Effective Visualization of Large Multidimensional Datasets

by

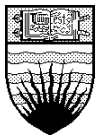
Christopher G. Healey
B.Math, The University of Waterloo, 1990
M.Sc., The University of British Columbia, 1992

A Thesis Submitted in Partial Fulfilment of
the Requirements for the Degree of

Doctor of Philosophy

in the Faculty of Graduate Studies
Department of Computer Science

We accept this thesis as conforming to the required standard



THE UNIVERSITY OF BRITISH COLUMBIA

Department of Computer Science
The University of British Columbia
2366 Main Mall
Vancouver, Canada
V6T 1Z4

Date: _____

Abstract

A new method for assisting with the visualization of large multidimensional datasets is proposed. We classify datasets with more than one million elements as large. Multidimensional data elements are elements with two or more dimensions, each of which is at least binary. Multidimensional data visualization involves representation of multidimensional data elements in a low dimensional environment, such as a computer screen or printed media. Traditional visualization techniques are not well suited to solving this problem.

Our data visualization techniques are based in large part on a field of cognitive psychology called preattentive processing. Preattentive processing is the study of visual features that are detected rapidly and with little effort by the human visual system. Examples include hue, orientation, form, intensity, and motion. We studied ways of extending and applying research results from preattentive processing to address our visualization requirements. We used our investigations to build visualization tools that allow a user to very rapidly and accurately perform exploratory analysis tasks. These tasks include searching for target elements, identifying boundaries between groups of common elements, and estimating the number of elements that have a specific visual feature. Our experimental results were positive, suggesting that dynamic sequences of frames can be used to explore large amounts of data in a relatively short period of time.

Recent work in both scientific visualization and database systems has started to address the problems inherent in managing large scientific datasets. One promising technique is knowledge discovery, "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data". We hypothesise that knowledge discovery can be used as a filter to reduce the amount of data sent to the visualization tool. Data elements that do not belong to a user-chosen group of interest can be discarded, the dimensionality of individual data elements can be compressed, and previously unknown trends and relationships

can be discovered and explored.

We illustrate how our techniques can be used by applying them to real-world data and tasks. This includes the visualization of simulated salmon migration results, computerized tomography medical slices, and environmental datasets that track ocean and atmospheric conditions.

Contents

List of Tables

List of Figures

Acknowledgements

A lot of different people helped me with various parts of this thesis. Technical support, moral support, mental support, you name it.

I want to thank both my supervisors, Kelly Booth and Jim Enns, for their advice, support, and guidance. I would also like to thank my committee members, Dr. Alain Fournier, Dr. Raymond Ng, Dr. Uri Ascher, Dr. Peter Graf, and Dr. Brian Funt, for the time and effort they spent to read and comment on my work.

Ron Rensink wrote the software which was used to run a number of our experiments. He also provided many helpful references in the area of preattentive processing. Ron was the one who introduced me to both preattentive processing and Jim Enns, so if I ever become famous, he can tell everyone "I'm the one who started his career".

I have friends here, but some have been with me (on and off) longer than others. In no particular order: Bill Gates, Gwen Litchfield, Raza Khan, and Vishwa Ranjan. There are also some people who left early, but with whom I still keep in touch: Chris Romanzin and Pierre Poulin.

The most important "acknowledgement" goes to my parents and my girlfriend. My parents put up with me for the first twenty-four years. Hiroko put up with me for the last five. I'm not sure who did more work. Either way, they both loved me, supported me, and pushed me when I needed motivation. What more could you ask for?

Chapter 1

Introduction

Scientific visualization in computer graphics is a relatively new field of research. The term "visualization" was used during a 1987 National Science Foundation (NSF) panel report on how to apply computer science techniques to data analysis problems [McCormick et al., 1987]. The panel defined the domain of visualization to be the development of general purpose tools and the study of research problems that arise in the process. Participants on the panel emphasised a number of research goals, specifically:

- visualization should combine research results from different disciplines (*e.g.,* computer science, computer graphics, psychology, and visual arts)
- visualization should address both the intelligent display of data and the intelligent management of the underlying dataset
- visualization does not have to be complicated in order to be useful
- visualization should be performed interactively while the data is being generated

Panel members at a similar visualization roundtable noted that the desire for computer-based data visualization arose from the need to analyse larger and more complex datasets [Wolfe and Franzel, 1988]. Scientific visualization has grown rapidly in recent years as a direct

result of the overwhelming amount of data being generated. New visualization techniques need to be developed that address this "firehose of information" if users hope to analyse even a small portion of their data repositories.

Many traditional computer software tools are now being extended to provide user interaction and real-time visualization of results. For example, visual interactive simulation studies ways of adding useful visualization and user interaction components to simulation programs [Hurrion, 1980; Bell and O'Keefe, 1987; Bell and O'Keefe, 1994]. Other types of applications also need to display data in real-time. In air traffic control screens are often shared by different operators who acquire visual data from different parts of the display at the same time. Visualization techniques for this environment must allow a variety of tasks to be performed rapidly and accurately on dynamically changing subsets of the overall display. Medical imaging systems such as CT, MRI, PET, and ultrasound are another type of application that could benefit from real-time visualization. Techniques that allowed rapid and accurate visual analysis of more than one aspect of the data might decrease the amount of time needed to complete the diagnostic task. This is important, since these types of systems often cannot be time-shared by multiple users. Any improvement in visualization would increase total throughput for the system. Moreover, better displays might reduce errors made during analysis. Even a small increase in accuracy is important in this type of environment.

A variety of methods have been used to convert raw data into a more usable visual format. Both Tufte [1983, 1990] and Collins [1993] give an interesting review of pre-computer visualization techniques. Two of the best known examples are maps and the conversion of numeric data into different types of graphs. Diverse solutions for displaying high-dimensional datasets in a low-dimensional environment such as the computer screen have been proposed [Pickett and Grinstein, 1988; Ware and Beatty, 1988; Grinstein et al., 1989; Enns, 1990a; Enns, 1990b]. Specialized software tools such as the Application Visualization System (AVS), apE, VIS-5D, and the Wavefront Data Visualizer [Upson, 1989; Hibbard and

Santek, 1990; Vande Wettering, 1990] have been developed for performing visualization on computer graphics workstations.

A recent update on the NSF visualization report described research being performed at a number of academic institutions [Rosenblum, 1994]. Although many visual presentation techniques have been studied (*e.g.,* volume visualization, fluid flow, and perceptual visualization), much less work has focused on formulating guidelines for their design. Results in this thesis are meant to address this more general issue.

1.1 Research Goals

Our goal is the investigation and development of techniques for visualizing rapidly and accurately large multidimensional datasets. We formally define "large" and "multidimensional" as follows:

- *large:* the size of a dataset is the combination of two separate characteristics: the absolute number of data elements within a single data frame, and the number of data frames that make up the dataset
- *multidimensional:* the dimensionality of a data element also depends on two separate characteristics: the number of different attributes or dimensions embedded in the element, and the number of unique values each attribute can represent (*e.g.,* a binary, multivalued, or continuous attribute)

A typical workstation monitor has a resolution of approximately one million pixels. This represents a limit on the number of data elements (one per pixel) that can be displayed on a single screen. We define large to be a dataset with more than one million elements (*i.e.,*

more than can be displayed on a single screen). We define multidimensional to be a data element with two or more dimensions, where each dimension is at least binary.

It is important to understand how existing visualization systems and techniques address our problem environment. We describe three types of visualization environments: standard visualization systems, multidimensional visualization techniques, and hybrid visualization packages that have access to an underlying database management system.

Standard Visualization Systems

A number of specialized software tools (*e.g.,* AVS, apE, Wavefront Data Visualizer, and Iris Explorer) have been developed to perform scientific visualization on computer graphics workstations. These systems have the potential to visualize large multidimensional datasets, in particular because they are extensible and because they support a wide range of simple visual presentation methods. In spite of this, we feel these systems are missing a number of key components that are necessary for solving the multidimensional visualization problem. First, the built-in data management facilities are usually limited to reading and writing files, plus simple filtering using comparison operators. Attempts to process large datasets often result in the visualization tool being overwhelmed by the amount of data that needs to be managed. Beyond that, no rules or guidelines are provided to deal specifically with displaying either large or multidimensional datasets. Users are left to answer key questions on their own, such as: How can I display a dataset that does not "fit" in one screen? How can I effectively display multidimensional data elements? How can I show structure in the data (*e.g.,* coherent regions, related elements, elements with unique attributes)? These packages offer a flexible foundation for building tools that deal with visualization of large multidimensional datasets, but proper data management and visualization techniques must first be identified and made available to the user. This thesis investigates exactly these kinds

of data management and visualization guidelines.

Multidimensional Visualization Techniques

Our research focuses on multidimensional data visualization. We are trying to address the question: How can I display multidimensional data in a spatially low-dimensional environment, such as a computer screen or printed media? Researchers have approached this problem in different ways. Enns and Rensink [1990a, 1990b] discuss using the human visual system to efficiently process multidimensional datasets; they describe geometric icons that combine the power of the computer and the human visual system. Ware and Beatty [1988] have designed a method that uses colour and spatial location to represent multidimensional data elements; subsets of the data with similar values appear as a spatial "cloud" of similarly coloured squares. Pickett and Grinstein [1988, 1989] use results from cognitive psychology as a basis for the design of their visualization tools; they display structure in the data as a set of textures and boundaries, so that groups of data elements with similar values appear as a spatial group with a unique texture in the display. None of these techniques explicitly address the problem of large datasets since they are restricted to, at best, the number of pixels that can be displayed on a single screen. Some of the techniques are limited to a specific task, for example, Ware and Beatty's tool helps a user perform coherency testing, while Pickett and Grinstein's displays regions and boundaries in a dataset. An important question to ask is whether additional visual features can be integrated into any of these techniques. For example, could Pickett and Grinstein use colour to encode additional information in their displays? Variations in colour might mask texture boundaries during visualization, or vise-versa. Finally, it is difficult to see how either Ware and Beatty or Enns and Rensink could easily extend their techniques to handle higher dimensional data elements.

Hybrid Visualization Packages

In order to address management of large datasets, researchers are now studying the usefulness of visualization tools that are a combination of a commercial database package (DBMS) and a visualization system like those described above [Kochevar et al., 1993; Stonebraker et al., 1993]. Because both the visualization system and the DBMS are extensible, it is relatively easy to provide a layer of software to "glue" the two systems together. This makes the functionality of the underlying database available from within the visualization system, dramatically enhancing both the scope and efficiency of data management operations. Unfortunately, this does not address all of the problems related to the display of large or multidimensional datasets. Most of the currently available DBMSs are relational. Many scientific datasets contain errors, missing values, or noise, all of which are difficult to represent in a relational database. Relational DBMSs have difficulty supporting datasets that do not map conceptually to a relational model (*e.g.,* spatial or temporal datasets). Finally, some of the most promising new data management techniques such as statistical databases, classification, temporal databases, spatial data handling, and knowledge discovery are not immediately available and are difficult to provide, even given the extensibility of the current DBMSs.

1.2 Research Overview

Our investigation of the problem of visualizing large multidimensional datasets is made up of two parts. First, we studied new methods for visualizing multidimensional data. Our techniques address the problems of dataset size and data element dimensionality by exploiting the built-in processing of the human visual system. Second, we studied the effectiveness of a new database technique, knowledge discovery, for compressing and summarizing the

important details buried in large datasets. We used our results to design visualization tools that allow users to perform rapid and accurate exploratory analysis tasks such as target detection, boundary detection, region tracking, and estimation. Users can filter their dataset using different knowledge discovery algorithms to reduce both its size and dimensionality. The resulting data is displayed in a 2D spatial environment (the computer screen) using visual features such as hue and orientation. Large datasets are divided into sequences of data frames; the frames are displayed rapidly (usually with an exposure duration of 250 msec or less per frame) one after another in an animated, movie-like fashion. This allows a user to explore large amounts of data in a relatively short period of time.

The next two sections provide a brief overview of our research and the results we obtained. Each section is divided by chapter titles. More detailed descriptions of our work can be found in the corresponding chapters.

1.3 Multidimensional Visualization

Within the multidimensional visualization context, we focused on techniques for performing rapid and accurate exploratory data analysis. This type of analysis is used to inspect a dataset and quickly identify areas that might benefit from further, more detailed investigation. The kinds of tasks performed during this stage of analysis include:

- *target detection*, where users attempt to rapidly and accurately detect the presence or absence of a "target" element in a display
- *boundary detection*, where users attempt to rapidly and accurately identify boundaries between groups of elements, where all the elements in each group have some property in common

- *region tracking*, where users attempt to track groups of elements with a common property as they move through time and space
- *counting and estimation*, where users attempt to count or estimate the number or percentage of elements in a display that have a specific attribute

We believe results from research in preattentive processing can be used to assist with the design of visualization tools that perform these kinds of tasks.

Chapter 2: Preattentive Processing

Researchers in psychology and vision have been working to explain how the human visual system analyses images. One interesting result has been the discovery of visual properties that are "preattentively" processed. These properties are detected immediately by the visual system. Viewers do not have to focus their attention on an image to determine whether elements with a given property are present or absent. Examples of common preattentive features include hue, intensity, orientation, length, and motion. Unfortunately, choosing visual features in an ad hoc manner and matching them to data attributes will not necessarily result in preattentive displays. Indeed, too often the tool itself inhibits the user's ability to extract the desired information. Results from research in preattentive processing can be used to identify and avoid this kind of visual interference.

Properties that are processed preattentively can be used to highlight important image characteristics. Experiments in psychology by Triesman, Julész, Quinlan, and others have used preattentive features to assist in performing exactly the visual tasks listed above. Research in visualization has shown that preattentive features allow users to better perform visual tasks such as grouping of similar data elements, detection of elements with a unique characteristic, and estimation of the number of elements with a given value or range of values [Pickett and Grinstein, 1988; Grinstein et al., 1989; Healey et al., 1993; Healey et al., 1996].

The key advantage of preattentive visualization techniques is that they are rapid (a preattentive task can usually be completed in less than 250 msec, moreover, the time required to complete the task is independent of the number of data elements being displayed) and accurate.

Chapter 3: Preattentive Estimation

Target detection, boundary detection, and grouping (for a single data frame) have been studied in depth in the preattentive processing literature [Julész, 1981; Julész and Bergen, 1983; Triesman, 1985; Triesman and Gormican, 1988; Duncan and Humphreys, 1989; Müller et al., 1990]. These results can be extended directly to scientific visualization. Researchers have also studied counting and enumeration in the low-level visual system [Varey et al., 1990; Trick and Pylyshyn, 1994]. Our work investigated another common analysis task, relative estimation, that had not been studied by the preattentive processing community. Our initial results showed that estimation using either hue or orientation was, in fact, preattentive [Healey et al., 1993]. We extended these results by answering three additional questions related to the use of hue and orientation during preattentive estimation [Healey et al., 1996]:

- How are hue and orientation estimation affected by varying display duration?
- How are hue and orientation estimation affected by varying feature difference?
- What is the tradeoff between display duration and feature difference during estimation?

We chose hue and orientation because they are two features that are commonly used in existing visualization software. Both hue and orientation have been shown to be preattentive by Triesman, Julész, and others [Julész and Bergen, 1983; Triesman, 1985; Quinlan and Humphreys, 1987]. Moreover, research has shown that hue exhibits a strong interference effect over form (or orientation) during certain preattentive tasks [Callaghan, 1984; Callaghan,

1989; Healey et al., 1996]. Understanding how hue and orientation interact in a preattentive visualization environment is important. If a visualization tool is being used to display multiple independent data values, interference among features must be avoided. If a visualization tool is being used to investigate a specific relationship, the "strongest" feature should be used to encode that relationship. Secondary features used to encode additional data values should not interfere with the primary feature and task.

Chapter 4: Colour

Colour is a common and often-used visual feature. Unfortunately, choosing colours for data visualization is complicated, since a number of different factors can affect the interactions that occur between colours. We completed a thorough review of colour in preparation for measuring and controlling these effects. Colour is studied from three different perspectives. First, we describe various three-dimensional models used in computer graphics to accurately represent colour. Next, we examine research on the use of colour in scientific visualization. Finally, we investigate how colour has been studied in the context of preattentive processing. This background information identifies three separate factors that can be used to measure the effectiveness of a set of colours for representing data values during scientific visualization: colour distance, linear separation, and colour category.

Chapter 5: Effective Hue Selection

Multidimensional visualization techniques must be able to encode data dimensions with more than two individual values. Our preattentive estimation experiments were restricted to data dimensions that were binary, since we wanted to pick hues (and orientations) that were easy to distinguish from one another. This is simple to do when only two different values are required (*e.g.,* during estimation two distinct hues, blue and red, were chosen). Continuous data attributes used during our experiments had to be split into two discrete

ranges, to "fit" our visualization design. In order to move beyond this restriction, we wanted to identify methods for mapping multivalued data attributes onto visual features such as hue and orientation, while still allowing rapid and accurate analysis on the resulting displays. The selection of multiple orientations for use during preattentive tasks has been studied in the preattentive processing literature [Nothdurft, 1985a; Nothdurft, 1991; Wolfe et al., 1992]. We investigated methods for selecting multiple hues for use during visualization by answering the following three questions:

- How can we support rapid and accurate identification of individual data elements through the use of colour?
- What factors determine whether a "target" element's colour will make it easy to find, relative to differently coloured "non-target" elements?
- How many colours can we display at once, while still allowing for rapid and accurate target identification?

Results from our experiments showed that we need to consider three separate factors when selecting multiple hues: colour distance, linear separation, and colour category. We developed simple methods for measuring and controlling each of these effects. This allowed us to chose up to seven isoluminant hues for use during visualization. Each hue was equally distinguishable, and each could be identified preattentively in our data displays.

Chapter 6: Real-Time Visualization

A key consideration for visualizing large datasets involves the difference between static and dynamic data frames. A static frame is shown to a user in isolation. The user then decides how to proceed based on information in the frame. Research to date in preattentive visualization has been restricted to static frames. Unfortunately, this limits the display

technique to the resolution of the screen. An obvious question to ask is: If I can perform tasks in 200 msec on a static frame, can I perform the same tasks on an sequence of frames displayed at five frames per second? A dynamic environment displays a sequence of frames to the user one after another in a movie-like fashion. Each frame is shown for a fixed period of time, after which it is replaced by the next frame in the sequence. The advantage of such a technique is the ability to scan through large amounts of data in a relatively short period of time. Suppose a data frame displays 400 elements. Since each element in the frame has an available screen region of approximately 50×50 pixels, it can use visual features an individual pixel cannot (*e.g.,* shape, size, orientation, length). A dynamic sequence of five frames per second could display one million elements in about nine minutes. Even if a user spent 20% of their time browsing the dataset and 80% performing more detailed analysis on individual frames, we can still exceed the display bounds of a single workstation screen in less than an hour. The ability to perform exploratory analysis in real-time allows us to access a large dataset in its entirety.

We show through a set of experiments that important aspects of preattentive processing extend to a real-time environment. A visualization tool that uses preattentive features allows users to perform rapid and accurate target and boundary detection, all in real-time on temporally animated data frames. Moreover, interference properties previously reported for static preattentive visualization were found to exist (with similar consequences) in the dynamic environment.

1.4 Knowledge Discovery

Current database research is extending the original database model in a number of novel and interesting ways. An NSF panel on advanced database systems described the focus

of past research, and presented a strong argument for continued work [Silbershatz et al., 1990]. Panel members felt rapid advances in a number of areas that use databases are overwhelming currently available data management techniques. Scientific visualization was specifically cited as an area that is moving beyond the boundaries of traditional database systems. The panel identified the following problems as essential to future database research:

- new data models that deal with complex objects such as spatial data, time, and uncertainty
- query and access methods to manage very large databases (*i.e.,* over one terabyte in size); this involves the scaling of current algorithms, the development of new storage and access techniques, and support for heterogeneous, distributed databases
- the ability to "mine" implicit patterns, trends, or relationships from very large scientific, biomedical, or business databases
- the ability to embed and process efficiently declarative and imperative rules within a database

The NSF panel discussed at some length scientific visualization and the problems inherent in managing scientific datasets. An area of research well suited to address this problem is knowledge discovery, "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" [Frawley et al., 1991]. Statistical, database, and machine learning algorithms can be combined to uncover trends, dependencies, and relationships implicit in large multidimensional datasets.

Chapter 7: Dataset Management

We believe that knowledge discovery techniques can be used to improve multidimensional data visualization. These techniques would lie between the visualization tool and the under-

lying dataset, acting as a filter to mark or compress regions of interest in the data. Although there is always a measure of uncertainty in the results returned by knowledge discovery algorithms, they seem well suited to an exploratory data analysis environment. A user could control, combine, and most importantly analyse the results of knowledge discovery against what is known about the original dataset. Users could interactively choose to pursue or ignore trends, dependencies, or groupings the algorithms suggest. Different techniques could be combined in various ways to improve results or increase confidence in the information being provided. Knowledge discovery could be used to advance our goal of visualizing large multidimensional datasets in the following ways:

- reducing the amount of data to be visualized, by classifying data elements into groups and ignoring groups that do not contribute to the relationship being investigated
- reducing the number of attributes associated with each data element, by using a discovered classification to replace multiple data dimensions
- reducing the number of attributes associated with each data element, by ignoring dimensions that are independent of the relationship being investigated
- showing structure in the dataset; this goes beyond simply showing whether data elements are coherent or not
- compressing time-varying data along the time axis, to reduce the amount of data to visualize, and the length of any corresponding "animation" of the data frames
- finding and visualizing previously unknown trends or relationships within or among data elements in the dataset

Chapter 8: Knowledge Discovery in Visualization

Four different knowledge discovery algorithms were implemented and integrated into our visualization environment. Each algorithm was modified to identify attribute dependencies that were found when classification rules were built. The algorithms were also extended to provide confidence weights for each classification they performed. This allows a user to assess the confidence an algorithm attaches to a particular result. The effectiveness of knowledge discovery in scientific visualization was measured by examining the increased accuracy and size reductions obtained when we visualized NASA's Comprehensive Ocean-Atmospheric Data Set.

1.5 Contributions

The research reported in this thesis directly addresses the requirements put forward by the NSF and other visualization panels. Figure 1.1 provides an overview of how our different experiments fit within the overall goal of visualizing large, multidimensional datasets. Our techniques build on earlier research in computer graphics, databases, and cognitive psychology. Preattentive visualization is computationally simple, which makes it applicable to interactive environments such as real-time displays and visual interactive simulation. Our research investigates both the display of data elements and management of the underlying dataset. We believe that our work contributes the following advances to current research in visualization, preattentive processing, and database systems.

1. Hue and orientation feature spaces have been investigated more fully. We determined through experimentation both the perceived feature difference and the display duration needed to perform an estimation task using either hue or orientation. We also measured the tradeoff between these two properties.

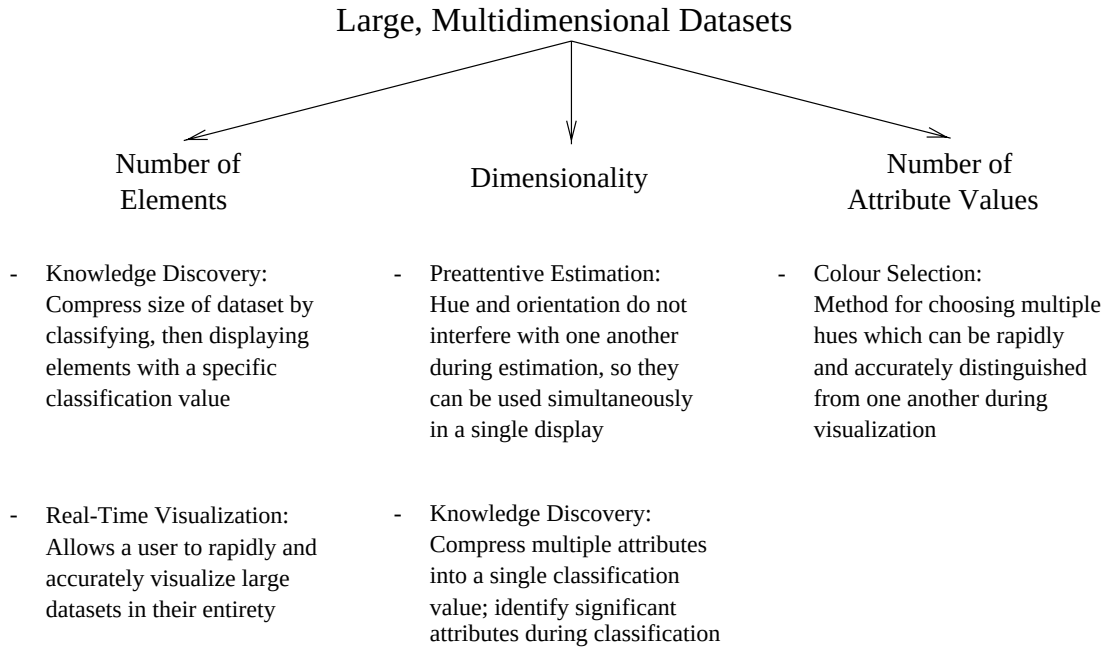


Figure 1.1: An overview of how research reported in this thesis addresses the three types of large (total number of elements, dimensionality of each element, and the number of unique values for each attribute) inherent in large, multidimensional datasets

2. Methods for mapping hue and orientation onto multivalued data attributes were investigated. Our techniques describe how to choose multiple hues and orientations that allow rapid and accurate analysis on the resulting displays.

