. DAIS is declarative and interactive. Kernighan [5.5], and Reid [5.7] both describe procedural languages.

6. CONCLUSION

In this thesis we investigate *visual knowledge* and its application to page layout problems. We argue that separating *visual knowledge* from *content knowledge* makes graphically adept systems easier to understand, build and maintain. To test this idea we have identified content free visual properties and relations, and modelled them in the Do As I Sketch page layout system. This visually knowledgeable page layout system can be expanded into a comprehensive page layout system by adding a module containing *content knowledge*. The content module would affect the layout process by communicating goals and constraints to the visual module, which directly controls the layout.

We have shown that content free *visual knowledge* can be represented and effectively utilized in an experimental page layout system. To show that *content knowledge* is separable from *visual knowledge* will require that DAIS be extended by adding a *content knowledge* module. This is an area for future investigation.

A primary goal of this research has been to use *visual knowledge* representations that will support human-computer communication of flexible design concepts for page layouts. DAIS *concept sketches* succeed in meeting this goal. They are independent of any particular grid, and the sizes, and shapes of any particular set of layout elements. *Concept sketches* are also independent of the number of layout elements provided there are at least as many layout elements as there are groupings in the *concept sketch*. Given a

concept sketch, a grid, and some layout elements, DAIS successfully produces layouts with groupings that match those of the *concept sketch*.

Future Work

The set of visual properties and relations DAIS uses can be extended to include color, size, and shape. This would allow the system to create groupings based on these new properties. How these additional groupings would be used by the system is an open question. One possibility would be to find the group hierarchies of the layout elements by color, size, and shape before starting the layout generator. Comparison of these hierarchies to the concept hierarchy might suggest possible sets of layout elements to be grouped by the layout generator.

DAIS assumes that the layout elements have been previously sized to fit the grid. Recursive application of the layout generator could produce individual layout elements in the same way that it lays out pages.

Extending DAIS to do multiple page layouts is another area for investigation. We showed earlier that the size of the design space increases exponentially based on the number of allowed positions and the number of layout elements. New means will have to be found for controlling the possible positions of layout elements. The addition of *content knowledge* to the system can be helpful. Layout elements can be ordered according to where they are referenced in the text. This ordering can limit the possible positions for each element to a small section of the total article. Michael Plass has done

work on optimal pagination of documents with text and figures that should be considered [5.6].

DAIS cannot be used for layouts where elements are tightly packed on the page, as they are in catalogs or newspapers. The grouping by distance function does not find useful groups in a packed layout. An investigation into visual relations that support grouping in packed layouts would address this problem. One possibility is to group by edge alignments.

The most interesting area for further research is to add *content knowledge* modules to the system. Content modules should find groupings for the layout elements based on their semantic relationships. The *visual knowledge* already in place could then arrange the layout elements so that their visual groupings reflect their semantic groupings.

BIBLIOGRAPHY

Artificial Intelligence

- 1.1 Brown, David C. and Chandrasekaran, B. "Knowledge and Control for a Mechanical Design Expert System" *Computer*, pp 92-100, July 1986.
- 1.2 Gardner "Search" in The Handbook of Artificial Intelligence, Barr, Avron & Feigenbaum (eds.) HeurisTech Press, Stanford California, 1981.
- 1.3 Lieberman, Henry "Getting Things to Fit" unpublished paper, 1987
- 1.4 Mackinlay, Jock "Automating the Design of Graphical Presentations of Relational Information" *ACM Transactions on Graphics*, Volume 5, Number 2, April 1986.
- 1.5 Mital, Dym, Morjaria "PRIDE: An Expert System for the Design of Paper Handling Systems" *Computer*, pp 102-114, July 1986.
- 1.6 Newell, Shaw, and Simon "Report on a General Problem Solving Program" *Proceedings of the International Conference on Information Processing*, pp 256 - 264, UNESCO, Paris, 1959.
- 1.7 Sacerdoti, Earl D. "Planning in a Hierarchy of Abstraction Spaces" *Artificial Intelligence* 5, 1971, 115 - 135.
- 1.8 Weitzman, Louis "Designer: A Knowledge-Based Graphic Design Assistant" *MCC Technical Report Number ACA-HI-017-88*, January 1988.