3. Experiments were conducted to show that traditional preattentive tasks such as target detection and boundary detection (and associated interference effects) extend to an environment where dynamic sequences of frames are displayed rapidly one frame after another.

4. Various knowledge discovery algorithms were investigated in the context of exploratory data analysis. Normalized confidence weights were provided for each task the algorithms performed so that users could assess the confidence an algorithm assigned to the results it returned.

The results we provide in each of these areas should help to improve understanding of scientific visualization in general, and visualization of large multidimensional datasets in particular. Our results can be interpreted as a set of general guidelines for the use of common visual features such as hue, form, and orientation. The results also help to describe the effects of visual phenomena such as feature interference and emergent features in scientific visualization. Finally, the integration of knowledge discovery into a visualization environment demonstrates that intelligent management of the underlying dataset can reduce both the amount and the dimensionality of the data that is displayed to the user.

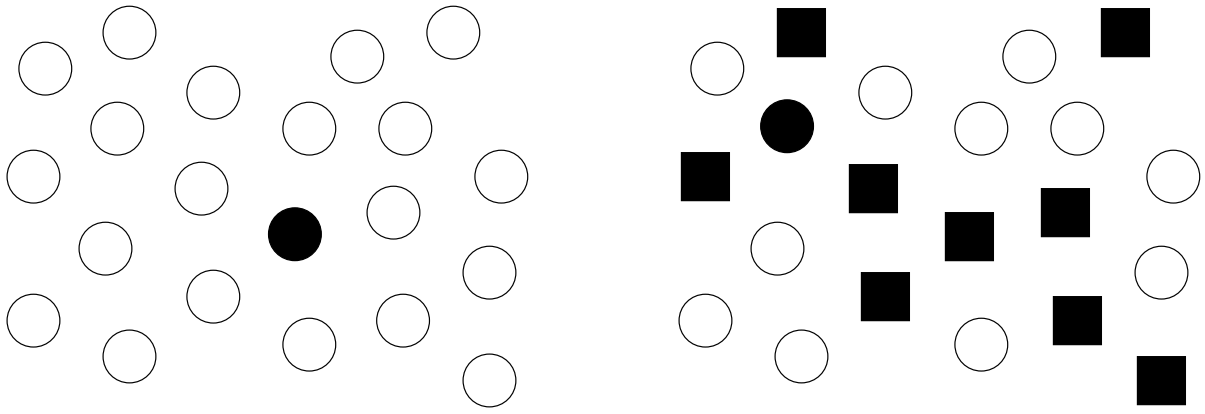
Chapter 2

Preattentive Processing

For many years vision researchers have been working to explain how the human visual system analyses images. One very interesting result has been the discovery of a limited set of visual properties that are processed preattentively, without the need for focused attention. Typically, tasks that can be performed on large multi-element displays in less than 200 to 250 msec are considered preattentive. Eye movements take at least 200 msec to initiate, and random locations of the elements in the display ensure that attention cannot be prefocused on any particular location, yet subjects report that these tasks can be completed with very little effort. This suggests that certain information in the display is processed in parallel by the low-level visual system.

A simple example of preattentive processing is the detection of a filled circle in a group of empty circles (Figure 2.1a). The target object has the visual feature "filled" but the empty distractor objects do not (all nontarget objects are considered distractors). A viewer can tell at a glance whether the target is present or absent.

Objects that are made up of a conjunction of unique features cannot be detected preattentively. A conjunction target item is one that is made up of two or more features, only one of which is contained in each of the distractors. Figure 2.1b shows an example of conjunction search. The target (again a filled circle) is made up of two features, filled and circular. One



(a) (b)

Figure 2.1: Examples of target detection: (a) filled circle target can be preattentively detected because it contains the unique feature "filled"; (b) filled circle target cannot be preattentively detected because it contains no preattentive feature unique from its distractors

of these features is present in each of the distractor objects (filled squares and empty circles). Numerous studies show that the target cannot be preattentively detected, forcing subjects to search serially through the display to find it.

Visual properties that are processed preattentively can be used to highlight important image characteristics. Table 2.1 lists some of the visual features that have been identified as preattentive. Experiments in psychology have used these features to perform the following preattentive visual tasks:

- *target detection*, where users attempt to rapidly and accurately detect the presence or absence of a "target" element with a unique visual feature within a field of distractor elements (Figure 2.1)
- *boundary detection*, where users attempt to rapidly and accurately detect a texture boundary between two groups of elements, where all of the elements in each group have a common visual property (Figure 2.2)

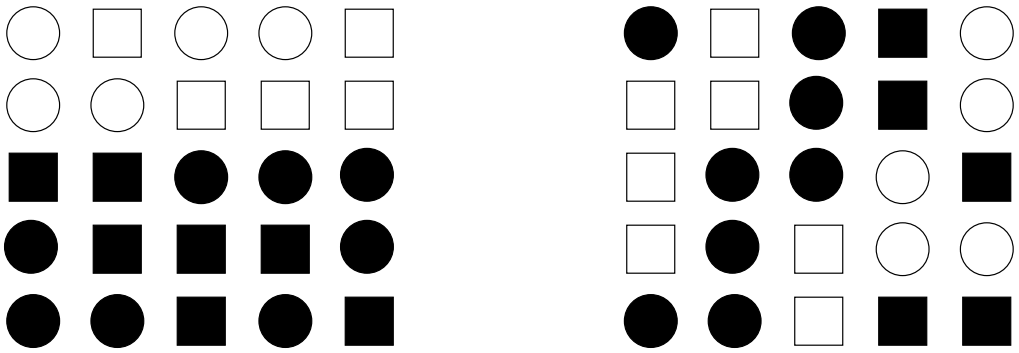
Feature	*Researchers*
line (blob) orientation	Julész & Bergen [1983]; Wolfe et al. [1992]
length	Triesman & Gormican [1988]
width	Julész [1985]
size	Triesman & Gelade [1980]
curvature	Triesman & Gormican [1988]
number	Julész [1985]; Trick & Pylyshyn [1994]
terminators	Julész & Bergen [1983]
intersection	Julész & Bergen [1983]
closure	Enns [1986]; Triesman & Souther [1985]
colour (hue)	Nagy & Sanchez [1990, 1992]; D'Zmura [1991]; Kawai et al. [1995]; Bauer et al. [1996]
intensity	Beck et al. [1983]; Triesman & Gormican [1988]
flicker	Julész [1971]
direction of motion	Nakayama & Silverman [1986]; Driver & McLeod [1992]
binocular lustre	Wolfe & Franzel [1988]
stereoscopic depth	Nakayama & Silverman [1986]
3-D depth cues	Enns [1990]
lighting direction	Enns [1990]

Table 2.1: A list of two-dimensional features that "pop out" during visual search, and a list of researchers who describe preattentive tasks performed using the given feature.

- *counting*, where users attempt to count or estimate the number of elements in a display with a unique visual feature

2.1 Feature Integration Theory

Triesman has provided some exciting insight into preattentive processing by researching two important problems [Triesman, 1985]. First, she has tried to determine which visual properties are detected preattentively. She calls these properties "preattentive features". Second, she has formulated a hypothesis about how the human visual system performs preattentive processing.



(a) (b)

Figure 2.2: Examples of boundary detection: (a) the horizontal boundary between two groups (empty objects on the top, filled objects on the bottom) is preattentively detected because each group contains a unique feature; (b) the vertical boundary is not apparent (filled circles and empty squares on the left, empty circles and filled squares on the right), because both groups use the same features (filled versus empty and square versus circle)

Triesman ran experiments using target and boundary detection to classify preattentive features. For target detection, subjects had to determine whether a target element was present or absent in a field of background distractor elements. Boundary detection involved placing a group of target elements with a unique visual feature within a set of distractors to see if the boundary could be preattentively detected.

Researchers test for preattentive target detection by varying the number of distractors in a scene. If search time is relatively constant and below some chosen threshold, independent of the number of distractors, the search is said to be preattentive. Similarly, for boundary detection, if users can classify the boundary within some fixed exposure duration, the feature used to define the boundary is said to be preattentive. A common threshold time is 200 to 250 msec, because this allows subjects "one look" at the scene. The human visual system cannot decide to change where the eye is looking within this time frame.

Triesman and others have used their experiments to compile a list of features that can

be preattentively detected (Table 2.1). It is important to note that some of these features are asymmetric. For example, a sloped line in a sea of vertical lines can be detected preattentively. However, a vertical line in a sea of sloped lines cannot be detected preattentively. Another important consideration is the effect of different types of background distractors on the target feature. These kinds of factors must be addressed when trying to design display techniques that rely on preattentive processing.

Triesman breaks low-level human vision into a set of feature maps and a master map of locations in an effort to explain preattentive processing. Each feature map registers activity in response to a given feature. Triesman proposes a manageable number of feature maps, including one for each of the opponent colour primaries green, red, yellow, and blue, as well as separate maps for orientation, shape, texture, and other preattentive features.

When the human visual system first sees an image, all the features are encoded in parallel into their respective maps. One can check to see if there is activity in a given map, and perhaps get some indication of the amount of activity. The individual feature maps give no information about location, spatial arrangement, or relationships to activity in other maps.

The master map of locations holds information about intensity or hue discontinuities at specific spatial locations. Focused attention acts through the master map. By examining a given location, one automatically gets information about all the features present at that location. This is provided through a set of links to individual feature maps (Figure 2.3).

This framework provides a general hypothesis that explains how preattentive processing occurs. If the target has a unique feature, one can simply access the given feature map to see if any activity is occurring. Feature maps are encoded in parallel, so feature detection is almost instantaneous. A conjunction target cannot be detected by accessing an individual feature map. Activity there may be caused by the target, or by distractors that share the given preattentive feature. In order to locate the target, one must search serially through

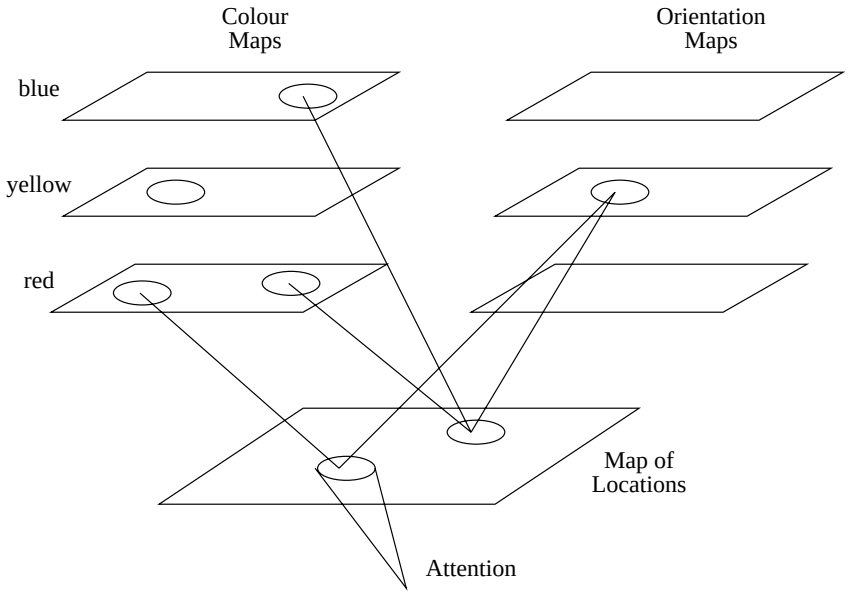


Figure 2.3: Framework for early vision that explains preattentive processing; individual maps can be accessed to detect feature activity; focused attention acts through a serial scan of the master map of locations

the master map of locations, looking for an object with the correct combination of features. This use of focused attention requires a relatively large amount of time and effort.

In later work, Triesman has expanded her strict dichotomy of features being detected either in parallel or in serial [Triesman and Gormican, 1988; Triesman, 1991]. She now believes that parallel and serial represent two ends of a spectrum. "More" and "less" are also encoded on this spectrum, not just "present" and "absent". The amount of differentiation between the target and the distractors for a given feature will affect search time. For example, a long vertical line can be detected immediately among a group of short vertical lines. As the length of the target shrinks, the search time increases, because the target is harder to distinguish from its distractors. At some point, the target line becomes shorter than the distractors. If the length of the target continues to decrease, search time decreases, because the degree of similarity between the target and the distractors is decreasing.

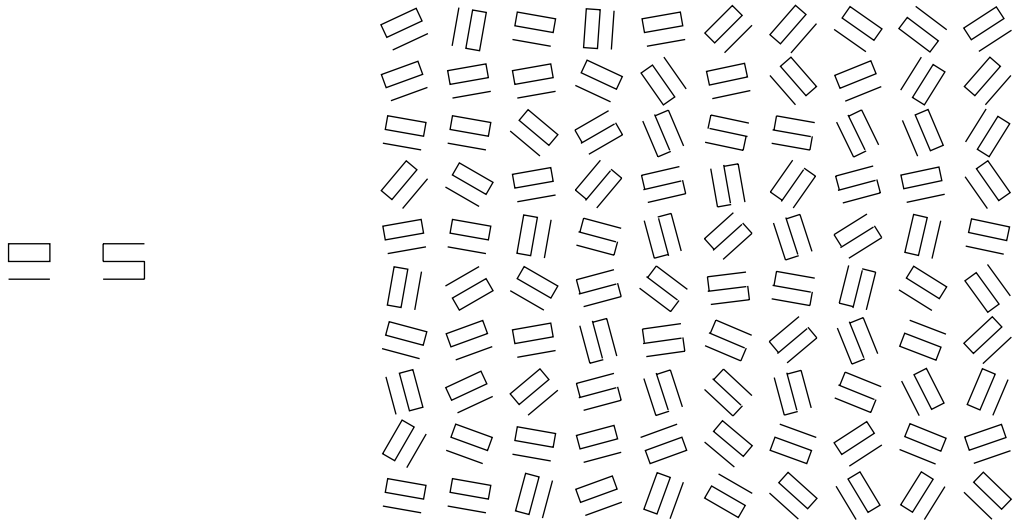
Triesman has also extended feature integration to explain certain cases where conjunc-

tion search is preattentive. In particular, conjunction search tasks involving motion, depth, colour, and orientation have been shown to be preattentive by Nakayama and Silverman [1986], Driver et al. [1992], and Wolfe et al. [1989b]. Triesman hypothesises that a significant target–nontarget feature difference would allow individual feature maps to ignore nontarget information contained in the master map. For example, consider a search for a green horizontal bar within a set of red horizontal bars and green vertical bars. This should result in conjunction search, since horizontal and green occur within each of the distractors. In spite of this, Wolfe et al. [1989b] showed that search times are independent of display size. If colour constituted a significant feature difference, the red colour map could inhibit information about red horizontal bars. Thus, the search reduces to finding a green horizontal bar in a sea of green vertical bars, which can be done preattentively.

2.2 Texton Theory

Texture segregation involves preattentively locating groups of similar objects and the boundaries that separate them. Triesman used texture segregation during her experiments with boundary detection. Figure 2.2a is an example of a horizontal texture boundary with empty shapes on the top and filled shapes on the bottom. Figure 2.2b is an example of a vertical texture boundary with filled circles and empty squares on the left, and empty circles and filled squares on the right.

Julèsz has also investigated texture perception and its relationship to preattentive processing [Julész, 1981; Julész and Bergen, 1983; Julész, 1984]. He has proposed his own hypothesis on how preattentive processing occurs. Julèsz believes that the early visual system detects a group of features called textons. Textons can be classified into three general categories:



(a) (b)

Figure 2.4: Example of similar textons: (a) two textons that appear different in isolation; (b) the same two textons cannot be distinguished in a randomly oriented texture environment

1. Elongated blobs (*e.g.,* line segments, rectangles, ellipses) with specific properties such as hue, orientation, and width

2. Terminators (ends of line segments)

3. Crossings of line segments

Julèsz believes that only a difference in textons or in their density can be detected preattentively. No positional information about neighbouring textons is available without focused attention. Like Triesman, Julèsz believes preattentive processing occurs in parallel and focused attention occurs in serial.

Figure 2.4 shows an example of an image that supports the texton hypothesis. Although the two objects look very different in isolation, they are actually the same texton. Both are blobs with the same height and width. Both are made up of the same set of line segments and each has two terminators. When oriented randomly in an image, one cannot preattentively

detect the texture boundary between the two groups of these objects.

2.3 Similarity Theory

Some researchers do not support the dichotomy of serial and parallel search modes. Initial work in this area was done by Quinlan and Humphreys [1987] . They investigated conjunction searches by focusing on two factors. First, search time may depend on the number of items of information required to identify the target. Second, search time may depend on how easily a target can be distinguished from its distractors, regardless of the presence of unique preattentive features. Triesman addressed this second factor in her later work [Triesman and Gormican, 1988]. Quinlan and Humphreys found that Triesman's feature integration theory was unable to explain the results they obtained from their experiments.

Duncan and Humphreys proceeded to develop their own explanation of preattentive processing. Their model assumes that search ability varies continuously, depending on both the type of task and the display conditions [Duncan, 1989; Müller et al., 1990]. Search time is based on two criteria: T-N similarity and N-N similarity. T-N similarity is the amount of similarity between the targets and nontargets. N-N similarity is the amount of similarity within the nontargets themselves. These two factors affect search time as follows:

- as T-N similarity increases, search efficiency decreases and search time increases
- as N-N similarity decreases, search efficiency decreases and search time increases
- T-N similarity and N-N similarity are related (Figure 2.5); decreasing N-N similarity has little effect if T-N similarity is low; increasing T-N similarity has little effect if N-N similarity is high

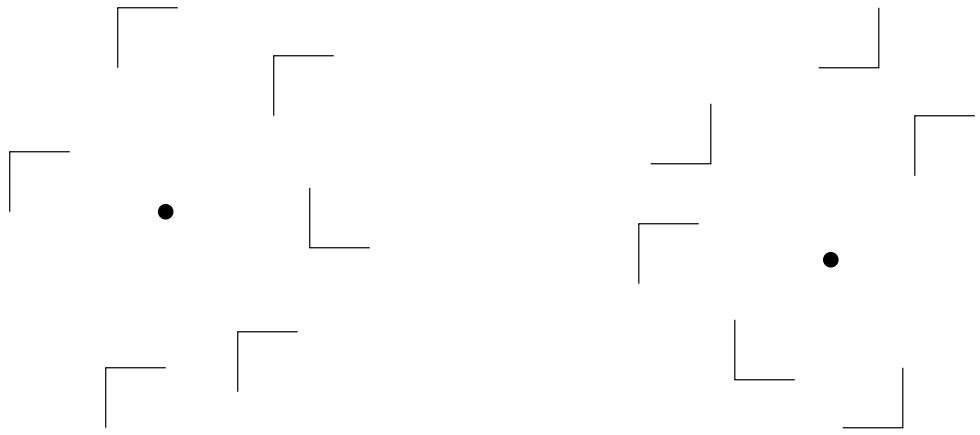


Figure 2.5: Example of N-N similarity affecting search efficiency: (a) high N-N similarity allows easy detection of target shaped like the letter L; (b) low N-N similarity increases difficulty of detecting target shaped like the letter L

Triesman's feature integration theory has difficulty explaining the results of Figure 2.5. In both cases, the distractors seem to use exactly the same features as the target, namely oriented, connected lines of a fixed length. Yet experimental results show displays similar to Figure 2.5a produce an average search time increase of 4.5 milliseconds per additional distractor, while displays similar to Figure 2.5b produce an average search time increase of 54.5 milliseconds per additional distractor.

In order to explain the above and other search phenomena, Duncan and Humphreys proposed a three-step theory of visual selection.

1. The visual field is segmented into structural units. Individual structural units share some common property (*e.g.,* spatial proximity, hue, shape, motion). Each structural unit may again be segmented into smaller units. This produces a hierarchical representation of the visual field. Within the hierarchy, each structural unit is described by a set of properties (*e.g.,* spatial location, hue, texture, size). This segmentation process

occurs in parallel.

2. Because access to visual short-term memory is limited, Duncan and Humphreys assume that there exists a limited resource that is allocated among structural units. Because vision is being directed to search for particular information, a template of the information being sought is available. Each structural unit is compared to this template. The better the match, the more resources allocated to the given structural unit relative to other units with a poorer match.

   Because units are grouped in a hierarchy, a poor match between the template and a structural unit allows efficient rejection of other units that are strongly grouped to the rejected unit.

3. Structural units with a relatively large number of resources have the highest probability of access to the visual short-term memory. Thus, structural units that most closely match the template of information being sought are presented to the visual short-term memory first. Search speed is a function of the speed of resource allocation and the amount of competition for access to the visual short-term memory.

Given these three steps, we can see how T-N and N-N similarity affect search efficiency. Increased T-N similarity means more structural units match the template, so competition for visual short-term memory access increases. Decreased N-N similarity means we cannot efficiently reject large numbers of strongly grouped structural units, so resource allocation time and search time increases.

2.4 Guided Search Theory

Jeremy Wolfe has recently suggested a visual search theory that he calls "guided search" [Wolfe and Cave, 1989; Wolfe et al., 1989; Wolfe, 1994]. He believes an activation map based on both bottom-up and top-down information is constructed during visual search. Attention is drawn to peaks in the activation map that represent areas in the image with the largest combination of bottom-up and top-down influence.

As with Triesman, Wolfe believes early vision divides an image into individual feature maps (Figure 2.6). In his theory, there is one map for each feature type (*e.g.,* one map for colour, one map for orientation, and so on). Within each map a feature is filtered into multiple categories. For example, in the colour map there might be independent representations for red, yellow, green, and blue. Wolfe has already found evidence that suggests that orientation is categorized into steep, shallow, right, and left [Wolfe et al., 1992]. The relationship between values within a feature map is different than the relationship between values from different maps (*i.e.,* the relationship between "red" and "blue" is different than the relationship between "blue" and "shallow").

Bottom-up activation follows feature categorization. It measures how different an element is from its neighbours. Differences for each relevant feature map are computed and combined (*e.g.,* how different are the elements in terms of colour, how different are they in terms of orientation?) The "metrics" used to measure differences in each feature map are still being investigated.

Top-down activation is a user-driven attempt to find items with a specific property or set of properties. For example, visual search for a blue element would generate a top-down request that activates "blue" locations. Previous work suggests subjects must specify requests in terms of the categories provided by each feature map [Wolfe et al., 1992]. Thus,

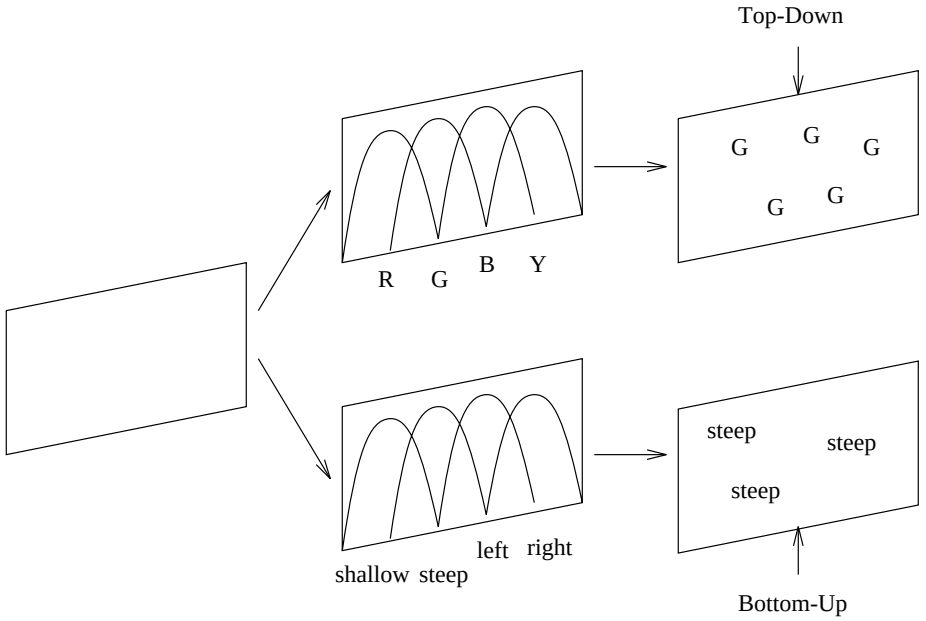


Figure 2.6: Framework for guided search, user wants to find a green steep target: image is filtered into categories for each feature map, bottom-up and top-down activation "mark" regions of the image; an activation map is built by combining bottom-up and top-down information, attention is draw to the highest "hills" in the activation map

subjects could search for "steep" or "shallow" elements, but not for elements rotated by a specific angle. Obviously, subjects should pick the category that best differentiates the target from its distractors. Finding the "best" category is often nonintuitive, however. Wolfe suggests this might explain cases where subjects' performance for a task improves over time.

The activation map is a combination of bottom-up and top-down activation. The weights assigned to these two values are task dependent. A conjunction search would place priority on top-down information, since bottom-up results are, in essence, useless. Search for a target with a unique feature would assign a high weight to bottom-up activation. Hills in the activation map mark regions that generated a relatively large amount of bottom-up or top-down influence. There is no information in the activation map about the source of a hill. High activation from a colour map looks exactly the same as high activation from an orientation map. A subject's attention is drawn from hill to hill in order of decreasing

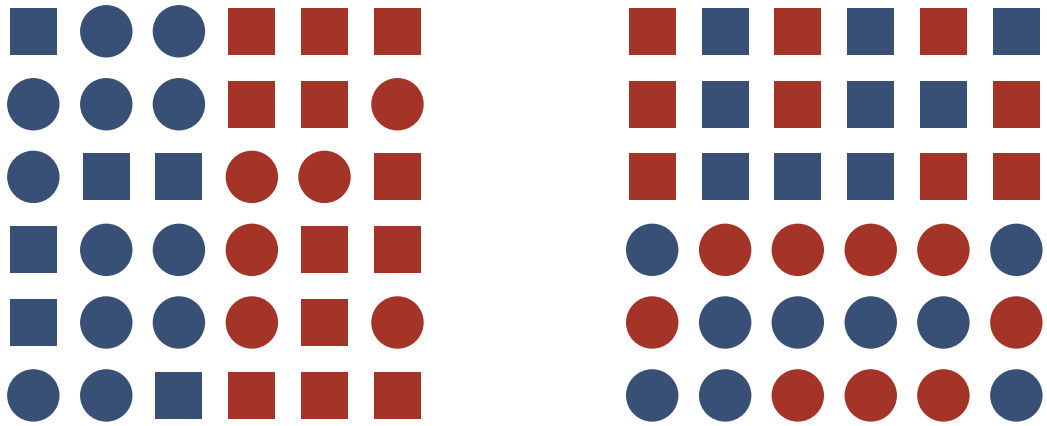
activation.

Wolfe's theory easily explains traditional "parallel" visual search. Target elements produce the highest level of activation, regardless of the number of distractor elements. This causes the target to "pop-out" of the scene in time independent of the number of distractors. This also explains Duncan and Humphreys' similarity theory results. Low N-N similarity causes distractors to report higher bottom-up activation, since they now differ from their neighbours. High T-N similarity causes a reduction in the target elements' bottom-up activation. Moreover, guided search also provides a possible explanation for situations where conjunction search can be performed preattentively [Nakayama and Silverman, 1986; Wolfe et al., 1989]. User-driven top-down activation may permit efficient searching for conjunction targets.

2.5 Interference Experiments

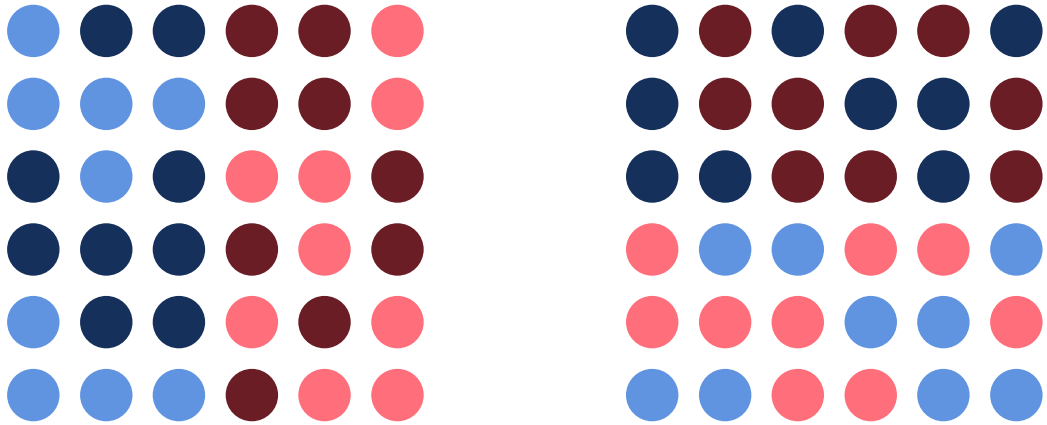
Results from preattentive processing can help to identify various types of "visual interference". These interference effects inhibit a user's low-level visual system, and should be avoided during visualization. One example is conjunction search. Visualization techniques designed to allow a user to rapidly search for data elements with a unique characteristic should ensure that the visual features chosen to represent the elements do not form a feature conjunction.

Tara Callaghan has conducted research to see how similarity within feature groups affects texture segregation [Callaghan, 1990]. She found that varying certain irrelevant features within a group can interfere with boundary detection. Her initial experiments investigated identifying a horizontal or vertical texture boundary [Callaghan, 1984]. Subjects were presented with a six by six array of elements. A texture boundary was formed by either a



(a) (b)

Figure 2.7: Form and hue segregation: (a) vertical hue boundary is preattentively detected (blue on the left, red on the right), even though form varies in both groups; (b) random hue interferes with detection of horizontal form boundary (squares on the top, circles on the bottom)



(a) (b)

Figure 2.8: Hue and brightness segregation: (a) random intensity interferes with detection of vertical hue texture boundary (blue on the left, red on the right); (b) horizontal brightness texture boundary is detected preattentively (dark elements on the top, bright elements on the bottom), even though hue varies in both groups

difference in hue or a difference in brightness. For hue segregation, the brightness in both groups varied randomly between two values. For brightness segregation, hue varied randomly between two values (Figure 2.8). Subjects had to determine whether the texture boundary was vertical or horizontal. Control experiments were run to see how quickly subjects could detect simple hue and brightness boundaries. The control arrays had a uniform brightness during hue segregation, and a uniform hue during brightness segregation.

Callaghan found that non-uniform brightness interferes with hue segregation. It took subjects significantly longer to identify the texture boundary, relative to the control array. However, a non-uniform hue did not interfere with brightness segregation. A brightness texture boundary was detected in a constant amount of time, regardless of whether hue varied or not. This asymmetry was verified through further experimentation [Callaghan, 1990].

Callaghan's more recent work has shown a similar asymmetry between form and hue [Callaghan, 1989]. As before, subjects were asked to identify a boundary as either horizontal or vertical in a six by six array. During the experiment, the arrays were segregated by either hue or form. For hue segregation, form varied randomly within the array (circle or square). For form segregation, hue varied randomly. Results showed that variation of hue interfered with form segregation, but that variation of form did not interfere with hue segregation (Figure 2.7).

These interference asymmetries suggest some preattentive features may be "more important" than others. The visual system reports information on one type of feature over and above other features that may also be present in the display. Callaghan's experiments suggest that brightness overrides hue information and that hue overrides shape information during boundary detection.

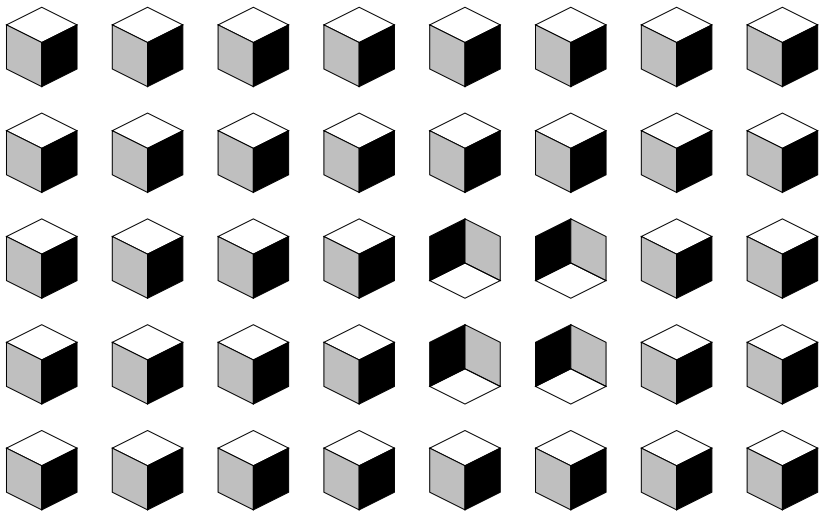
2.6 Three-Dimensional Icons

To date, most of the features identified as preattentive have been relatively simple. Examples include hue, orientation, line length, and size. Enns and Rensink have identified a class of three-dimensional elements that can also be detected preattentively [Enns and Rensink, 1990b; Enns and Rensink, 1990a]. They have shown that the element's three-dimensionality is what makes it "pop-out" of a visual scene. This is important, because it suggests that more complex high-level concepts may be processed preattentively by the low-level vision system.

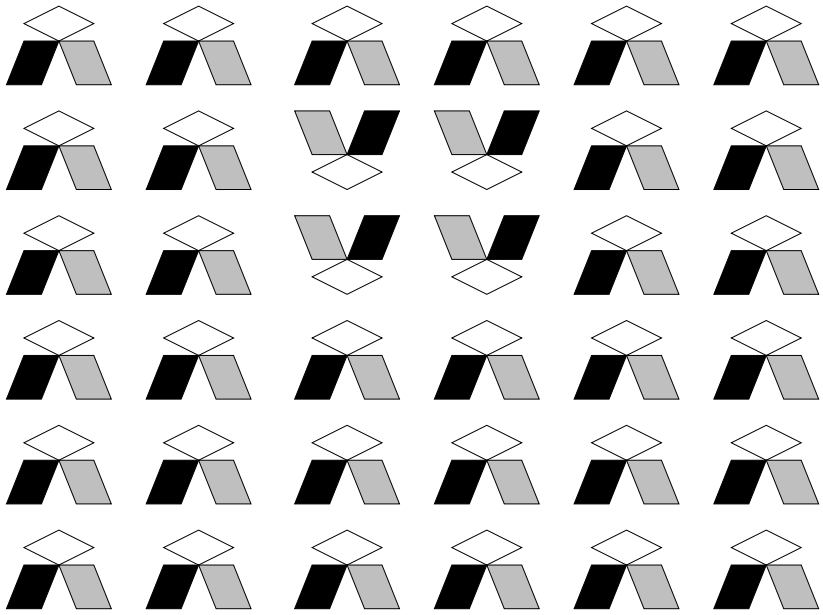
Figure 2.9 shows an example of these three-dimensional icons. The elements in Figure 2.9a are made up of three planes. The planes are arranged to form an element that looks like a three-dimensional cube. Subjects can preattentively detect the group of cubes with a three-dimensional orientation that differs from the distractors. The elements in Figure 2.9b are made up of the same three planes. However, the planes are arranged to produce an element with no apparent three-dimensionality. Subjects cannot preattentively detect the group of elements that have been rotated 180 degrees. Apparently, three-dimensional orientation is a preattentive feature.

Enns and Rensink have also shown how lighting and shadows provide three-dimensional information that is processed preattentively [Enns, 1990a; Enns, 1990b]. Spheres are drawn with shadows so they appear to be lit either from above or from below. Subjects can preattentively detect the group of spheres that appear to be lit differently than the distractors.

Three-dimensional icons are related to an area of preattentive processing that studies emergent features. An emergent feature is created by grouping several simpler shapes together. The emergent feature cannot be predicted by examining the simpler shapes in isolation (Figure 2.10). A careful choice of simple shapes will form a target element that



(a)



(b)

Figure 2.9: Three-dimensional icons: (a) when the cubes appear "three-dimensional", the 2×2 group with a different orientation is preattentively detected; (b) when three-dimensional cues are removed, the unique 2×2 group cannot be preattentively detected

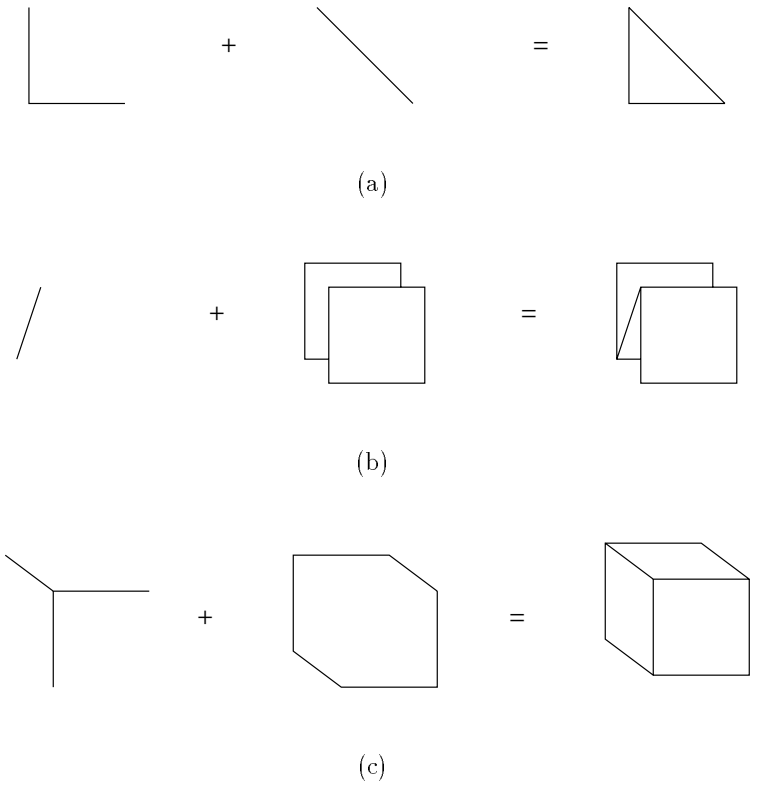
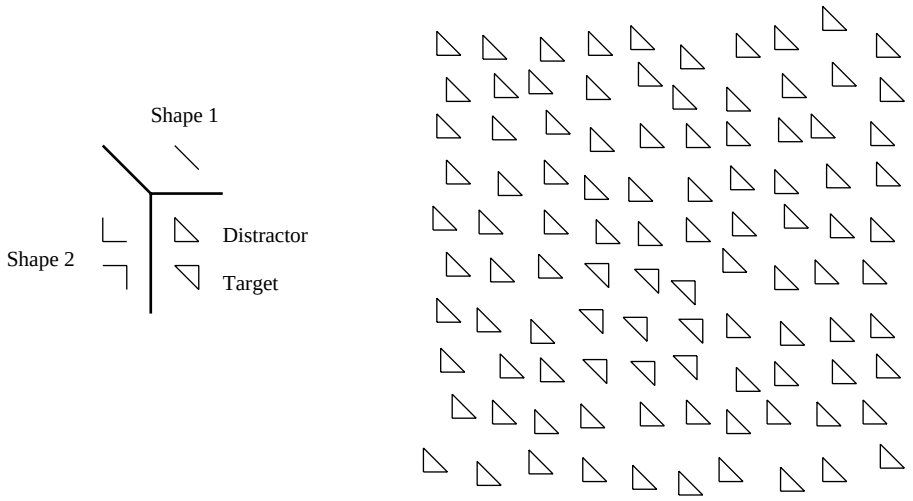


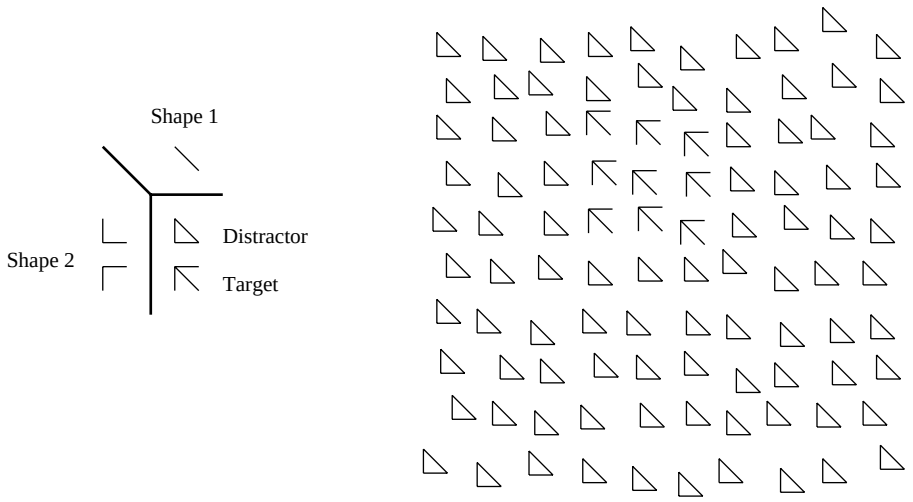
Figure 2.10: Combination of simple components to form emergent features: (a) closure, a simple closed figure is seen; (b) 3-dimensionality, the figure appears to have depth; (c) volume, a solid figure is seen

can be detected as an emergent feature. For example, in Figure 2.11a, the target element cannot be identified preattentively. However, by simply rotating one of the component elements, we create a new target with an emergent feature, non-closure, that is easily detected (Figure 2.11b).

Additional experiments by Brown confirmed that three-dimensional orientation is preattentive [Brown et al., 1992]. One explanation of visual processing is the recognition-by-components (RBC) theory proposed by Biederman. RBC suggests that objects can be decomposed into the combination of a limited number of basic volumetric shapes called geometric icons or geons. This is analogous to the idea of phonemes in speech recognition.



(a)



(b)

Figure 2.11: A proper choice of initial components will form a target with an emergent feature that can be detected preattentively: (a) the target contains no unique emergent feature, so detecting the target group is difficult; (b) the target contains a unique emergent feature, non-closure, so the target group is easily detected

Brown tested the hypothesis that geons are preattentive features, like colour or shape. His results showed that geons were not detected preattentively. It appeared that only two-dimensional feature differences were being detected by the low-level visual system. Brown did confirm, however, that target geons with a unique three-dimensional orientation "pop out" during the search task.

2.7 Motion and Depth

Initial research on motion and depth in preattentive processing was completed by Nakayama and Silverman [1986] . Their results showed that motion was preattentive. Moreover, stereoscopic depth could be used to overcome the effects of conjoin. Like the work done by Enns and Rensink, this suggests that conceptually high-level information is being processed by the low-level visual system. Initial experiments using a target detection task showed that both motion and depth are preattentive. For motion, subjects were asked to detect a target moving in a direction opposite to the distractors. Target and distractors were randomly coloured either blue or red. For depth, subjects were asked to detect a target with a different binocular disparity, relative to the distractors. Again, colour for target and distractors was randomly either red or blue. Both tasks displayed a flat response time under 250 msec as the number of distractors was increased.

A set of traditional conjoin tasks followed the initial experiments. Now, motion and colour (MC), stereo and colour (SC), or stereo and motion (SM) were combined. In MC trials distractors were either red and moving up, or blue and moving down. The target was blue and moving up. In SC trials distractors were blue on the front plane, or red on the back plane. The target was either red on the front plane or blue on the back plane. In SM trials distractors were on the front plane moving up, or on the back plane moving down.

The target was either on the front plane moving down, or on the back plane moving up.

Results showed that the MC trials were not preattentive. The response time increased by about 100 msec per ten additional distractors. However, the response times for SC and SM trials were constant and below 250 msec regardless of the number of distractors. All three sets of trials involved conjoin, because the target had no feature unique from all the distractors. However, it was found that observers could search each plane in the stereo conjoin trials preattentively, in turn. Thus, reaction times for these searches were constant over the number of distractors. The visual system can perform a parallel search in one depth plane without interference from distractors in another plane.

Other work has focused on oscillating motion and its effect on conjunction search. Driver described an experiment where subjects had to search for an X oscillating vertically among O's oscillating vertically and X's oscillating horizontally [Driver et al., 1992]. This task was preattentive if elements in each group oscillated coherently (*i.e.,* vertically oscillating stimuli moved up and down together, horizontally oscillating stimuli moved left and right together). When elements oscillated "out of phase" with one another, subjects reverted to a serial search.

Triesman, Wolfe and others have tried to explain the ability to perform these kinds of conjunction search tasks within the framework of feature maps. Triesman suggests that an important feature distinction would allow individual feature maps to ignore nontarget information from the master map. For example, if green objects could be ignored, then a search for a red X among red O's and green X's would switch from conjunction to preattentive. Green X's would be ignored, and the search would be based only on shape (a red X among red O's). Wolfe's proposal is similar, but he suggests targets are excited or emphasized over nontargets.

Driver investigated these possibilities by running additional experiments in which either

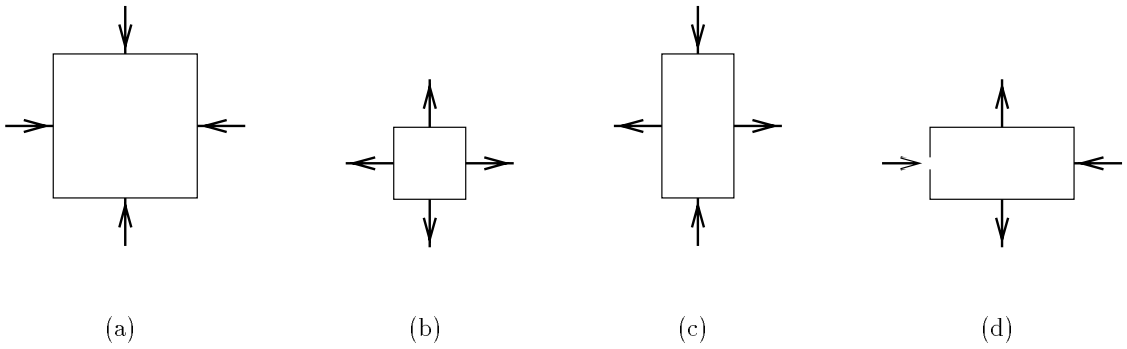


Figure 2.12: Examples of conjunctive motion, all cases are made up of the same four basic motion patterns; (a) compression; (b) expansion; (c) flat deformation; (d) compress deformation

the horizontal X's or vertical O's moved out of phase. If nontargets are inhibited, a reasonable horizontal coherence should be required. It would be difficult to inhibit a set of horizontally oscillating X's that moved out of phase. Similarly, if targets are excited, then vertically moving elements would need some kind of motion coherence. In both these cases, Driver assumed oscillating motion is the "salient feature" that marks targets and nontargets.

Results showed that moving one group out of phase made the task more difficult than when both groups moved in phase. However, it was still much easier than when both groups moved out of phase. When only the target group moved coherently, search was slightly slower than when only the distractor group moved coherently. This implies that selecting a group for search by excitation is harder than selecting a group for search by inhibition. Both operations seem to be contributing some effect, and may be working together.

Braddick and Holliday studied whether more complicated types of motion are still detected preattentively [Braddick and Holliday, 1987]. Their experiments involved detection of a target defined by differences in divergence or deformation (Figure 2.12). Divergence experiments involved squares that were either expanding or contracting. The squares increased (or decreased) in size by 15 pixels in one pixel steps, then jumped back to their original state.

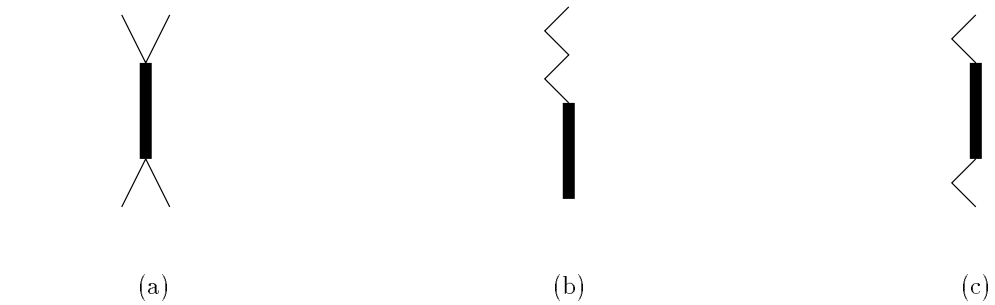
This pattern cycled repeatedly. In trials with expanding squares, the target was a single square that contracted. In trials with contracting squares, the target was a single square that expanded. Deformation experiments involved rectangles that changed their shape from long and thin to tall and skinny, again over a 15 step cycle. Targets, when present, deformed in the opposite direction.

Results showed that targets defined by either divergence or deformation required serial searching. Braddick and Holliday confirmed their experimental method by running a final set of trials. These were exactly like the divergence experiments, except that only a single side of each square was displayed. This meant the experiment involved the detection of differences in direction of motion, a task known to be preattentive, within the context of their experiment design. Results from these trials showed preattentive search times similar to those reported by Nakayama and others. Braddick and Holliday concluded that deformation and divergence represent a conjunction task. Motion itself is a preattentive feature, but deformation and divergence involve targets with a number of different motions, each of which is shared by the distractors. Thus, serial searching is required to detect the presence or absence of the target.