Computer Graphics

- 2.1 Beach, Stone "Graphical Style: Towards High Quality Illustrations" *Computer Graphics*, pp 127-135, Volume 17, Number 3, July 1983.
- 2.2 Pavlidis, Theo and Van Wyk, Christopher J. "An Automatic Beautifier for Drawings and Illustrations" *Computer Graphics* pp 225-234, Volume 19, Number 3, July 1985.

Design Theory

- 3.1 Albers, Joseph "The Interaction of Color", Yale University Press, New Haven, 1963.
- 3.2 Arnheim, Rudolph "The Power of the Center: A Study of Composition in the Visual Arts", University of California Press, Berkely, California, 1982.
- 3.3 Dondis, Donis A. "A Primer of Visual Literacy", MIT Press, Cambridge, Ma., 1973.
- 3.4 Gerstner "Designing Programs" Arthur Niggli, Ltd. Jeuten, Switzerland, 1968.
- 3.5 von Goethe, Johan Wolfgang "Theory of Colors", MIT Press, Cambridge, Ma., 1970.
- 3.6 Itten, Johannes "Design and Form; The Basic Course at the Bauhaus", Reinhold Publishing Corporation, New York, 1964.
- 3.7 Ocvirk, Bone, Stinson and Wigg "Art Fundamentals" Wm. C. Brown Co. 1975.
- 3.8 Simon, Herbert A. "Style in Design" in Spatial Synthesis in Building Design, Charles M. Eastman (ed) Wiley and Sons, 1975.
- 3.9 Wingler, Hans M. "The Bauhaus" MIT Press, Cambridge, Ma., 1969.

Diagram Understanding

- 4.1 Lakin, Fred "Spatial Parsing for Visual Languages" in *Visual Languages*, Shi-Kuo Chang (ed.) Plenum Press, 1987.
- 4.2 Montalvo, Fanya S. "Diagram Understanding: The Intersection of Computer Vision and Graphics" *AI Memo No. 873*, MIT November, 1985.

Page Layout

- 5.1 Amari, Thomas R. "Automating the Design of Packaging Families Using *PackIT*, The Packager's Inferencing Tool" Masters Thesis, MIT, 1987.
- 5.2 Badshah, Alka G. "GRID - Graphic Intelligence in Design: An Expert Layout System" Masters Thesis, MIT, 1987.
- 5.3 Beach, Richard J. "Tabular Typography" in Text Processing and Document Manipulation, van Vliet, J. C. (ed), Proceedings of the International Conference University of Nottingham, 14-16 April, 1986. Cambridge University Press, New York, 1986.
- 5.4 Chamberlin, Donald D. et al. "JANUS: An Interactive System for Document Composition" *SIGPLAN Notices*, Volume 16, Number 6, June, pp. 82-91.
- 5.5 Kernighan, Brian W. "PIC - A Language for Typesetting Graphics" *SIGPLAN Notices*, Volume 16, Number 6, June, pp. 92-98.
- 5.6 Plass, Michael F. "Optimal Pagination Techniques for Automatic Typesetting Systems" PhD dissertation, Stanford University Department of Computer Science, Report Number STAN-CS-81-870, 1981.
- 5.7 Reid, Brian K. "Procedural Page Description Languages" in Text Processing and Document Manipulation, van Vliet, J. C. (ed), Proceedings of the International Conference University of Nottingham, 14-16 April, 1986. Cambridge University Press, New York, 1986.
- 5.8 Reid, Brian K. "Scribe: A Document Specification Language and Its Compiler" Ph.D. dissertation, Carnegie-Mellon University, Oct. 1980.
- 5.9 Rubinstein, Richard Digital Typography - An Introduction to Type and Composition for Computer System Design Addison-Wesley Publishing Co., Reading Massachusetts, 1988.