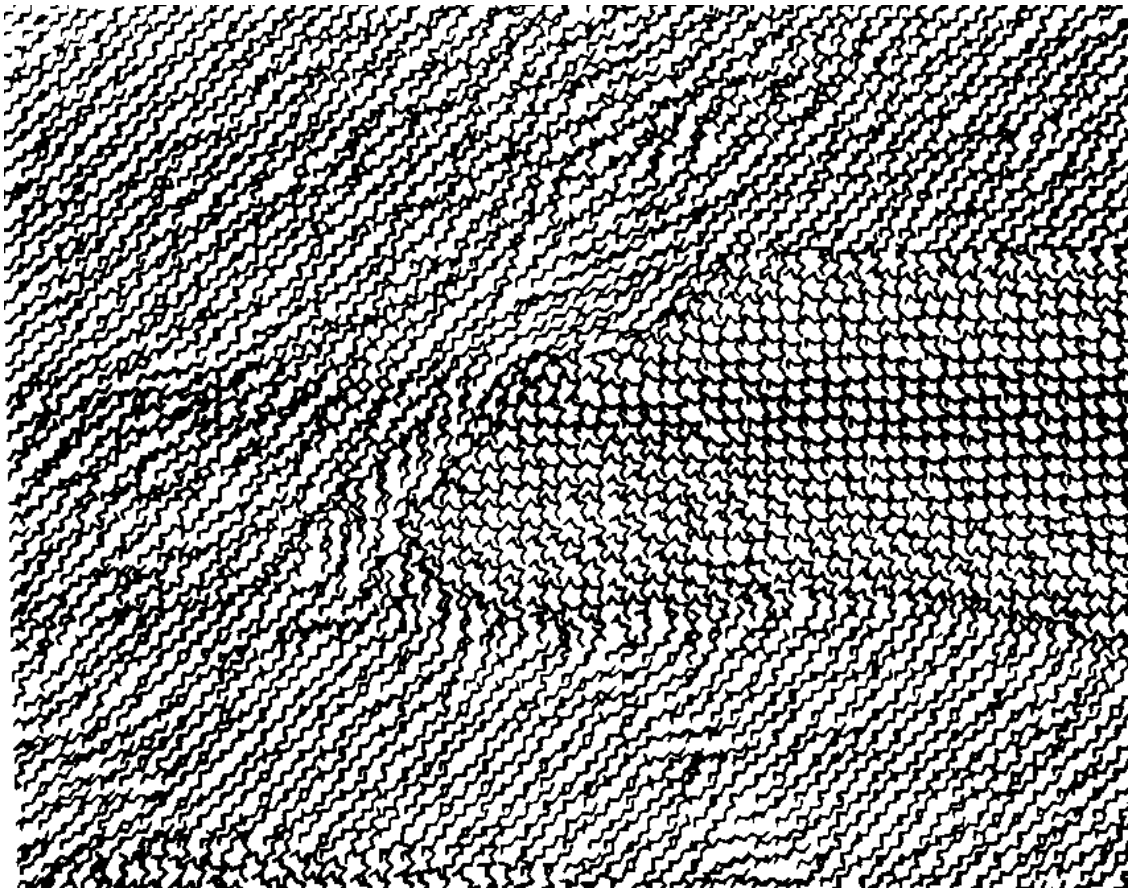
2.8 Iconographic Displays

Pickett and Grinstein have been working to develop a method of displaying multidimensional data in a two-dimensional environment [Pickett and Grinstein, 1988]. The two most common display mediums, computer screens and printed documents, are both two-dimensional.

Initially, work has focused on spatially or temporally coherent data sets. This type of data generally contains clusters of data elements with similar values. Previously, data with up to three dimensions was plotted in colour. Each data dimension controlled the intensity



(a) (b) (c)



(d)

Figure 2.13: Examples of "stick-men" icons: all three icons have four limbs and one body segment (shown in bold), so they can support five-dimensional data elements; (d) an iconographic display of 5-D weather satellite data from the west end of Lake Ontario

of one of the three primary colours red, green, and blue. Coherent areas within the data set occurred where colour values were similar. Relationships between data elements were shown as spatial changes in colour. Pickett decided to use texture as a medium that could show relationships among higher dimensional data. The texture segregations would preattentively display areas with similar data elements. Researchers could then decide quickly whether further analysis was required.

Pickett developed icons that could be used to represent each data element. An icon consists of a body segment plus a number of limbs (Figure 2.13). Each value in the data element controls the angle of one of the limbs. The icons in Figure 2.13 can support five-dimensional data elements. The four limbs support the first four dimensions. The final dimension controls the orientation of the icon's body in the image.

Once the data-icon mapping is defined, an icon can be produced for each data element. These icons are then displayed on a two-dimensional grid in some logical fashion. The result is an image that contains various textures that can be detected preattentively. Groups of data elements with similar values produce similar icons. These icons, when displayed as a group, form a texture pattern in the image. The boundary of this pattern can be seen, because icons outside the given group have a different form and produce a different texture pattern.

The key to this technique is designing icons that, when displayed, produce a good texture boundary between groups of similar data elements. To this end, Pickett and Grinstein have been working on an icon toolkit, which allows users to design and test a variety of icons with their datasets [Grinstein et al., 1989]. They have also added an audio component to the icons. Running the mouse across an image will produce a set of tones. Like the icons, the tones are mapped to values in each data element. It is believed these tones will allow researchers to detect interesting relationships in their data.

(a) (b)

Figure 2.14: Examples of Chernoff faces: the various facial characteristics, such as nose length, eyes, mouth, jowls, etc., are controlled by various data values in each face

Other researchers have suggested using various types of "icons" to plot individual data elements. One of the more unique suggestions has been to use faces (Figure 2.14) with different expressions to represent multidimensional data [Chernoff, 1973; Bruckner, 1978].

Each data value in a multidimensional data element controls an individual facial characteristic. Examples of these characteristics include the nose, eyes, eyebrows, mouth, and jowls. Chernoff claims he can support data with up to eighteen dimensions. He also claims groupings in coherent data will be drawn as groups of icons with similar facial expressions. This technique seems to be more suited to summarizing multidimensional data elements, rather than segmenting them. Still, it shows that researchers are exploring a wide variety of different ideas.

Chapter 3

Preattentive Estimation

Researchers in psychology have identified a number of tasks that can be performed preattentively. These include target detection, boundary detection, and region tracking. Some research has also been done on counting and estimation in preattentive processing. Carol Varey describes experiments where subjects are asked to estimate the relative frequency of white or black dots [Varey et al., 1990]. Subjects were asked to judge "percentage" of white dots, "percentage" of black dots, "ratio" of black dots to white dots, and "difference" between the number of black and white dots. Her survey of past research shows that all four methods have been separately proposed as the (only) method used by subjects to perform relative estimation. Varey hypothesised that subjects were in fact capable of using any of the four methods; her results showed subjects used the operation they were told to use. She also found subjects consistently overestimated small proportions, and underestimated large proportions.

We extended work on relative frequency to investigate a subject's ability to preattentively estimate percentages [Healey et al., 1993; Healey et al., 1996]. This was done by addressing two specific questions about preattentive estimation:

- *Question 1:* Is it possible for subjects to rapidly and accurately estimate the relative number of elements in a display within the constraints of preattentive vision? Under

what conditions is this possible for the well-studied features of hue and orientation?

- *Question 2:* Does encoding an independent data dimension with a task-irrelevant feature interfere with a subject's estimation ability? If so, which features interfere with one another and which do not?

Results from the experiments confirmed that rapid and accurate numerical estimation can be performed on large multi-element displays using either hue or orientation. The absence of any significant interference effects in both cases (*i.e.,* during both estimation of hue and estimation of orientation) suggests that it is indeed possible to develop effective multidimensional visualization tools for numerical estimation based on these features.

One limitation of the original investigation was its use of a fixed exposure duration and feature difference. We conducted two additional experiments designed to address the following questions:

- How are hue and orientation estimation affected by varying display duration?
- How are hue and orientation estimation affected by varying feature difference?
- What is the tradeoff between display duration and feature difference for the estimation task?

Results from these experiments provide important boundary conditions on display duration and feature difference. They also show the effect of varying both display duration and feature difference at the same time.

3.1 Salmon Migration Simulations

The experimental displays we tested were motivated by the need to examine data generated from salmon migration simulations being run in the Department of Oceanography at the University of British Columbia [Thomson et al., 1992; Thomson et al., 1994]. Salmon are a well-known fish that are found, among other areas, on the western coast of Canada. The life history of a salmon consists of four stages: birth, freshwater growth stage, ocean growth stage, and migration and spawning. Salmon are born as fry in freshwater rivers and streams. After birth, the fry spend time feeding and maturing before swimming downstream to the open ocean. Upon reaching the ocean, the salmon moves to its "open ocean habitat", where it spends most of its ocean life feeding and growing. For example, sockeye salmon are thought to feed in the Subarctic Domain, an area of the Pacific Ocean north of 40° latitude stretching from the coast of Alaska to the Bearing Sea. After a period of one to six years, salmon begin their return migration. This consists of an open ocean stage where they swim back to the British Columbia coast and a coastal stage where they swim back to a freshwater stream to spawn. This is almost always the same stream in which they were born. Scientists now know salmon find their stream of birth using smell after they reach the coast.

The ocean phase of salmon migration is not as well understood. It is recognized that it is rapid, well directed, and well timed. Previous work has examined the climate and ocean conditions during migration to see how they affect the position where Fraser River salmon arrive at the British Columbia coast (hereafter point of landfall). The entrance to the Fraser River is located on the southwest coast of B.C., near Vancouver (Figure 3.1). Generally, if the Gulf of Alaska is warm, salmon will make their point of landfall at the north end of Vancouver Island and approach the Fraser River primarily via a northern route through the Johnstone Strait. When the Gulf of Alaska is colder, salmon are distributed further south, making landfall on the west coast of Vancouver Island and approaching the Fraser

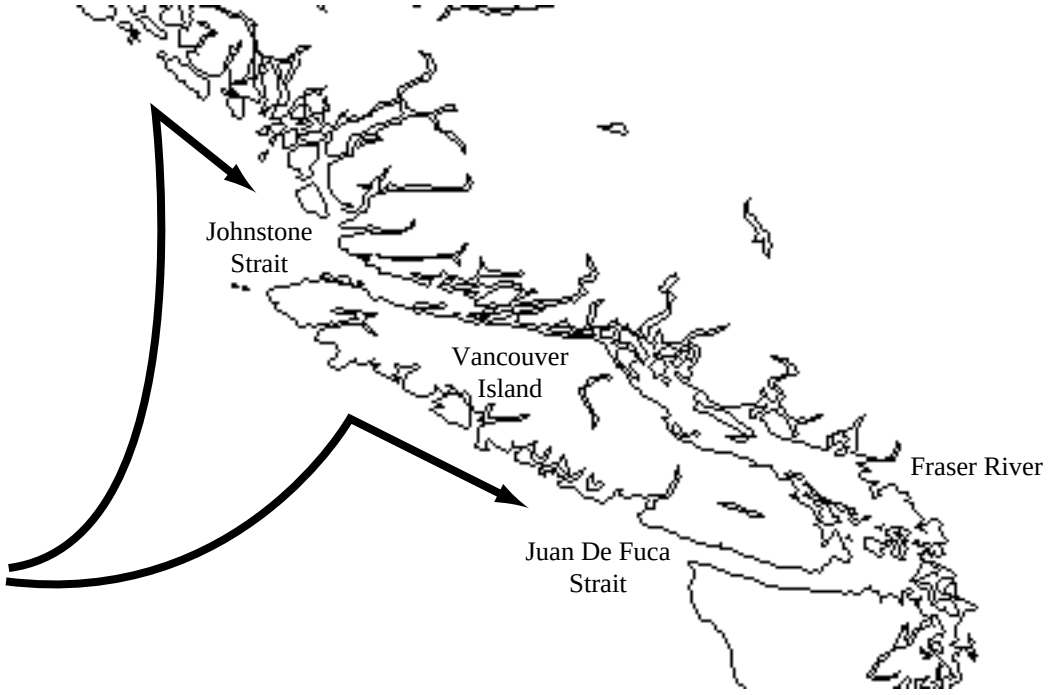
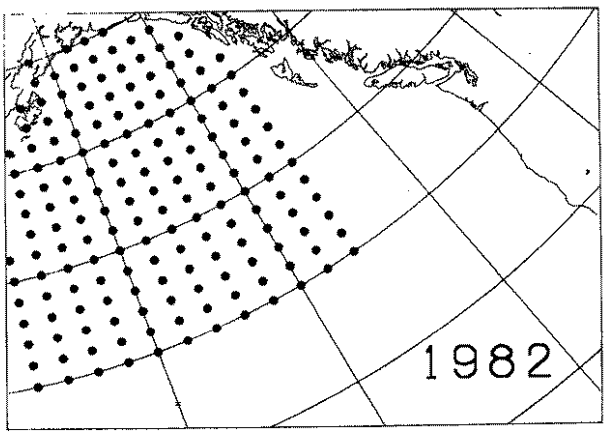


Figure 3.1: The British Columbia coast, showing Vancouver Island, the Juan de Fuca Strait, the Johnstone Strait, and the Fraser River. Arrows represent the two possible migration paths for returning salmon.

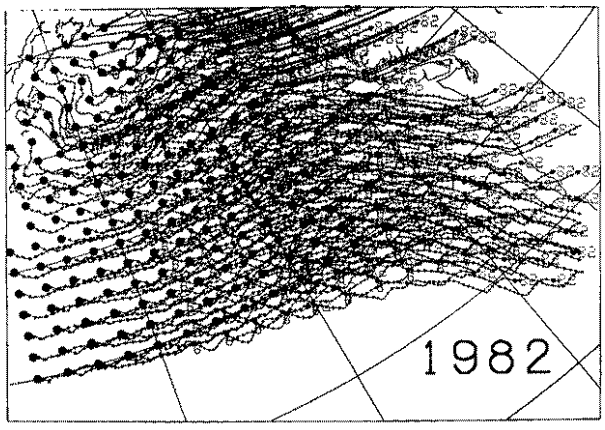
River primarily via a southern route through the Juan De Fuca Strait. Research is being conducted to determine the factors that drive this interannual variability.

Recent work in plotting ocean currents has provided scientists with a possible explanation for salmon migration patterns. It has been speculated that the variability of ocean currents has impact on where the salmon make their point of landfall. A multi-institutional investigation has been initiated to examine the influences of currents on open ocean return migrations of salmon using the Ocean Surface Circulation Simulation (OSCURS) model [Ingraham and Miyahara, 1988; Ingraham and Miyahara, 1989]. OSCURS can simulate accurately ocean currents in the North Pacific for any day during the years 1945 to 1990.

Researchers in oceanography simulate possible return migration paths of the fish by placing 174 simulated salmon at fixed locations in an OSCURS ocean model (Figure 3.2a).



(a)



(b)

Figure 3.2: Examples of output from OSCURS program: (a) dots represent starting positions of 174 simulated salmon; (b) a trailer beginning at each salmon's starting position tracks its path to the British Columbia coast.

The simulated salmon use a "compass-oriented" rule to find their way back to the British Columbia coast. Salmon take a single "look" before their migration run to determine the direction to the coast. They use a biological compass to swim in this fixed direction, regardless of external forces (*i.e.,* ocean currents) that may shift their migration path. OSCURS is used to apply daily ocean currents to the fish as they move towards the coast. Oceanographers record the point of landfall for each salmon (Figure 3.2b). These are compared with the actual distribution of fish for the given year (provided by the Department of Fisheries and Oceans) to test the accuracy of the simulation.

Task Selection

We consulted with researchers in Oceanography to identify a suitable task for our original experiments. We wanted to choose a task that was common during visualization, but that still addressed a question of potential interest to the oceanographers. Part of this involved using results from the salmon migration simulations during our experiments. Oceanographers hypothesize that a strong northerly current flow will drive most salmon to a point of landfall north of Vancouver Island. It follows that the majority of the fish will pass through the Johnstone Strait to arrive at the Fraser River. This suggests that salmon migration routes can be predicted in part by studying prevailing ocean current patterns. We decided to ask subjects to estimate the percentage of simulated salmon whose point of landfall was north of some fixed latitude, since this was one of the visualization tasks being performed by the oceanographers. Subjects were not informed that the data represented salmon migration results. They were simply asked to estimate the number of elements with a given visual feature.

Relative point of landfall (either north or south of the fixed latitude) was encoded on a two-dimensional map of the open ocean at the spatial position where the salmon started its

migration run. A preattentive feature was used to represent relative landfall. For example, during one experiment salmon that landed north of the fixed latitude were coloured blue, while salmon that landed south were coloured red. Subjects were then asked to estimate the percentage of blue elements.

A second question we wanted to investigate was how preattentive features interfered with one another. Callaghan's experiments showed an interference effect during a boundary detection task. We wanted to see if a similar effect existed when subjects were performing estimation. We decided to use stream function as our "interference" attribute, in part because the oceanographers are interested in visualizing these values during their analysis. Given the stream function $\phi(x, y)$, the x and y components of the ocean current vector can be calculated as $V_x = \frac{\partial \phi}{\partial x}$ and $V_y = \frac{\partial \phi}{\partial y}$. The stream function is a scalar function whose gradient is the current (*i.e.,* the stream function is the potential function for the vector-valued current field). In our experiments, stream function values were divided into two groups (low and high) and encoded at each spatial location where a salmon started its migration run. A second preattentive feature was used to represent stream function.

3.2 Original Estimation Experiments

Our original experiments were designed to investigate numerical estimation using either hue or orientation. Two unique orientations were used, 0° rotation and 60° rotation. Two different hues that corresponded roughly to red and blue were chosen from the Munsell colour space. This allowed us to display two-dimensional data elements as coloured, oriented rectangles (Figure 3.3). We chose a pair of hues that satisfied the following two properties:

- *Property 1*: The perceived intensity of the two hues was equal (*i.e.,* the hues were isoluminant).

- *Property 2*: The perceptual discriminability between the two hues was equal to the perceptual discriminability of a rectangle rotated 0° and one rotated 60°.

The method described by Healey et al. [1996] was used to choose a red hue and a blue hue that met these requirements. Our design allowed us to display data elements with two dimensions encoded using hue and orientation. Both dimensions were two-valued (encoded using red and blue or 0° and 60°). This is a very simple example of the general multidimensional visualization problem.

Data displayed during the experiments were taken from oceanography's salmon migration simulations. We used the following data variable to visual variable mappings:

- the longitude on which a salmon started its migration run controlled the $x-$position of the icon representing the salmon
- the latitude on which a salmon started its migration run controlled the $y-$position of the icon representing the salmon
- the point of landfall controlled either the hue or the orientation of the icon representing the salmon (the mapping depended on the mapping condition being tested)
- the stream function value controlled either the orientation or the hue of the icon representing the salmon (the mapping depended on the mapping condition being tested)

Our experiment was divided into four mapping conditions run in separate blocks of trials. The task-relevant dimension (*i.e.,* the dimension on which the percentage estimate was based) varied from mapping condition to mapping condition, as did the task-relevant feature (*i.e.,* the feature used to encode the task-relevant dimension). This gave us the following design:

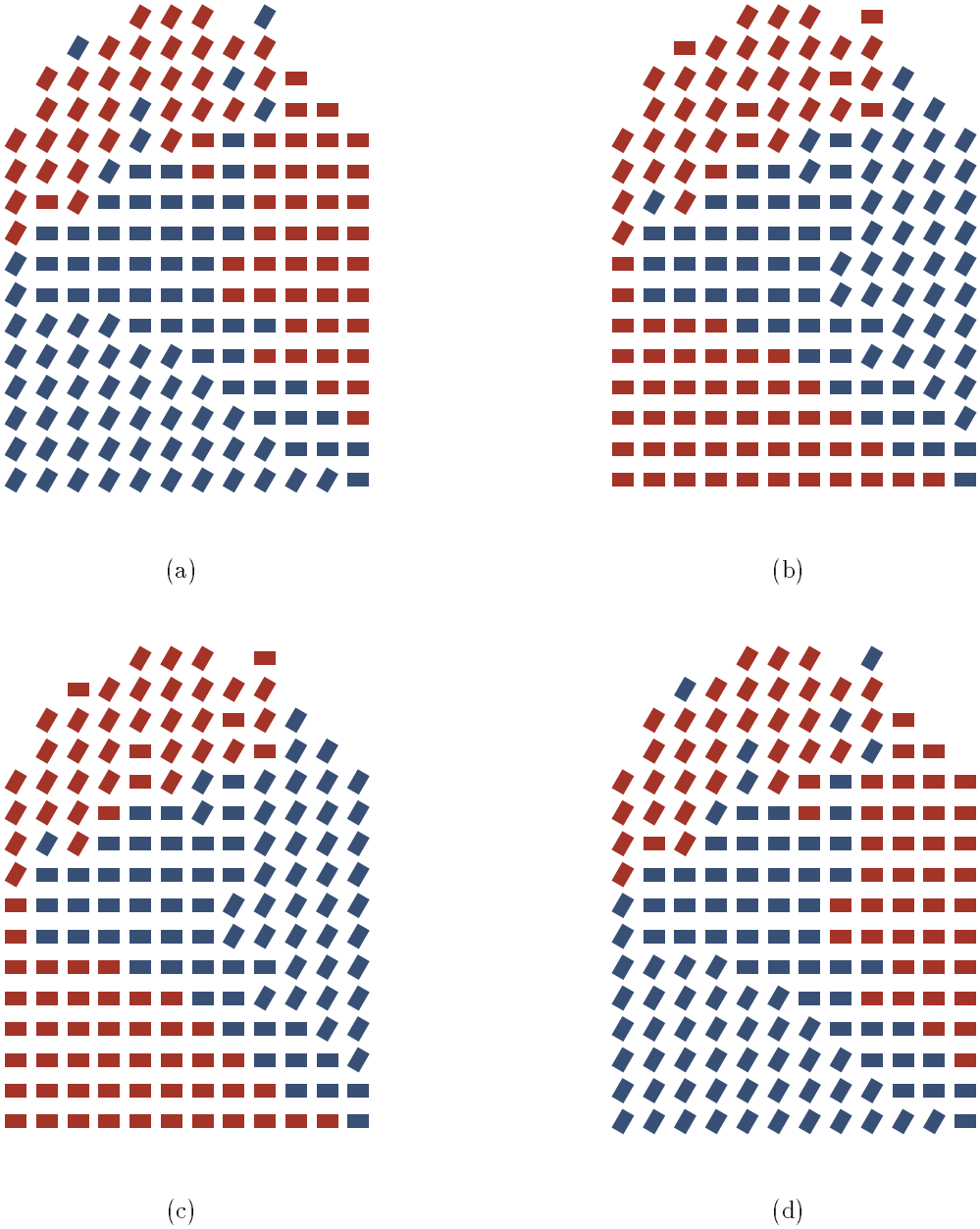


Figure 3.3: Examples of a single data frame from each of the four experimental conditions (in each frame 58% of the rectangles are targets): (a) condition B_1 (landfall represented by hue), user estimates the percentage of elements coloured blue; (b) condition B_2 (landfall represented by orientation), user estimates the percentage of elements rotated 60°; (c) condition B_3 (stream function represented by hue), user estimates the percentage of elements coloured blue; (d) condition B_4 (stream function represented by orientation), user estimates the percentage of elements rotated 60°

- Mapping condition Landfall-Hue: The task-relevant data dimension was point of landfall, represented by hue; the task-irrelevant data dimension was stream function, represented by orientation (Figure 3.3a).

- Mapping condition Landfall-Orientation: The task-relevant data dimension was point of landfall, represented by orientation; the task-irrelevant data dimension was stream function, represented by hue (Figure 3.3b).

- Mapping condition Stream Function-Hue: The task-relevant data dimension was stream function, represented by hue; the task-irrelevant data dimension was point of landfall, represented by orientation (Figure 3.3c).

- Mapping condition Stream Function-Orientation: The task-relevant data dimension was stream function, represented by orientation; the task-irrelevant data dimension was point of landfall, represented by hue (Figure 3.3d).

For each trial in the experiment, subjects were shown a display similar to Figure 3.3 for 450 msec. The screen was then cleared, and subjects were asked to estimate the number of elements in the display with a specific feature, to the nearest 10%. For example, in mapping conditions Landfall-Hue and Stream Function-Hue subjects were asked to estimate the number of rectangles coloured blue, to the nearest 10%. In mapping conditions Landfall-Orientation and Stream Function-Orientation they were asked to estimate the number of rectangles rotated 60°.

Within a mapping condition, trials were divided equally between Constant trials, where the task-irrelevant feature was fixed to a constant value, and Variable trials, where the task-irrelevant feature varied from element to element. Better performance in Constant versus Variable trials would suggest that using a task-irrelevant feature to encode an independent data dimension interferes with estimation based on the task-relevant feature. We tested

both for orientation interfering with hue estimation and for hue interfering with orientation estimation.

The main dependent variable examined was estimation error, defined as the absolute difference between the subject's interval estimate and the correct interval containing the percentage of target elements present in the display. Results showed that rapid and accurate estimation can be performed using either hue or orientation. There was no subject preference for either feature, and no interference occurred in any of the conditions. This suggests hue and orientation can be used in a visualization tool to encode different data values independently, if an estimation task is being used.

3.3 Experiment 1: Display Duration

Our conclusions from the original experiments apply to data displayed for 450 msec. This leaves two important questions unanswered. First, at what display duration are subjects no longer able to perform accurate estimation? Second, do any feature interference effects begin to appear at shorter display durations?

In Experiment 1 display duration was randomly varied among five possible values: 15, 45, 105, 195, and 450 msec. Fifteen subjects with normal or corrected acuity and normal colour vision were tested in the following manner:

1. A blank screen was displayed for 195 msec.
2. A focus circle with diameter roughly twice the width of the rectangular elements was displayed for 105 msec.
3. The trial was shown for its display duration (one of 15, 45, 105, 195, or 450 msec).

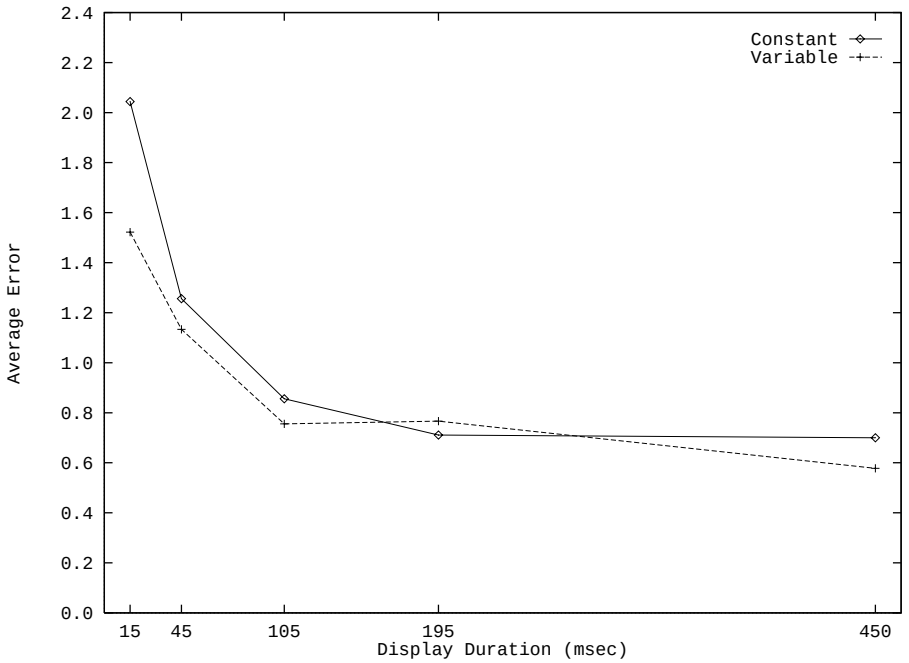


Figure 3.4: Graph of average error (absolute difference between a subject's interval estimate and the correct interval) as a function of display duration for combined results from the hue display duration experiment.

4. A "mask" of randomly oriented grey rectangles was displayed for 105 msec.

5. The screen was blanked, and subjects were allowed to enter their estimations.

Five subjects estimated the percentage of elements defined by a blue hue (mapping condition Landfall-Hue), and 10 subjects estimated the percentage of elements defined by a 60° rotation (mapping condition Landfall-Orientation). As in the original experiment, an equal number of trials was used for each interval (10 Constant and 10 Variable). Trials were split evenly among the five possible display durations, and were presented to the subjects in a random order.

We found that the minimum display duration for robust estimation using either hue or orientation lay somewhere between 45 and 105 msec. Since the original experiment had shown that estimation was relatively accurate at all percentage levels, we simplified the dependent

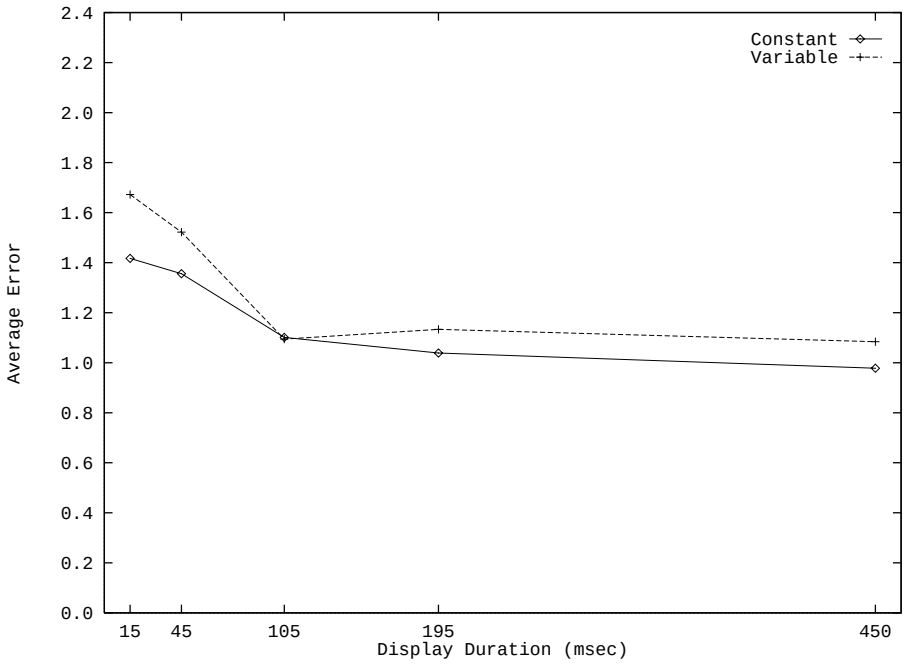


Figure 3.5: Graph of average error (absolute difference between a subject's interval estimate and the correct interval) as a function of display duration for combined results from the orientation display duration experiment.

measure by averaging error over all nine intervals. The results are shown in Figure 3.4 for hue estimation and in Figure 3.5 for orientation estimation. Inspection of these figures shows that estimation accuracy was reasonably stable at all durations of 105 msec and higher. Below that duration, error values increased rapidly. ANOVAs confirmed that accuracy varied reliably with display duration. For estimation using hue, F-values comparing mean error across display duration were significant (all p-values < 0.05) with $F(4, 445) = 19.77$, $MSe = 1.464$ and $F(4, 446) = 13.13$, $MSe = 0.979$ for Constant and Variable trials. A similar set of F-values were obtained for estimation using orientation: $F(4, 894) = 3.65$, $MSe = 1.895$, and $F(4, 894) = 7.54$, $MSe = 1.810$ for Constant and Variable trials. Fisher's Protected Least Significant Difference (PLSD) tests were computed to identify display duration pairs with significant differences in average error. As we expected, the statistical significance reflects the higher average error from the 15 and 45 msec display duration trials for both hue and

orientation estimation. During hue estimation the duration pairs (15,45), (15,105), (15,195), (15,450), (45,105), (45,195) and (45,450) were significant in both the Constant and Variable subsections. During orientation estimation the duration pairs (15,105), (15,195), (15,450) (45,195) and (45,450) were significant in the Constant subsection, and (15,105), (15,195), (15,450), (45,105), (45,195) and (45,450) were significant in the Variable subsection.

There was no evidence of feature interference for either estimation based on hue or estimation based on orientation. Results suggested that random variation in orientation did not interfere with numerical estimation based on hue. The t-values comparing mean estimation error across Constant and Variable trials had p-values greater than 0.05 at every display duration except 15 msec ($t(178) = 2.18$, $p < 0.05$, $t(178) = 0.76$, $t(178) = 0.69$, $t(178) = 0.40$, and $t(178) = 1.09$ for the display durations 15, 45, 105, 195, and 450 msec). Similar results were found when we checked to see if hue interfered with estimation based on orientation (all t-values had $p > 0.05$ with $t(358) = 1.64$, $t(358) = 1.04$, $t(357) = 0.05$, $t(357) = 0.69$, and $t(358) = 0.83$ for the display durations 15, 45, 105, 195, and 450 msec).

3.4 Experiment 2: Feature Difference

Experiment 2 was designed to address two additional questions related to numerical estimation. First, how much of a feature difference is necessary to allow accurate estimation? Second, how is this difference affected by display duration? Three mapping conditions were tested using three different hue-orientation pairs during estimation:

1. Mapping condition Small: Rectangles were drawn using two Munsell hues 5R 7/8 and 5RP 7/8, and two orientations 0° and 5°

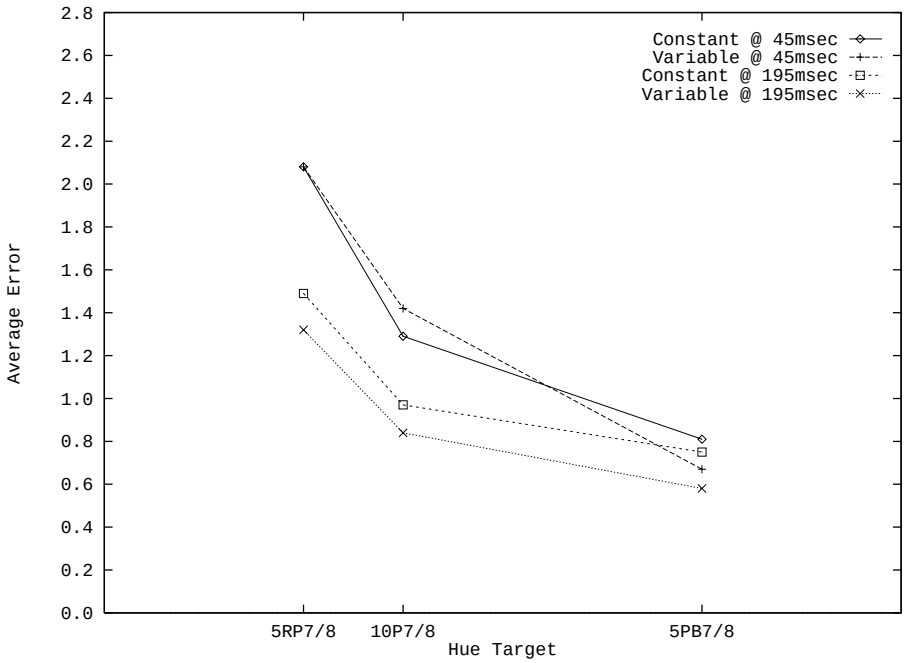


Figure 3.6: Graph of average error (absolute difference between a subject's interval estimate and the correct interval) across target hue type for combined results from hue feature difference experiment.

2. Mapping condition Medium: Rectangles were drawn using two Munsell hues 5R 7/8 and 10P 7/8, and two orientations 0° and 15°

3. Mapping condition Large: Rectangles were drawn using two Munsell hues 5R 7/8 and 5PB 7/8, and two orientations 0° and 60°

The perceptual discriminability between the hues and orientations is smallest in mapping condition Small and largest in mapping condition Large. This latter condition was essentially a replication of the hue and orientation values tested in the previous experiment. Within each mapping condition the discriminability of the two hues and two orientations were calibrated to be roughly equal, following the procedures described by Healey et al. [1993]. Trials within each mapping condition were randomly displayed at two display durations: 45 msec and 195 msec. Otherwise, the details of this experiment were identical to the previous experiment.

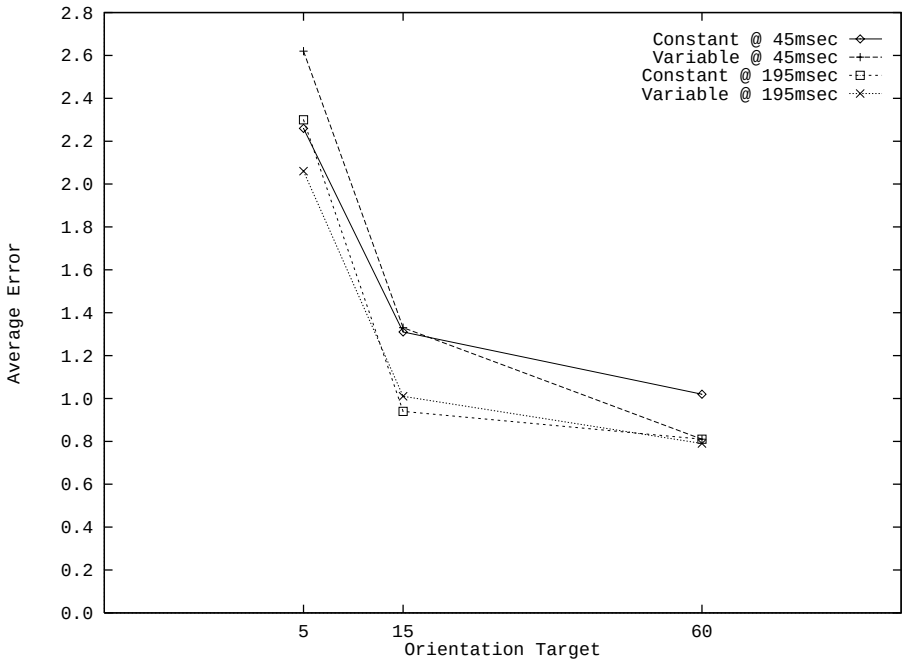


Figure 3.7: Graph of average error (absolute difference between a subject's interval estimate and the correct interval) across target orientation type for combined results from orientation feature difference experiment.

Six subjects estimated percentages based on hue. The target hue was one of 5RP 7/8, 10P 7/8, or 5PB 7/8, depending on which mapping condition a given trial belonged to. Another six subjects estimated percentages based on orientation (where target orientation was one of 5°, 15°, or 60°). Trials from the three mapping conditions were intermixed and presented to the subjects in a random order.

Subjects were able to perform accurate hue estimation at 195 msec using targets 10P 7/8 and 5PB 7/8, and at 45 msec using 5PB 7/8. Similar results were found for orientation estimation (accurate for targets oriented at 15° and 60° at 195 msec and 60° at 45 msec). Figure 3.6 graphs mean estimation error for hue trials across the three mapping conditions and both display durations. Figure 3.7 shows a similar graph for estimation based on orientation. Outside of the above cases, estimation error increased rapidly.

During hue estimation, ANOVAs comparing mean error across hue difference were significant at both exposure durations (all p-values < 0.05). F-values for Constant and Variable trials were $F(2,645) = 54.16$, $MSe = 1.579$ and $F(2,639) = 61.26$, $MSe = 1.719$ for 45 msec trials; they were $F(2,642) = 26.63$, $MSe = 1.166$ and $F(2,641) = 31.36$, $MSe = 0.964$ for 195 msec trials. During orientation estimation a similar set of F-values were found (all p-values < 0.05). For Constant and Variable trials they were $F(2,645) = 40.74$, $MSe = 2.232$ and $F(2,645) = 79.91$, $MSe = 2.198$ for 45 msec trials; they were $F(2,321) = 77.07$, $MSe = 1.929$ and $F(2,645) = 50.45$, $MSe = 1.955$ for 195 msec trials.

Finally, there was no evidence of feature interference during either hue or orientation estimation. The t-values comparing mean estimation error across Constant and Variable trials for all six display duration-target hue pairs were not significant (all p-values were greater than 0.05). Tests for hue interference during orientation estimation were also negative (for all six display duration-target orientation pairs, p-values > 0.05).

Chapter 4

Colour and Orientation

This chapter begins with brief discussion of the physical properties of colour and light. This is followed by descriptions of the four colour models used in this thesis: CIE XYZ, monitor RGB, CIE LUV, and Munsell. Methods for describing different colours in each of the models is explained. Formulas that map colours between the colour models are also provided. This allows us to convert a colour described using any of the four models into a value that can be displayed on an RGB monitor. Finally, we describe research on the use of colour, both from a scientific visualization and a preattentive processing perspective. We conclude with a review of research on two related visual features, orientation and texture. Much is already known about how to select multiple orientations for rapid and accurate search and identification in a visualization environment. We hope to build similar guidelines for the use of hue in data visualization.

Although colour is a frequently used term, its fundamental meaning is more complicated than simply stating that something is "red" or "green" or "blue". In order to properly understand and use colour, we might start by asking: What is the physical phenomenon we perceive as colour? and: How can we describe different colours in an unambiguous manner? Answers to these questions will allow us to specify and control precisely the colours we display on an RGB monitor.

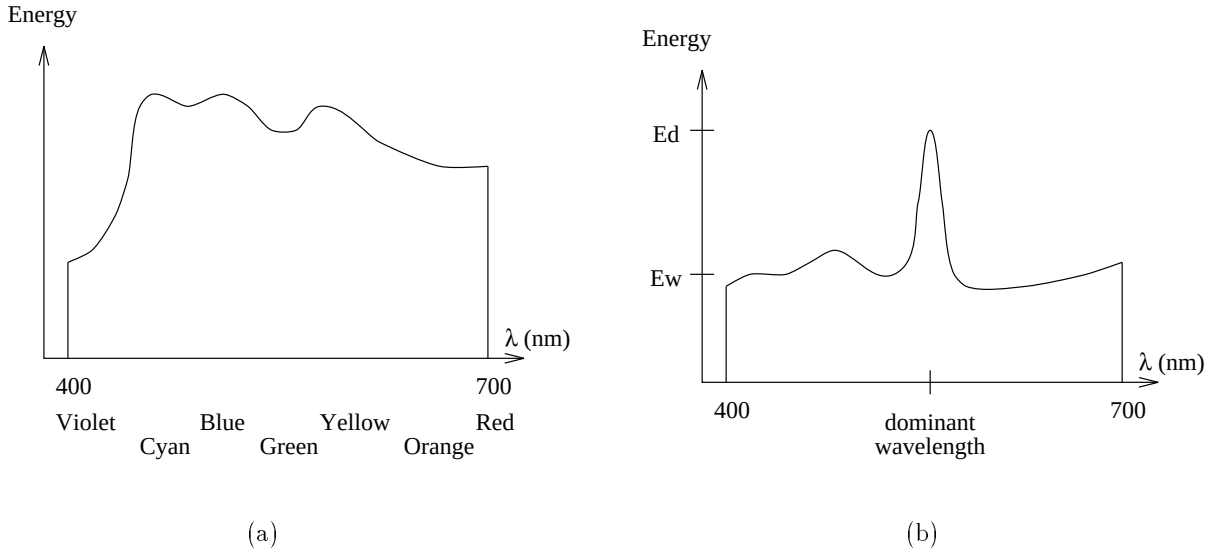


Figure 4.1: Diagram showing visible frequency domain: (a) variation in colour as wavelength λ ranges from 400 nm to 700 nm; (b) energy distribution curve with dominant wavelength somewhere around yellow, E_d is energy density at dominant wavelength, E_w is energy density of uniform white light

4.1 Properties of Colour

What we commonly call "colour" is actually our perception of light waves from a thin band of frequencies within the electromagnetic spectrum. This region of visible light ranges from about 4.3×10^{14} hertz to about 7.5×10^{14} hertz.

An individual colour can be described by providing its *dominant wavelength*, *excitation purity*, and *luminance*. The dominant wavelength is the wavelength we "see" when viewing light of the given colour. Excitation purity is related to saturation, and describes how strong (or how far from grey) the colour is. Luminance describes the intensity or brightness of the colour. We often refer to colours by their dominant wavelength λ. Using this notation, colours range from about 400 nanometres (1 nm = 10^{-7} cm) for violet to about 700 nm for red (Figure 4.1a). Consider Figure 4.1b, which shows an example energy distribution curve with a dominant wavelength somewhere around yellow. Excitation purity depends on the

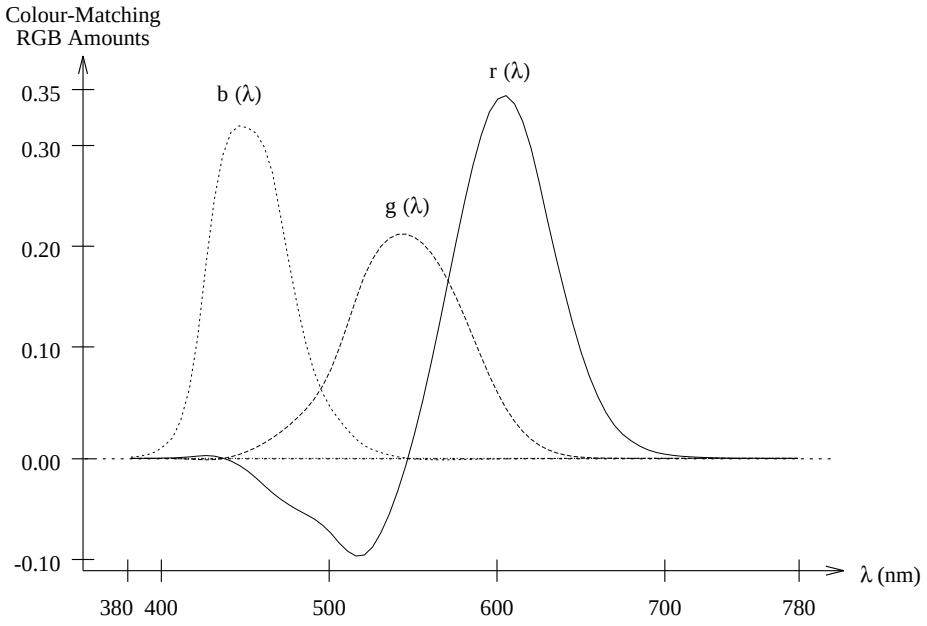
relationship between the energy distribution of the dominant wavelength E_d and the energy distribution of uniform white light E_w. If $E_d = E_w$ (0% pure), we see a fully unsaturated shade of grey. If $E_w = 0$ (100% pure), we see a fully saturated colour. Luminance is proportional to the area under the energy distribution curve.

4.2 CIE XYZ Colour Model

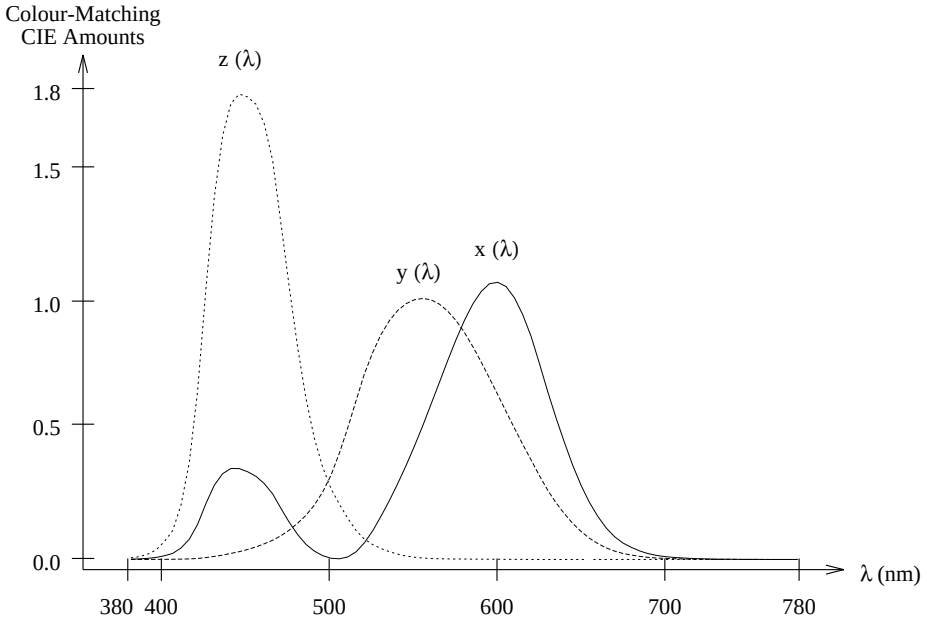
Different-coloured lights can be combined to produce a wide range of colours. One of the most commonly used methods is the combination of red, green, and blue (where **R** was chosen to be the colour with a wavelength of 700 nm, **G** at 546.1 nm, and **B** at 435.8 nm). Figure 4.2a shows the amount of **R**, **G**, and **B** needed to produce any colour in the visible frequency domain. An important point to note is that the red curve $\overline{r}(\lambda)$ is negative from 438 nm to 546 nm. Colours with a dominant wavelength in this region cannot be produced through a positive combination of **R**, **G**, and **B**.

In 1931, the Commission Internationale de L'Éclairage (CIE) addressed the problem of negative weights in the **RGB** colour matching method. They defined three new primaries, called **X**, **Y**, and **Z**, to replace **R**, **G**, and **B** during colour matching. Figure 4.2b shows the amount of **X**, **Y**, and **Z** needed to produce colours in the visible frequency domain. None of the colour matching curves are negative, which means any colour can be produced by some positive combination of **X**, **Y**, and **Z**.

Suppose the amount of **X**, **Y**, and **Z** needed to match some colour **C** is defined to be (X, Y, Z); that is, $\mathbf{C} = X\mathbf{X} + Y\mathbf{Y} + Z\mathbf{Z}$. Chromaticity values (x, y, z) are defined by normalizing over $X + Y + Z$:



(a)



(b)

Figure 4.2: Diagram showing two sets of colour matching curves: (a) curves $\overline{r}(\lambda)$, $\overline{g}(\lambda)$, and $\overline{b}(\lambda)$ show the amount of **R**, **G**, and **B** required to match the wavelength of any visible colour; (b) curves $\overline{x}(\lambda)$, $\overline{y}(\lambda)$, and $\overline{z}(\lambda)$ show the amount of **X**, **Y**, and **Z** required to match the wavelength of any visible colour

$$x = \frac{X}{X+Y+Z}\ ,\quad y = \frac{Y}{X+Y+Z}\ ,\quad z = \frac{Z}{X+Y+Z} \tag{4.1}$$

The chromaticity values (x, y, z) depend only on the dominant wavelength and excitation purity of **C**. Luminance information is lost during normalization. Since $x + y + z = 1$, we can recover z from x and y by $z = 1 - x - y$. To obtain the original (X, Y, Z) values, we also need the luminance Y. Given (x, y, Y), we can recover the corresponding (X, Y, Z) by:

$$X = \frac{x}{y}Y\ ,\quad Y = Y\ ,\quad Z = \frac{1-x-y}{y}Y \tag{4.2}$$

A plot of (x, y) values for all visible colours produces the CIE chromaticity diagram shown in Figure 4.3. Points on the boundary of the horseshoe represent fully saturated colours (*i.e.,* excitation purity of 100%). Points in the interior of the horseshoe represent a mixture of a fully saturated colour with uniform white light (*i.e.,* excitation purity of less than 100%). Colours with the same chromaticity but difference luminances project onto the same point in the horseshoe.

The chromaticity diagram can be used to define a colour gamut, which shows the range of values possible from combining two or more individual colours. Two colours R and G can be combined in various ways to produce all the colours that lie on the line RG. Three colours R, G, and B can be combined to produce all the colours that lie in the triangle RGB. Figure 4.3 shows an RGB colour gamut formed from typical phosphor values $R = (0.61, 0.35)$, $G = (0.29, 0.59)$, and $B = (0.15, 0.06)$ for an RGB monitor. A monitor that uses these phosphors can only display colours inside the RGB triangle, an area that does not include a large number of visible colours.

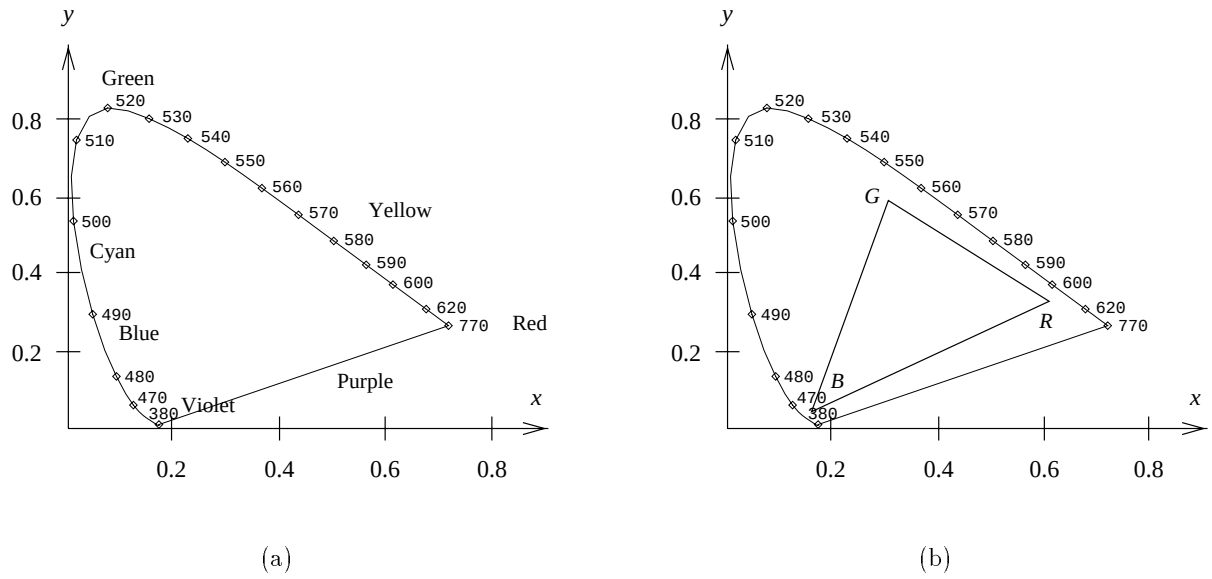


Figure 4.3: CIE chromaticity diagram, the dot marks the position of standard white: (a) wavelengths (in nanometres) and positions of saturated colours specified on the boundary of the horseshoe; (b) colour gamut formed from typical phosphor values for an RGB monitor, colours inside the triangle can be displayed by the monitor

4.3 Monitor RGB Colour Model

A monitor RGB colour model is used by most colour CRT monitors. Different amounts of the monitor's R, G, and B are added together to produce different colours. The gamut of an RGB monitor is often drawn as a unit cube, as in Figure 4.4. The monitor's full intensity red, yellow, green, cyan, blue, and magenta, along with black and white, are situated at the corners of the cube. The diagonal line from black to white represents shades of grey.

It is often useful to be able to convert colours specified in CIE XYZ to a particular monitor's RGB colour gamut. The conversion from XYZ to monitor RGB is a linear transformation. We begin by obtaining the chromaticity values (x_r, y_r), (x_g, y_g), and (x_b, y_b) of the monitor's red, green, and blue phosphors, and the luminance of the monitor's maximum-brightness red, green, and blue (Y_r, Y_g, Y_b). From these we can compute the following:

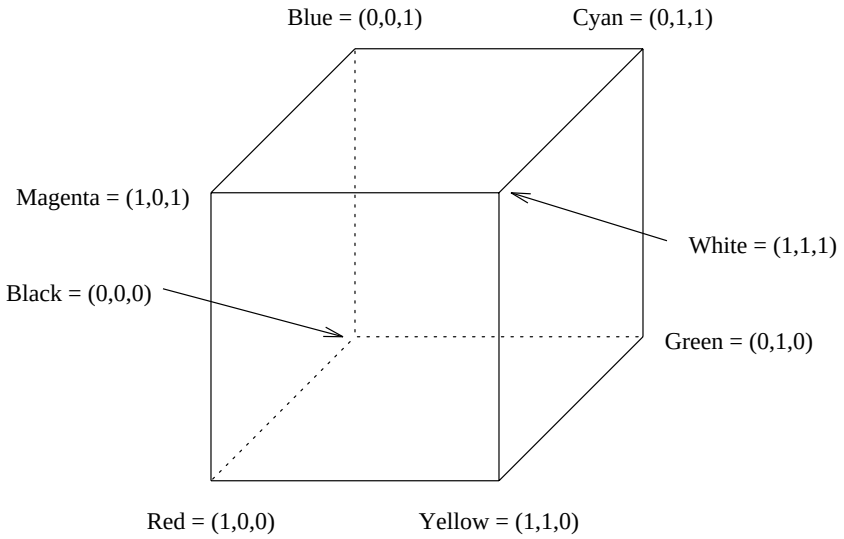


Figure 4.4: RGB colour gamut represented as a unit cube, with fully saturated colours displayed at each corner; the diagonal from the black corner to the white corner represents shades of grey of increasing intensity

$$\begin{aligned} z_r &= 1 - x_r - y_r \\ z_g &= 1 - x_g - y_g \\ z_b &= 1 - x_b - y_b \end{aligned} \tag{4.3}$$

$$\begin{aligned} C_r &= \frac{Y_r}{y_r} \\ C_g &= \frac{Y_g}{y_g} \\ C_b &= \frac{Y_b}{y_b} \end{aligned} \tag{4.4}$$

Given a colour specified as (x, y, Y), X and Z can be computed using Equation 4.2. The (X, Y, Z) values are then inserted into Equation 4.5 to obtain the monitor (R, G, B) values for the given colour.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} x_r C_r & x_g C_g & x_b C_b \\ y_r C_r & y_g C_g & y_b C_b \\ z_r C_r & z_g C_g & z_b C_b \end{bmatrix}^{-1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.5}$$

Recall that a unit cube is used to represent the monitor's gamut. Any colour with an (R, G, B) value outside the range $(0 \ldots 1, 0 \ldots 1, 0 \ldots 1)$ cannot be displayed on the given monitor. Colours with an R, G, or B value greater than 1 fall outside the luminance range of the monitor. Colours with an R, G, or B value less than 0 fall outside the chromaticity range of the monitor. Colours inside the monitor's gamut can be scaled as necessary, as long as the monitor performs proper gamma correction. For example, Silicon Graphics workstations specify RGB primaries as numbers between 0 and 255. (R, G, B) values obtained from Equation 4.5 can be multiplied by 255 and rounded to produce the primaries necessary to display a given colour on the monitor.

4.4 CIE LUV Colour Model

One problem with both the CIE XYZ and RGB models is their lack of perceptual balance. Suppose we start from some colour in CIE XYZ, say C_1, and move in any direction a distance of ΔC to colour $C_1 + \Delta C$. Starting from some other colour C_2, we can move to a new colour $C_2 + \Delta C$. The distance between the initial and final colours in both cases is ΔC. However, a human observer would not necessarily perceive the difference between the two pairs of colours to be equal. In CIE XYZ space (and in RGB space), colours that are the same distance from one another are not necessarily perceptually equidistant.

In 1976, the CIE proposed the CIE LUV space to address this problem. CIE LUV is a perceptually uniform colour space. Colours that are the same distance from one another

in CIE LUV have a perceived difference that is roughly equal. This means that distance and difference can be interchanged as required. If colours A and B are twice as far apart as colours C and D, then the perceived difference in colour between A and B is about twice the perceived difference between C and D.

The equations for computing LUV assume we have the (X, Y, Z) for the colour to convert, and (X_w, Y_w, Z_w) of a standard white. Given these values, the corresponding LUV colour is:

$$
\begin{aligned}
L^* &= \begin{cases} 116(Y/Y_w)^{1/3} - 16, & Y/Y_w > 0.01 \\ 903.3(Y/Y_w), & Y/Y_w \leq 0.008856 \end{cases} \\
u^* &= 13L^*(u' - u'_w) \\
v^* &= 13L^*(v' - v'_w) \\
u' &= \frac{4X}{X + 15Y + 3Z}, \quad v' = \frac{9Y}{X + 15Y + 3Z} \\
u'_w &= \frac{4X_w}{X_w + 15Y_w + 3Z_w}, \quad v'_w = \frac{9Y_w}{X_w + 15Y_w + 3Z_w}
\end{aligned}
\tag{4.6}
$$

L^* encodes the luminance or intensity of a given colour, while u' and v' control its chromaticity. In CIE LUV, Euclidean distance and perceived colour difference (specified in ΔE^* units) can be interchanged, since the colour difference between two colour stimuli x and y is roughly:

$$\Delta E^*_{xy} = \sqrt{(\Delta L^*_{xy})^2 + (\Delta u^*_{xy})^2 + (\Delta v^*_{xy})^2} \tag{4.7}$$

Colours in CIE LUV are normally specified as (L^*, u', v'). Conversion from (L^*, u', v') to (X, Y, Z) and then to a monitor (R, G, B) value requires solving Equation 4.6 for X, Y, and Z.

$$\begin{aligned} Y &= \left(\frac{L^* + 16}{116}\right)^3 Y_w \\ X &= \frac{9u'}{v'}Y \\ Z &= \frac{3}{v'}Y - 5Y - \frac{3u'}{4v'}Y \end{aligned} \tag{4.8}$$

By fixing L^* and varying u' and v' (or u^* and v^*), it is possible to obtain a set of colours that are both isoluminant and equidistant from one another. This technique was used during our research to control both the intensity and the perceived difference of various colour elements.

4.5 Munsell Colour Model

The Munsell colour model was originally proposed by Albert H. Munsell in 1898. It was later revised by the Optical Society of America in 1943 to more closely approximate Munsell's desire for a functional and perceptually balanced colour system. A colour from the Munsell colour model is specified using the three "dimensions" hue, chroma, and value.

In Munsell space, hue refers to some uniquely identifiable colour, or as Munsell suggested, "the quality by which we distinguish one colour from another, as a red from a yellow, a green, a blue, or a purple"[1]. Hue is represented by a circular band divided into ten sections. Munsell named these sections red, yellow-red, yellow, green-yellow, green, blue-green, blue, purple-blue, purple, and red-purple (or R, YR, Y, GY, G, BG, B, PB, P, and RP for short). Each section can be further divided into ten subsections if finer divisions of hue are needed. A number preceding the hue name is used to define the subsection (*e.g.,* 5R or 7BG).

Value refers to a colour's lightness or darkness. Munsell defined value as "the quality by

[1] *Munsell: A Grammar of Colour.* New York, New York: Van Nostrand Rienhold Company, 1969, pg. 18

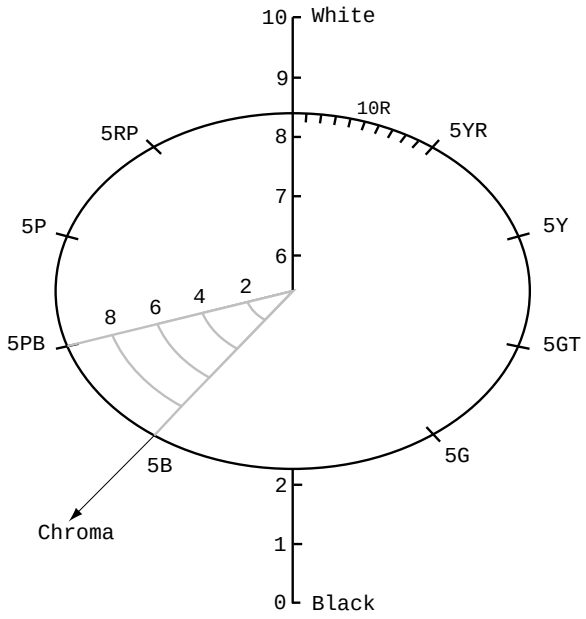


Figure 4.5: Munsell colour space, showing it's three dimensions hue, value, and chroma

which we distinguish a light colour from a dark one"[2]. Value is divided into eleven sections numbered 0 through 10. Dark colours have a low value, while lighter colours have a higher value. Value 0 represents black, and value 10 represents white.

Chroma defines a colour's strength or weakness. Chroma is measured in numbered steps starting at 1. Weak colours have low chroma values. Strong colours have high chroma values. Greys are colours with a chroma of zero. The maximum possible chroma depends on the hue and value being used.

A visual representation of the Munsell colour space is shown in Figure 4.5. The circular band represents hue. The pole running through the center of the colour space represents value. Colours with increasing chroma radiate outward from the value pole. A Munsell colour is specified by writing "hue value/chroma". For example, 5R 6/6 would be a relatively strong red. 5BG 9/2 would be a weak cyan.

[2]Ibid, pg. 20

The Munsell space provides a number of interesting and desirable properties. Munsell originally designed his colour space to be used by artists. One feature he tried to incorporate into his system was perceptual "balance". Hues directly opposite one another will be balanced, provided their value and chroma are equal. Thus, 5BG 5/5 is perceptually balanced with 5R 5/5, and 5Y 2/3 is balanced with 5PB 2/3. Opposite hues with different values and chromas can also be balanced by varying the amount of each colour used within a given area. Given two Munsell colours $H_1\,V_1/C_1$ and $H_2\,V_2/C_2$, we need V_2C_2 parts of hue H_1 and V_1C_1 parts of hue H_2. For example, colours 5R 5/10 and 5BG 5/5 can be balanced by using 5BG 5/5 in two-thirds of the area, and 5R 5/10 in one-third of the area. As we would expect, the stronger chroma and higher value take up less of the total area than the weaker chroma and lower value.

A second and perhaps more important property is that Munsell colours with the same value are isoluminant. Thus, colours 5R 5/5, 5G 5/6, 5B 5/3, and any other colours with value 5 are all perceived as having equal luminance. This property was provided when the Munsell colour table was revised in 1943.

Wyszecki and Stiles provide (X, Y, Z) values for many Munsell colours [Wyszecki and Stiles, 1982]. They divide individual hues into four subsections: 2.5, 5, 7.5, and 10. This provides a total of 40 different hues, plus nine values for each hue and chromas for every hue/value pair. Equation 4.5 can then be used to convert the (X, Y, Z) triples into monitor (R, G, B) values.

4.6 Colour in Scientific Visualization

Colour is one of the most commonly-used visual features in scientific and data visualization. Because of this, a large body of past work has studied the use of colour for a variety of

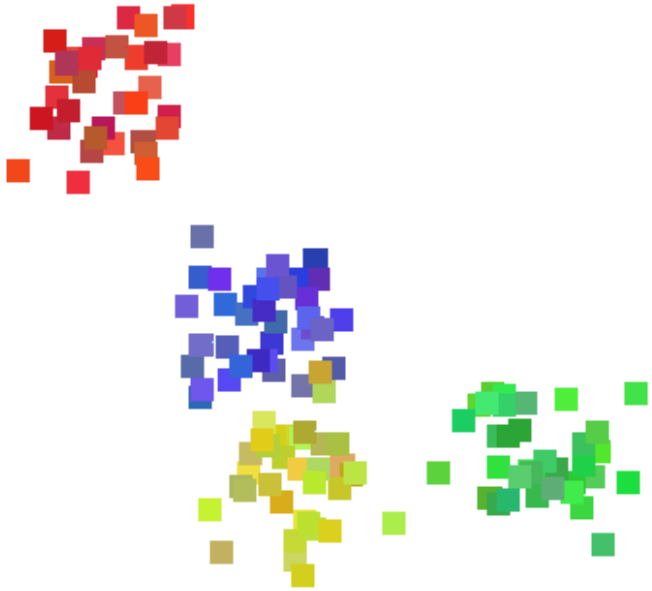


Figure 4.6: An example of Ware and Beatty's coherency visualization technique, the four "clouds" of similarly-coloured squares represent four coherent groups of data elements

visualization tasks. In many instances, colour is divided into its component parts luminance and hue, to gain greater control during its use for representing multidimensional data.

An early paper by Ware and Beatty describes using colour to detect coherence in five-dimensional data elements [Ware and Beatty, 1988]. Each of the five data attributes is mapped to one of the visual features x-position, y-position, red, green, and blue. The result is a two-dimensional display of coloured squares (one for each data element in the dataset). Groups of elements with all five attributes in common will appear as a spatial cloud of similarly-coloured squares (Figure 4.6). Ware and Beatty's tool shows coherence in a dataset as spatially coherent colour regions.

Much of the work-to-date has studied the problem of building effective "univariate colourmaps" or "colour scales" to apply to a monotonic data attribute. Such a colourmap needs to convey visually an increase (or decrease) in the attribute's value through a corresponding increase (or decrease) in perceived colour. Ideally, the relative difference between pairs of values should be displayed with an equal relative perceived difference in colour; in other words, steps of a constant size through the attribute's range of values should result in perceptually equal colour steps through the colourmap used to visualize the attribute.

A separate paper by Ware discusses the use of luminance and hue for the display of metric and form data on a continuous two-dimensional surface [Ware, 1988]. During a metric visualization task, users attempt to accurately determine an attribute's value at a specific location on the surface. Colourmaps used for this type of task must address the problem of simultaneous contrast. The perceived colour of a target patch will be affected by the colour of its neighbouring patches. Ware suggests that the amount and direction of simultaneous contrast error can be predicted by examining variations in a colourmap with respect to the visual opponent-colour channels: luminance, red-green, and blue-yellow.

In an initial experiment, Ware built a spectral colourmap that held simultaneous contrast to a minimum across its entire range. The spectral colourmap ranged through a set of colours starting at red + blue, to blue, blue + green, green, green + red, and finally to red. Ware compared his colourmap against four other common systems: linear grey-scale (linear steps in intensity), perceptual grey-scale (perceptually equal steps in intensity), saturation (a colourmap that ranges from grey to red), and red-green (a colourmap that ranges from red to green). A parabolic surface was displayed using the entire range of the colourmap being tested. A small target patch was placed at the center of the surface. A colour scale with sixteen values was provided, and subjects were asked to report the value of the target patch (which was randomly chosen from one of the twelve interior positions on the colour scale). As anticipated, Ware's spectral colourmap gave significantly lower average error when compared

to the other four colourmaps.

Ware's second visualization task, form, occurs when users try to detect different kinds of spatial patterns in their data. Ware identified five common forms: gradient, ridges, convexity and concavity, saddle, and cusp. Because of the effectiveness of the luminance channel for processing shape, motion, and depth, Ware expected the grey-scale colourmaps to perform best during the identification of form. Subjects were shown one of the common forms using all five colourmaps. Then were then asked to rate the effectiveness of each colour map from 0 (poor) to 4 (good). Results showed that the grey-scale colourmaps were judged to be significantly better at showing gradient and ridge forms. For the other three forms the saturation colourmap was judged to be significantly worse than any of the other encoding techniques.

Using the results from both experiments, Ware attempted to build a single colourmap that would provide good results for both metric and form visualization. Colours in the colourmap spiralled up through the colour space. This provided a monotonic increase in the luminance channel, and ensured no monotonic variation (and hence no large or constant simultaneous contrast) in either the red-green or blue-yellow channels. Ware tested his "optimal" colourmap by rerunning the metric experiment. This was done to ensure that variations in luminance did not interfere with a subject's ability to accurately report metric information. Ware's optimal colourmap performed as well as the spectral colourmap. This suggests that subjects were able to selectively ignore luminance information during the metric visualization task.

Levkowitz and Herman have also studied the problem of creating colourmaps for data visualization [Levkowitz and Herman, 1992]. Their specific problem environment is the visualization of medical images such as PET, CT, and MRI slices. They begin by noting that a grey-scale (*i.e.,* luminance-based) colourmap can provide somewhere between 60 and

90 just-noticeable difference (JND) steps. They attempted to build a linearized optimal colour scale (LOCS) in order to provide a much larger perceptual dynamic range during visualization.

Levkowitz and Herman assume the data attribute to be visualized has a range of possible values $v_1 \leq v_2 \leq \ldots \leq v_n$. The LOCS $c_1, c_2, \ldots, c_n$ used to represent such an attribute is designed to satisfy the following three properties:

- *order:* c_1 precedes c_2 precedes ... precedes c_n (as described below, "c_i precedes c_{i+1}" implies $r_i \leq r_{i+1}$, $g_i \leq g_{i+1}$, and $b_i \leq b_{i+1}$)
- *uniformity:* for any case where $v_{i+1} - v_i = v_{j+1} - v_j$, the perceived difference between (c_i, c_{i+1}) should be equal to the perceived difference between (c_j, c_{j+1})
- *boundaries:* the choice of colours should not introduce perceptual artifacts within the colour scale (*i.e.,* the colour scale should appear continuous)

Construction of the LOCS under the above conditions began by placing a number of restrictions on the colours that were used. It was assumed that c_1 would be black and c_n would be white. A natural scale was enforced by requiring the RGB values of (c_i, c_{i+1}) to satisfy $r_i \leq r_{i+1}$, $g_i \leq g_{i+1}$, and $b_i \leq b_{i+1}$. Colours in the LOCS were either all achromatic (*i.e.,* grey) or all chromatic (*i.e.,* colour). Finally, colours were chosen to guarantee a monotonic increase in saturation.

Given the requirements and restrictions placed on the LOCS, construction was reduced to an optimization problem, specifically the maximization of Euclidean distance between colour pairs specified in any colour model where Euclidean distance roughly equals perceived colour difference. Levkowitz and Herman chose to use the CIE LUV colour model.

Levkowitz and Herman showed that an LOCS with 32 values has a perceived colour-pair

difference six times larger than a linear grey-scale colourmap with 32 values. They then tested the LOCS against a grey-scale colourmap and a heated-object colourmap (both are commonly used to display medical image data). The LOCS was judged to be better than the heated-object colourmap, but was significantly worse than the grey-scale colourmap. Levkowitz and Herman explained this by suggesting that the task and effects of colour surround (*i.e.,* simultaneous contrast) might have favoured the grey-scale map.

Rather than choosing to use a particular colourmap, a user might want to be able to choose a colour path through a particular colour model. The path defines the range of colours to map onto the data attribute being visualized. Rheingans and Tebbs describe just such a tool [Rheingans and Tebbs, 1990]. The data being visualized occupies the center of the display window. A three-dimensional representation of a colour model (one of either RGB, HSV, or HLS) made up of discrete patches is overlaid on top of the visualization. Users can interactively sketch a path through the colour model. This defines the data to colour mapping. Users can also control the way changes in the data attribute map to the colour path. For example, a linear mapping would provide a constant interval width along the path for a constant increase in the data attribute's value. An exponential mapping would assign the majority of the attribute's values to colours at the front of the path. The largest values would be spread out along the middle and the back of the path. This might allow a user to see more subtle changes occurring in large attribute values.

Recent work at the IBM Thomas J. Watson Research Center has focused on a rule-based visualization tool [Rogowitz and Treinish, 1993; Bergman et al., 1995]. Initial research discussed the need for rules that take into account how a user perceives visual features such as hue, luminance, height, and so on. The representation of isomorphic data (Rogowitz and Treinish call continuous surfaces isomorphic data) is used to motivate this requirement. An n-fold increase in the data attribute's value should result in a corresponding n-fold increase in perceived difference in the visual feature being used to represent the data. An understanding

of how a user perceives difference is necessary to guarantee this result. For example, a simple doubling of intensity does not mean the user will perceive the larger data values as being "twice as bright". This is because perceived intensity increases in an exponential, as opposed to linear, manner.

Rogowitz and Treinish described three common visualization tasks: isomorphism, segmenting, and highlighting. Interestingly, the last two tasks appear to correspond directly to preattentive boundary and target detection. A framework system for rule-based visualization was then described. Various "rules" are used to guide or restrict a user's choice during the data-feature mapping. The "rules" take into account various metadata, for example, the visualization task being performed, the visual features being used, and the spatial frequency of the data attribute being visualized.

A subsequent paper describes the colourmap selection tool PRAVDAColor used by the IBM Visualization Data Explorer [Bergman et al., 1995]. The colourmap tool uses a variety of system-generated and user-provided metadata to limit the choice of colourmaps available to the user. Selection criteria include information on the attribute being visualized (is it ratio data or interval data?), the spatial frequency of data values (low frequency or high frequency?), and the visualization task being performed by the user (isomorphic, segmenting, or highlighting?). As an example of the "rules" used to guide colourmap selection, isomorphic tasks require a perceptually monotonic representation. In these cases, PRAVDAColor provides users with a choice between monotonically increasing luminance or hue colourmaps. Low frequency data is represented using hue. High frequency data is represented using either luminance or saturation. Once a colourmap is chosen, the user can control the data-colour mapping in a fashion similar to Rheingans interactive colourmap tool (*e.g.,* linear mapping, exponential mapping, and so on).

4.7 Colour in Preattentive Processing

Colour has been studied extensively in the preattentive processing literature. Initial work discusses the use of colour during visual search; results by various authors show that small colour differences between target and non-target elements produce serial search, while larger colour differences produce parallel search. Some work has been conducted to try to identify the minimum colour difference (or critical colour difference, as it is sometimes called) required for parallel search. Recent work has investigated other factors that make a target colour "stand out" from the background non-target colours. Results have shown that colour distance alone is not the only important factor. The named category occupied by the target colour and the ability to separate the target from its non-targets in colour space can also have a large effect on search efficiency.

One of the first papers discussing the use of colour during visual search was written by Carter [1982] . Three experiments were described in the paper. In the first experiment subjects were told to search for a specific three-digit number located in a circular array of random three-digit non-targets. Subjects were also told the colour of the target. During certain trials some of the non-targets used the same colour as the target. Results showed that the amount of time required to find the target increased linearly with the number of non-targets that shared the target's colour.

In the second experiment, the target's colour was unique and fixed to be light purple. Background non-targets were all coloured either dark purple or green. When the non-targets were coloured dark purple (*i.e.,* when the non-targets had a colour similar to the target) mean search time increased linearly with respect to the number of non-targets in the display (hereafter display size). When the non-targets were coloured green (*i.e.,* when the non-targets had a colour dissimilar from the target) mean search time was constant with respect to display size. Carter concluded that visual search can be either serial or parallel, depending

on the perceived difference between the target and the non-target colours.

In the third experiment, Carter tested the effects of placing potential target elements in a group. In a manner similar to the first experiment, both the target and the twenty-nine non-targets used a common colour. Twenty-five of the target-coloured elements were placed in a spatial group. The remaining five elements were randomly located in the display. In half the trials the target element was one of the elements in the spatial group. In the other half, it was one of the randomly located elements. It took significantly longer to find the target when it was part of the spatial group. Moreover, searching for an element outside the spatial group was faster than the results from Experiment 1 (*i.e.,* when all the potential target elements were randomly located in the display). Apparently, subjects scanned the outlying elements before examining elements within the group. Since there were only five outlying elements, search was faster than when all the elements were randomly placed.

An earlier paper by Carter and Carter [1981] measured search times for different combinations of a single colour target and a fixed number of differently coloured non-targets. As before, they found that search times were long for small colour differences and short for large colour differences. The colour difference required for parallel search was many times the JND threshold. Distance in CIE LUV and CIE Lab appeared to be related to search times by an exponential function. Nagy and Sanchez continued this work by defining what they called the "critical colour difference" for visual search [Nagy and Sanchez, 1990]. In their first experiment, Nagy and Sanchez replicated the results of Carter and Carter for both a red and a blue target in a set of off-white non-targets. A red or a blue target that was a large distance from the non-targets in colour space produced flat search times; a red or a blue target that was close to the non-targets produced search times that were linear in the number of non-targets being displayed.

Nagy and Sanchez continued their investigation by trying to find the minimum distance

required for parallel search. They chose five target colours along eight lines radiating out from the non-target in CIE XYZ colour space (Figure 4.7). The time required to find any one of the five targets in a display containing a fixed number of non-targets was measured and plotted for each of the eight lines. Two important results were found. First, the distance required for parallel search varied for each of the eight lines. In other words, the critical colour difference seems to depend on the colour of both the target and its non-targets. Second, the smallest critical colour difference (found during search for blue and green targets) was approximately 20 JNDs; this result was measured through the use of MacAdam JND ellipses [Wyszecki and Stiles, 1982].

John Duncan has also discussed various aspects of the use of colour during visual search [Duncan, 1989]. This work was part of a set of results used to support his similarity theory of preattentive processing [Duncan and Humphreys, 1989]. An initial experiment asked a subject to determine presence or absence of a patch with a specific target colour. Each display randomly contained one, two, three, or four uniquely coloured non-target patches. All of the colours used were chosen to lie along the boundary of the monitor's gamut in CIE XYZ (Figure 4.8a). Results showed that average response time was independent of the number of differently-coloured non-target patches. However, when non-target colours were chosen to be immediate neighbours of the target colour (*e.g.,* bg and/or p for a b target, or g and/or r for a y target), search was more difficult compared to displays where the non-target colours were more distant from the target colour. This matches the results of both Carter and Nagy and Sanchez, specifically, search time is affected by the colour distance between the target and its non-targets.

A second set of experiments by Duncan investigated specifically the use of colour and its relationship to his similarity theory of preattentive processing. Duncan chose four colours (named 1-2-3-4) along a path in CIE XYZ (Figure 4.8b). The target patch was coloured using either colours 1-4 (end targets) or colours 2-3 (middle targets). In the end target

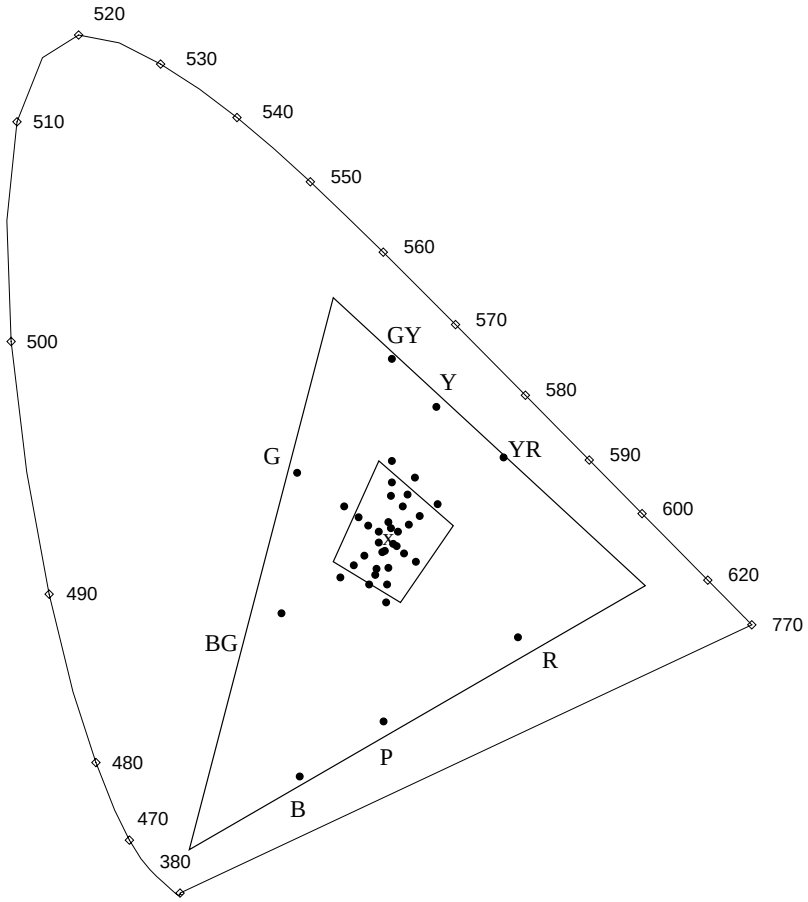


Figure 4.7: CIE diagram showing the monitor's gamut as a triangular region; Nagy and Sanchez chose five colours along eight colour lines radiating out from the distractor colour (the `x` near the center of the triangle); the parallelogram approximates the critical colour boundary where search times moved from serial to parallel for each of the eight colour lines

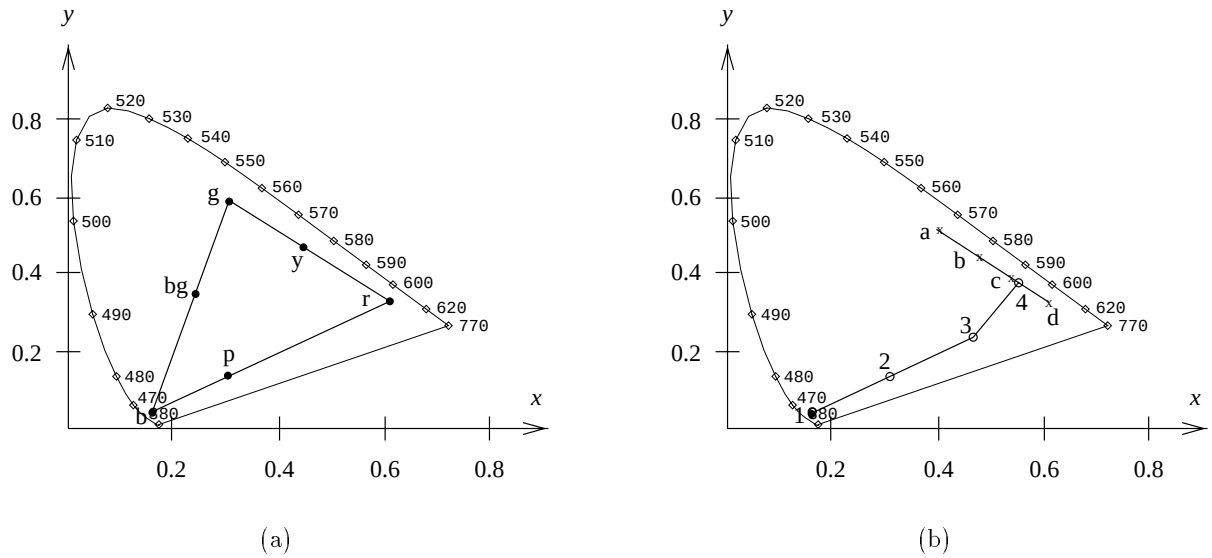


Figure 4.8: The sets of colours used during Duncan's colour boundary experiments: (a) the six colours used during Experiment 1, plotted along the boundary of the monitor gamut in CIE XYZ; (b) the colour sets 1-2-3-4 and a-b-c-d used during Experiment 2

case, non-target (N-N) similarity was high. In the middle target case, N-N similarity was low. Target–non-target (T-N) similarity was the same in both cases. Duncan replicated his experiment with a more narrowly spaced set of colours (named a-b-c-d) chosen from a different region of the monitor's gamut. This was designed to emphasise poor performance with end non-targets (*i.e.,* when a-d are non-targets and b-c are targets) due to the decrease in colour similarity between the non-targets. As predicted by Duncan's similarity theory, the middle target case gave much worse performance than the end target case. Moreover, the middle target case was worse for the a-b-c-d colour set, when compared to the 1-2-3-4 colour set. Duncan concluded this was due to the difference in N-N similarity: his theory predicts that a decrease in N-N similarity will result in an increase in search difficulty. Regardless of the explanation, his results show that the choice of non-target colours can have an affect on search performance.

Work by other authors has some relationship to the use of colour during visual search.

For example, Nagy and Sanchez discuss the differences between the use of chromaticity versus luminance during visual search [Nagy and Sanchez, 1992]. Results showed that a luminance difference of +33% (a ratio of $1 : \frac{4}{3}$) or −23% (a ratio of approximately $\frac{3}{4} : 1$) of the monitor's available gamut was required between the target and the non-targets in order to obtain parallel search. Similar critical differences between the target and the non-targets along the three chromaticity channels were +12% of the monitor's available gamut for red, +13% for blue, and +5% for yellow. This suggests that a monitor's chromaticity range can more easily provide distinguishable elements, when compared to the monitor's luminance range. Jeremy Wolfe describes an experiment where a target colour is found in parallel in a field of ten highly homogeneous background non-targets [Wolfe et al., 1990]. The colour distance between the target colour and the group of background non-targets was large. However, this suggests that it is possible to search a display containing many different colours for a highly-distinguishable target colour.

4.8 Linear Separation Effect

More recent work has examined the effects of selecting two or more heterogeneous colours for use in a single display. The "linear separation" effect was originally described by D'Zmura [1991] . He was investigating how the human visual system finds a target colour in a sea of background non-target colours. One hypothesis was that the visual system uses opponent-colour filters (*i.e.,* red, green, blue, and yellow filters) to search for the target. If this were true, a target colour could be rapidly and accurately detected only when it used an opponent-colour unique from its non-targets.

D'Zmura tested the opponent-colour hypothesis using four target detection experiments. In each experiment, observers were asked to determine the presence or absence of an orange

target. Half of the displays were randomly chosen to contain the target; the other half did not. A variable number of non-target elements (randomly one of 1, 8, 16, or 32 elements) were also present in each display. During one experiment half of the non-targets in each display were coloured green and half were coloured red. During the other three experiments they were coloured yellow and blue, green-yellow and purple, and yellow and red.

Results showed that the time required to determine the target's presence or absence was constant (at approximately 450 milliseconds) and independent of the total number of elements being displayed when the non-targets were coloured green and red, yellow and blue, or green-yellow and purple. This suggests detection occurs in parallel in the low-level visual system. When the non-targets were coloured yellow and red, however, the time required to identify the target was linearly proportional to the number of elements being displayed (it ranged from approximately 450 milliseconds for displays with one element to approximately 1500 milliseconds for displays with 32 elements). Subjects had to search serially through each display to determine whether the target was present or absent. An increase in the total number of elements being displayed resulted in a corresponding increase in the time required to complete the target detection task for this case.

D'Zmura's results were not consistent with the opponent-colour hypothesis. In particular, the time required to find an orange target in a sea of green-yellow and purple non-targets should not have been independent of the total number of elements being displayed, since no single opponent-colour filter can be used to completely separate the orange (*i.e.,* red-yellow) target from its green-yellow and purple (*i.e.,* red-blue) non-targets. D'Zmura suggested that the criteria for parallel target detection was the ability to separate a target linearly from its non-targets in the colour model being used. Figures 4.9a–4.9d show that the orange target can be linearly separated by a minimum threshold from the non-targets in the first three experiments, but not in the fourth. D'Zmura tested his new hypothesis by increasing the saturation of the orange target; this moved the target out from the non-targets, providing

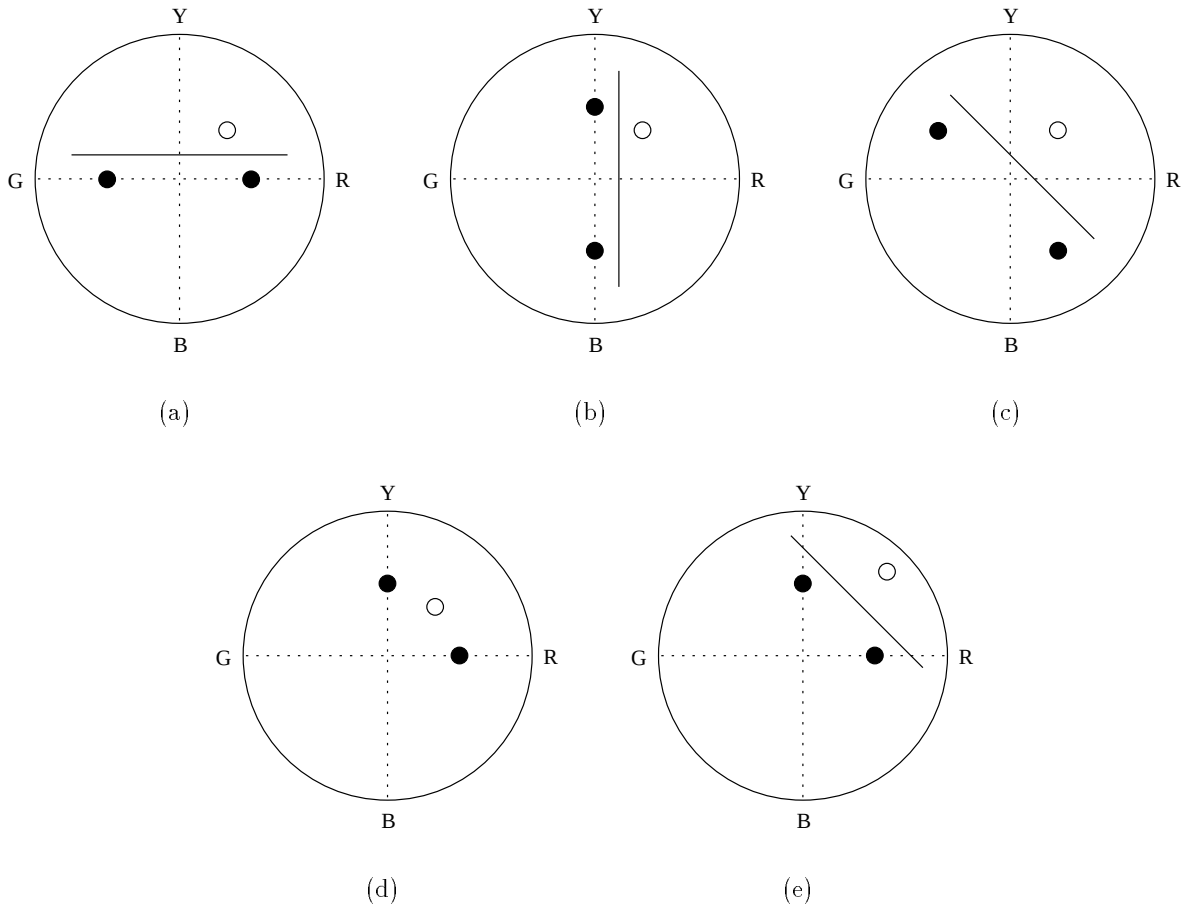


Figure 4.9: Opponent-colour discs (axes correspond to the four opponent-colours red, blue, green, and yellow) with the target (an open circle) and the non-targets (filled circles) for D'Zmura's five experiments; a straight line in (a)–(c) and (e) shows that the target can be linearly separated from the non-targets: (a) Experiment 1, with green and red non-targets; (b) Experiment 2, with yellow and blue non-targets; (c) Experiment 3, with green-yellow and purple non-targets; (d) Experiment 4, with yellow and red non-targets; (e) Experiment 5, a saturated orange target with yellow and red non-targets

the desired linear separation (Figure 4.9e). Results from using this new target were similar to those of the first three experiments; the more saturated orange target could be detected in time independent of the number of yellow and red non-targets being displayed.

D'Zmura extended his hypothesis by showing that linear separation continued to apply when either the saturation or the luminance of the target and non-targets were varied. He also presented a similar set of results for two additional target colours. This suggests that the linear separation effect is not restricted to a specific region in the colour model.

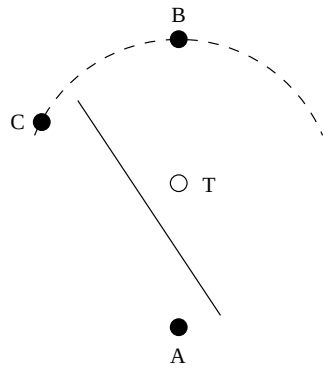


Figure 4.10: Example of the target and non-targets used in the experiments run by Bauer; the colours are shown in a u^*, v^*-slice from the CIE LUV colour model, notice that target T is equidistant from all three non-targets A, B, and C; in Experiment 1, target colour T was collinear with the non-target colours A and B; in Experiment 2, target T was linearly separable from its non-targets A and C

Work by Bauer et al. [1996] provides a number of additional results that strengthen D'Zmura's hypothesis. First, Bauer et al. showed that perceptual colour models cannot be used to overcome linear separation. Two target detection experiments similar to D'Zmura's were run using the CIE LUV colour model. In both experiments, the target–non-target distance was fixed to a constant value, in an attempt to control the perceived target–non-target colour difference (Figure 4.10). During the first experiment, the target T was collinear with the two non-targets (elements A and B in Figure 4.10). In the second experiment, the target T was linearly separated from the two non-targets (elements A and C in Figure 4.10). Because $\overline{TA} = \overline{TB} = \overline{TC}$ in CIE LUV, the perceived colour difference between the target

T and any of its three distractors was expected to be roughly equal. In spite this, the time required to identify the target as present or absent was significantly longer in the first experiment, when compared to the second.

Bauer described three other important results. First, he replicated his initial findings in three additional colour regions: green, blue, and green-yellow. This suggests linear separation applies to colours from any part of the visible colour domain. Second, he showed through a number of experiments testing subject performance–linear separation tradeoffs that the linear separation "line" is indeed a line, and not a curve. Third, he showed that the time required to identify a target decreased as the target–non-target colour distance increased. Rapid and accurate target detection was possible without linear separation when the target–non-target colour distance was relatively large.

4.9 Colour Category Effect

Results reported by Kawai et al. [1995] provide additional insight into the interactions that occur between groups of coloured data elements. Kawai et al. believe that the time required to identify a colour target depends in part on the named colour regions occupied by the target and its non-targets. Kawai et al. tested their hypothesis by running five target detection experiments for five different target colours, which they roughly identified as red, yellow, green, blue, and purple.

Colours were chosen using the Optical Society of America (OSA) uniform colour system; the OSA model is perceptually balanced, and specifies colours using the three dimensions (L, j, g). L encodes luminance or intensity, j represents the blue-yellow opponent-colour dimension (negative j values give blue colours, while positive j values give yellow colours), and g represents the red-green opponent-colour dimension (negative g values give red colours,

while positive g values give green colours). Wyszecki and Stiles [1982] give a complete description of the OSA model. Experiments for a given colour target were split into subblocks; each subblock displayed a single type of non-target element. Target–non-target distance was varied by moving the non-target colour farther and farther from the target in both the g and j directions. In these experiments, there were only two different colours in each display: the uniquely coloured target, and a constant number of uniformly coloured non-targets.

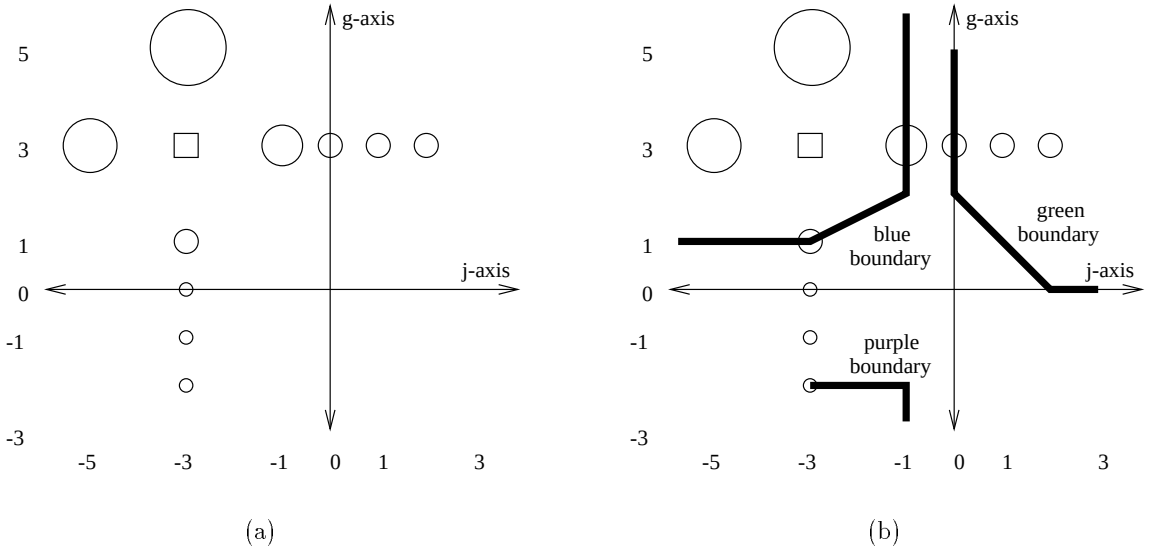


Figure 4.11: Graph of mean detection times for the blue target and ten different non-targets, colours are shown in a j, g-slice from the OSA colour model: (a) the square represent the target's colour; each circle represents the colour of one of the ten non-targets; the size of each circle represents the mean detection time required to identify the target when it was displayed with the given non-target; (b) the same figure as in (a), but with the boundaries of the blue, green, and purple colour regions shown as thick lines

If target identification was dependent only on the colour distance between the target and the non-targets, the time required to detect presence or absence of the target should have decreased uniformly as target–non-target distance increased. This was not the case, however. Figure 4.11 shows results for the blue target. Non-targets that were $-2g$ units from the target resulted in very low mean detection times. However, moving the non-target $+2g$ units from the target (*i.e.,* using exactly the same target–non-target distance as for $-2g$

non-targets) resulted in very large mean detection times. A non-target that was $+2j$ from the target also showed a large mean detection time; however, moving it one more unit to $+3j$ resulted in a dramatic decrease in the time required to identify the target.

Apparently, target detection times are not based solely on the target–non-target colour distance. Kawai et al. suggest that the named colour regions of the target and non-target affect search time. If the target and non-target occupy the same named region, search time will be relatively large. If the target and non-target are in different named regions, search time will be smaller. This hypothesis was tested by experimentally dividing the j, g-slice into named colour regions. Figure 4.11b shows the boundaries of the blue, green, and purple colour regions. Notice that search time decreases dramatically whenever the non-target is moved outside the blue colour region.

Kawai et al. found that similar search time asymmetries for green, purple, and red targets could also be explained by a difference in colour regions. They concluded that search time depends not only on colour distance, but also on the named colour regions occupied by a target and its non-targets.

4.10 Orientation and Texture

Although colour is a popular and important visual feature, it is certainly not the only method used to represent multidimensional information. Other features like orientation and size can also be used to visualize data elements. In fact, it has been suggested that orientation, size, and contrast can be used together to specify visual textures, in a manner similar to the way hue, value and saturation are used to specify different colours.

The systematic use of orientation and texture in visualization has not received much focus to date. Notable exceptions are the EXVIS system [Pickett and Grinstein, 1988; Grinstein

et al., 1989], Liu and Picard's use of Wold features [Liu and Picard, 1994], Li and Robertson's use of Markov random fields [Li and Robertson, 1995], and Ware and Knight's discussion of the fundamental dimensions of a texture element [Ware and Knight, 1992; Ware and Knight, 1995]. EXVIS is used to show spatial coherence in a multidimensional dataset. An n-dimensional data element is represented using a "stick-man" with $n-1$ arms. The n values from each data element are used to control the orientation of the stick-man's arms and body. Spatially neighbouring elements with a similar set of n values will result in a group of similar looking stick-men. This appears as a spatial region with a unique texture in the display (Figure 2.13). Pickett and Grinstein make reference to preattentive processing when they describe the ability of viewers to rapidly and accurately identify both the texture regions and the boundaries they form.

Colin Ware and William Knight have conducted a number of experiments that investigate the use of Gabor filters for building visual textures. Ware and Knight provide evidence to suggest that orientation, size, and contrast can be used as fundamental dimensions during visual texture construction. They describe the range of each dimension, and discuss how to measure the perceived difference between values from each dimension. For example, orientation values are normally scaled linearly over the range $0 \ldots \pi$. Size and contrast values, on the other hand, should be scaled exponentially. The result is a perceptually balanced texture model, analogous to perceptually balanced colour models like CIE LUV, CIE Lab, or Munsell. Ware and Knight then show how Gabor filters can be used to build visual textures that have a specific orientation, size, and contrast. In a similar manner, Liu and Picard believe the three perceptual dimensions of a texture pattern are periodicity (or repetitiveness), directionality, and granularity (or randomness). Liu and Picard use Wold features to control their texture dimensions when they build visual textures. Finally, Li and Robertson show how properties of Markov random fields can be controlled by individual data attributes to produce visual patterns that represent an underlying dataset.

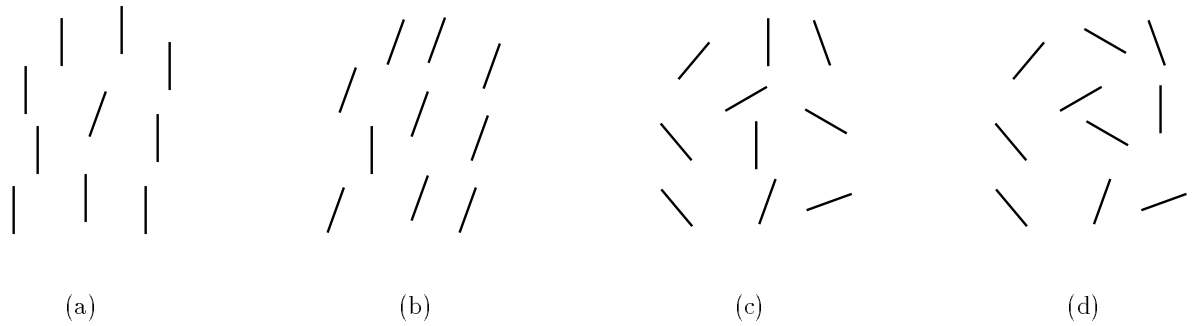


Figure 4.12: Examples of the tilted and steep orientation categories: (a) a tilted target (20° rotation) is easy to find in a sea of steep distractors; (b) a steep target (0° rotation) is easy to find in a sea of tilted distractors; (c) a tilted target (20° rotation) is hard to find in a sea of steep, tilted, and shallow distractors; (d) a steep target (0° rotation) is hard to find in a sea of steep, tilted, and shallow distractors

Jeremy Wolfe has recently studied how orientation is categorized during preattentive visual search [Wolfe et al., 1992]. His results suggest that the low-level visual system detects three different categories of orientation: steep, shallow, and tilted. A target element must fall within a category unique from its distractors to guarantee rapid and effortless visual search.

Wolfe's initial experiments tested detection of a vertical (0° rotation) target in a set of oblique (20° rotation) distractors (Figure 4.12b), and detection of an oblique target in a set of vertical distractors (Figure 4.12a). Subjects performed both tasks in times independent of the number of distractors. During a second experiment, however, search for a vertical element in a set of distractors rotated 20°, 40°, 60°, 80°, −20°, −40°, −60°, and −80° (Figure 4.12d) was significantly more difficult. An increase in the number of non-targets brought about a corresponding increase in subject response time. This occurred in spite of the fact that the orientation difference between the target and its nearest background non-target was 20° rotation, exactly as in the first experiment. Similar poor results were found when searching for an oblique target in a display of non-targets rotated 0°, 40°, 60°, 80°, −20°, −40°, −60°, and −80° (Figure 4.12c).

Wolfe suggested that the orientation categories occupied by the target and non-targets are what control search efficiency. He tested his theory by performing a number of visual search experiments. In all four cases, the minimum orientation difference between the target and non-targets was at least 20°, exactly as in the first experiment that allowed easy search.

- Search for a 0° target (steep orientation) among 40° (tilted) and −40° (tilted) distractors was easy.
- Search for a 0° target (steep) among 20° (steep and tilted) and −20° (steep and tilted) distractors was hard.
- Search for a 20° target (steep and tilted) among 0° (steep) and 40° (tilted) distractors was hard.
- Search for a 20° target (steep and tilted) among 60° (tilted) and −20° (steep and tilted) distractors was hard.

Wolfe conducted similar experiments that showed that shallow and tilted also acted as orientation categories during visual search. Wolfe's results suggest that only three distinguishable orientations are available for rapid and accurate visualization. This limited range of values will be particularly important when a user decides which data attribute to represent using orientation.

One other interesting orientation property was described by Wolfe et al. [1990] during their study of conjunction search. Wolfe's guided search theory suggests that both bottom-up and top-down information are used during preattentive visual search. This means that, in certain cases, top-down information from a subject can be used to search preattentively (*i.e.,* rapidly and independent of the number of non-targets) for a conjunction target. Wolfe shows results from an experiment where subjects searched for a red-oblique target (colour ×

orientation) in a sea of blue-oblique and red-vertical distractors. An increase in the number of distractors resulted in almost no increase in the amount of time required to detect presence or absence of the target. This suggests that subjects are capable of finding a between-feature conjunction target, in this case, a colour $\times$ orientation target. However, Wolfe showed that subjects are not capable of finding a within-feature conjunction target. Wolfe tested both colour $\times$ colour targets (a red-blue square in a sea of red-green and green-blue squares) and orientation $\times$ orientation targets (a combination of a vertical and an oblique rectangle in a sea of vertical-horizontal and horizontal-oblique distractors). In both cases subjects needed to use a serial search to determine presence or absence of the target. This suggests that although orientation can be categorized into three separate values, conjunctions of these values cannot be used to form a salient target.

Nothdurft has provided much insight into the phenomena of texture segregation (*i.e.,* boundary detection and region identification). Some of his initial work examined the effects of element spacing on shape identification [Nothdurft, 1985a; Nothdurft, 1985b]. Shape identification increased in efficiency with an increase in orientation difference Δ_{tg} between target and distractor elements. Shape identification decreased in efficiency when the spacing between individual elements Δ_x increased. Nothdurft concluded that identification performance was based on the spatial gradient of foreground to background orientation contrast Δ_{tg}/Δ_x across the texture boundary.

Later work by Nothdurft investigated the effect of a background shift in element orientation Δ_{bg} during texture segregation [Nothdurft, 1991]. Two preattentive tasks were tested. In the first, subjects were asked to identify a "bar" of differently-oriented elements as either vertical or horizontal. In the second, subjects were asked to provide the rough location of a single element with an orientation different from its neighbouring background distractors.

The orientation of background distractors in some trials varied from neighbour to neigh-

bour by a constant amount Δ_{bg}. For example, trials with a constant distractor orientation had $\Delta_{bg} = 0°$. Other trials varied neighbour-to-neighbour distractor orientation anywhere from 5° to 30°. The amount of orientation difference Δ_{tg} required between background and target elements for a 75% correct response rate was plotted again Δ_{bg}.

An increase in Δ_{bg} required a corresponding increase in Δ_{tg} in order to obtain a 75% correct response threshold. For example, during bar identification, a Δ_{bg} of 0° (*i.e.,* constant background element orientation) required a Δ_{tg} of 20°. Increasing Δ_{bg} to 20° brought about a corresponding increase in Δ_{tg} to 50° for a 75% correct response rate. A Δ_{bg} of greater than 30° in either task (bar or target identification) rendered it impossible to complete with a 75% accuracy rate. Nothdurft concluded that the required orientation contrast Δ_{tg} depends on the average orientation contrast (*e.g.,* "noise") between neighbouring background elements.

Nothdurft compiled a final study of orientation, motion, and colour cues during region segmentation [Nothdurft, 1993]. He confirmed that for a continuously varying background feature, targets popped-out only when the target difference was much larger than the continuous background difference. This was shown to be true for all three features. Nothdurft also showed that the large difference requirement depends on local neighbours for region segmentation based on orientation and motion. Specifically, if the non-target neighbours around the target region provided a large difference in orientation or motion with the target, the region can be easily identified. Target regions that were embedded in a group of non-targets that had a small orientation or motion difference with the target were difficult to find. For colour, this property was not present. Colour pop-out depended on the amount of colour difference between the target and the most similar non-target. Changing the type of surrounding non-targets to provide a relatively larger or smaller local feature difference had no effect on this property.

We can draw the following conclusions from the works of Wolfe and Nothdurft:

- During target detection, the visual system divides orientation into three rough categories: steep, shallow, and tilted. A target element can be identified rapidly only when it occupies a category unique from its non-targets.

- For rapid boundary and region identification, an orientation difference of 20° is required to identify the region in a field of constant-orientation non-targets.

- For rapid boundary and region identification, any increase in orientation difference between neighbouring non-targets requires a corresponding increase in orientation difference between targets and non-targets.

One question that has not been answered is how the results of Wolfe and Nothdurft are related. It appears that Nothdurft used only vertical and horizontal non-targets while searching for his oblique target region. This explains why he needed only a 20° orientation difference for parallel search; if he had used an oblique target (say 40° rotation) in a sea of oblique non-targets (say 20° rotation), Wolfe's results suggest his subjects would have moved towards serial search. This argument might also explain, in part, results from displays that varied the non-target orientation. Changing the orientation of certain background elements may have pushed them into the same orientation category as the target region. Since target elements now share an orientation category with some of their non-targets, detection (according to Wolfe) should be more difficult. Further experiments are needed to build a general model that explains both Nothdurft's and Wolfe's results.

Chapter 5

Effective Colour Selection

Put simply, visualization is the mapping of data attributes onto visual features such as shape, size, spatial location, and orientation. A "good" visualization technique will provide an effective mapping that allows a user to rapidly and accurately explore their dataset. Obviously, the effectiveness of the mapping depends on a number of factors, including the context of the original dataset (*e.g.,* geographic data, medical images, time-varying flow field slices), the number of data elements, the dimensionality of each data element, and the task the user wishes to perform.

Colour is an important and frequently-used visual feature. Examples include colour temperature gradients on maps and charts, colour-coded vector fields in flow visualization, or colour icons displayed by real-time simulation systems. If we use colour to represent our data, one important question to ask is: How can we choose effective colours that provide good differentiation between data elements during the visualization task? We address this problem by trying to answer three related questions:

- How can we allow rapid and accurate identification of individual data elements through the use of colour?
- What factors determine whether a "target" element's colour will make it easy to find,

relative to differently coloured "non-target" elements?

- How many colours can we display at once, while still allowing for rapid and accurate target identification?

Target identification is a necessary first step towards performing other exploratory data analysis tasks. If we can rapidly and accurately differentiate elements based on their colour, we can apply our results to other important visualization techniques. These include rapid and accurate detection of data boundaries, the tracking of data regions in real-time, and enumeration tasks such as counting and estimation [Varey et al., 1990; Triesman, 1991; Healey et al., 1993].

We are most interested in guidelines that help us choose effective hues for use during visualization. An intuitive first step to gaining more control over colour would be to use a perceptual colour model such as CIE LUV, CIE Lab, Munsell, or the OSA Uniform Colour System [Birren, 1969; Wyszecki and Stiles, 1982]. These models use Euclidean distance to approximate the perceived colour difference between pairs of colours. Unfortunately, fixing the colour distance to a constant value does not guarantee that each colour will be equally easy to detect. Other factors such as linear separation [D'Zmura, 1991; Bauer et al., 1996] and colour category [Kawai et al., 1995] can affect how groups of coloured elements interact with one another.

5.1 Colour Selection Technique

During the design of our colour selection technique, we assumed that the user might choose to search for any one of the available data elements at any given time. This is typical during exploratory data analysis; users will often change the focus of their investigation based on

the data they see as the visualization unfolds. This requirement meant that we could not predefine which elements were targets and which were non-targets. The colour selection technique had to allow for rapid and accurate identification of any of the elements being displayed.

Results discussed in the Colour chapter suggest that choosing effective colours for data visualization depends on at least three separate criteria:

- *colour distance:* the amount of colour distance between different elements as measured in a perceptually balanced colour model
- *linear separation:* the ability to linearly separate targets from non-targets in the colour model being used
- *colour category:* the named colour regions occupied by both the target and non-target elements

We needed a simple method for measuring and controlling all three of the above effects during colour selection. We measured colour distance by choosing our colours using the CIE LUV colour model. We had to ensure that the colours we chose had the same perceived intensity; previous research by Callaghan has shown that random variation in intensity can interfere with an observer's ability to perform visualization tasks based on colour [Callaghan, 1984]. Because of this, colours were chosen from an isoluminant u^*, v^*-slice through CIE LUV. We guaranteed linear separability by picking colours that lay on the circumference of a circle embedded in our isoluminant u^*, v^*-slice.

We wanted to maximize the number of available colours, while still maintaining control over colour distance and linear separability. To do this, we computed the monitor's gamut within the u^*, v^*-slice being used. We then found the largest circle inscribed within the

gamut. We chose colours that were equally spaced around the circle's circumference (Figure 5.1). This method ensured that neighbouring colours had a constant colour distance. It also ensured that any colour acting as a target had a constant linear separation from every other (non-target) colour.

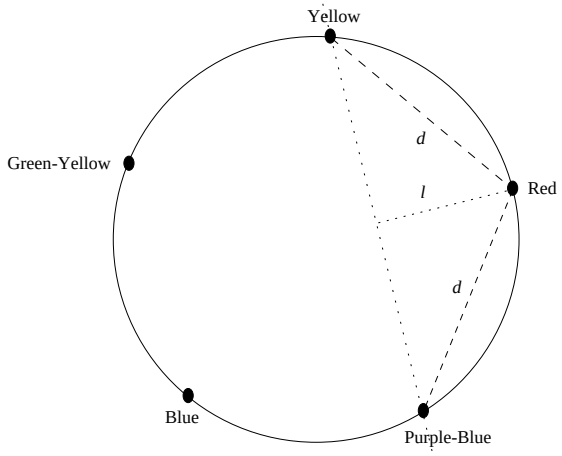


Figure 5.1: An example of five colours chosen around an inscribed circle's circumference; each element has a constant colour distance d with its two neighbours; moreover, when any element (for example, the Red element) acts as a target, it has a constant linear separation l from the remaining (non-target) elements

In order to control colour category effects, we designed a simple method for segmenting a colour region into individual colour categories. Our algorithm consists of two parts: an automatic step that divides the colour region into an initial starting state, and an experimental step that provides a user-chosen name, strength, and a measure of perceptual overlap for each category.

5.2 Estimating Monitor Gamut

Estimating a monitor's gamut in a two-dimensional slice through a colour space (in our case, in a u^*, v^*-slice through CIE LUV) begins with the chromaticity values of the monitor's phosphors, and the luminance of its maximum-brightness red, green, and blue. These

values are needed to convert colours from CIE LUV into monitor RGB (for a more complete description, see the sections on RGB, CIE XYZ, and CIE LUV in the Colour chapter).

Next, we built a set of "high" and "low" boundaries for each of the three monitor primaries red, green, and blue. One side of the boundary represents colours that fall outside the monitor's gamut; the other side represents colours that may or may not be within the gamut. For example, the Blue-Lo boundary divides the colour space into two regions: colours with a monitor blue of less than zero, and colours with a monitor blue of greater than zero. Any colour with a blue value less than zero cannot be displayed on the given monitor. Colours with a blue value greater than zero may or may not be displayable. Similarly, Blue-Hi divides the colour space into colours with a blue value of greater than one (which are undisplayable), and colours with a blue of less than one (which may or may not be displayable).

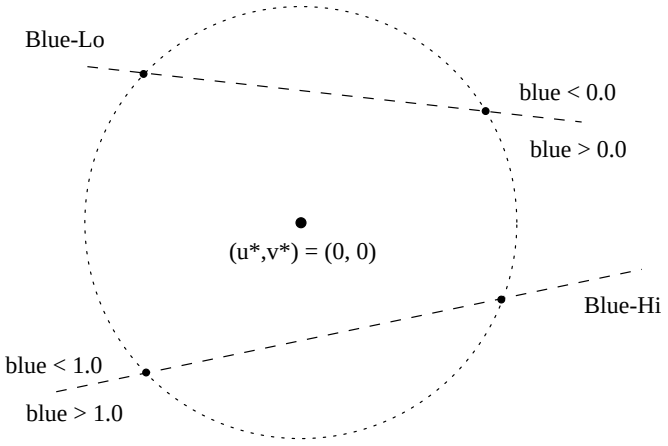


Figure 5.2: The boundary lines Blue-Lo and Blue-Hi are computed using a circle centered at $(u^*, v^*) = (0, 0)$ in CIE LUV; the two points where the circle leaves and then reenters the monitor's gamut are identified; these are used to build a corresponding boundary line

All of our gamut boundaries were built in the CIE LUV colour space. In CIE LUV uniform chromaticity is a projective transformation of chromaticity, which is itself a projective transformation of the monitor's primaries [Wyszecki and Stiles, 1982; Robertson, 1988]. This means our boundary lines are, in fact, straight lines in our u^*, v^*-slice through CIE LUV.

The equation of each line was computed by drawing a large circle centered at $(u^*, v^*) = (0, 0)$ that intersected the given boundary at two points (Figure 5.2). These points were used to construct our boundary line.

The monitor's gamut is represented by the largest convex polytope formed by the six boundary lines. We assume that the CIE LUV white-point $(u^*, v^*) = (0, 0)$ is within the monitor's gamut at every L^* luminance value. This means we can compute the largest convex polytope as follows:

1. Convert each boundary line into a dual point; given a boundary line in the form $ax + by = 1$, the corresponding dual point is $< a, b >$.
2. Compute the convex hull of the dual points.
3. Dual points that are on the convex hull represent lines that bound the convex polytope.

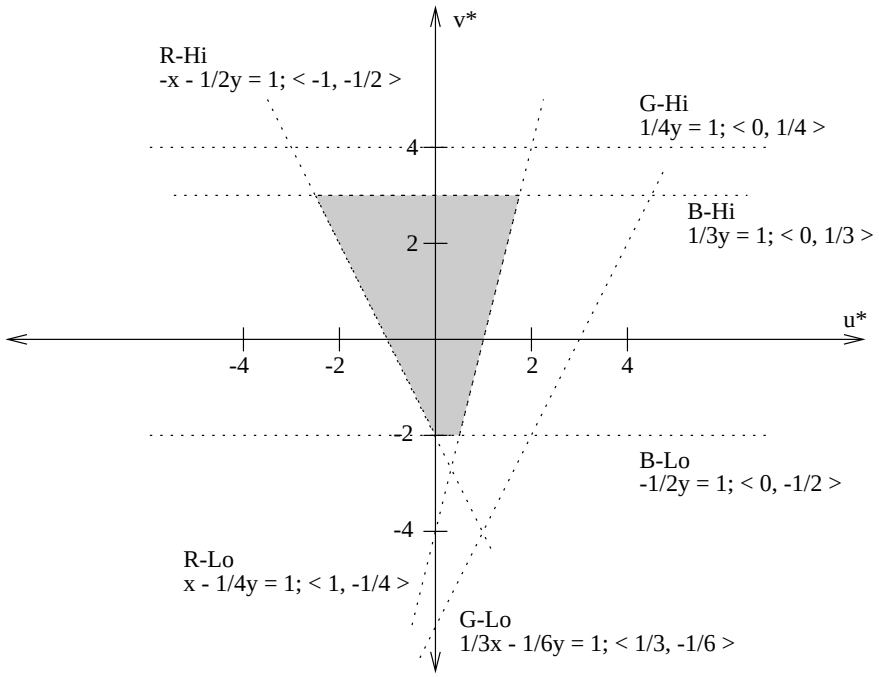
We wanted to construct the largest circle inscribed within the resulting convex polytope. This problem is divided into four cases, based on the number of sides in the given polytope.

Case 1: Three-sided triangle

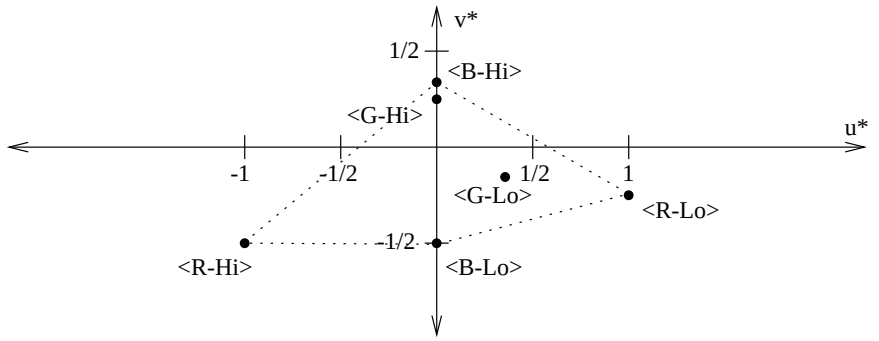
The largest circle inscribed within a triangle will touch all three sides of the triangle (Figure 5.4a). The center of the triangle C is defined by the intersection of any two angle bisectors. Given the center C and vertex P_1's bisector angle α (Figure 5.4b), the radius r of the inscribed circle is $r = \overline{P_1 C} \sin \alpha$.

Case 2: Four-sided quadrilateral

The largest circle inscribed within a quadrilateral will touch three or four sides of the quadrilateral. We begin by computing the bisectors of opposite sides of the quadrilateral. We identify the two sides that form the longest bisector, then select the longer of the sides X (Figure 5.4c). The inscribed circle will touch X and the two sides W and Y adjacent to it.



(a)



(b)

Figure 5.3: Finding the largest convex polytope involves computing the convex hull of the duals of each boundary line: (a) the convex polytope is formed by the intersection of boundary lines B-Lo, R-Lo, B-Hi, and R-Hi; (b) points <B-Lo>, <R-Lo>, <B-Hi>, and <R-Hi> are vertices of the dual point convex hull, and therefore identify R-Lo, R-Hi, B-Lo, and B-Hi as the boundaries that form the corresponding convex polytope

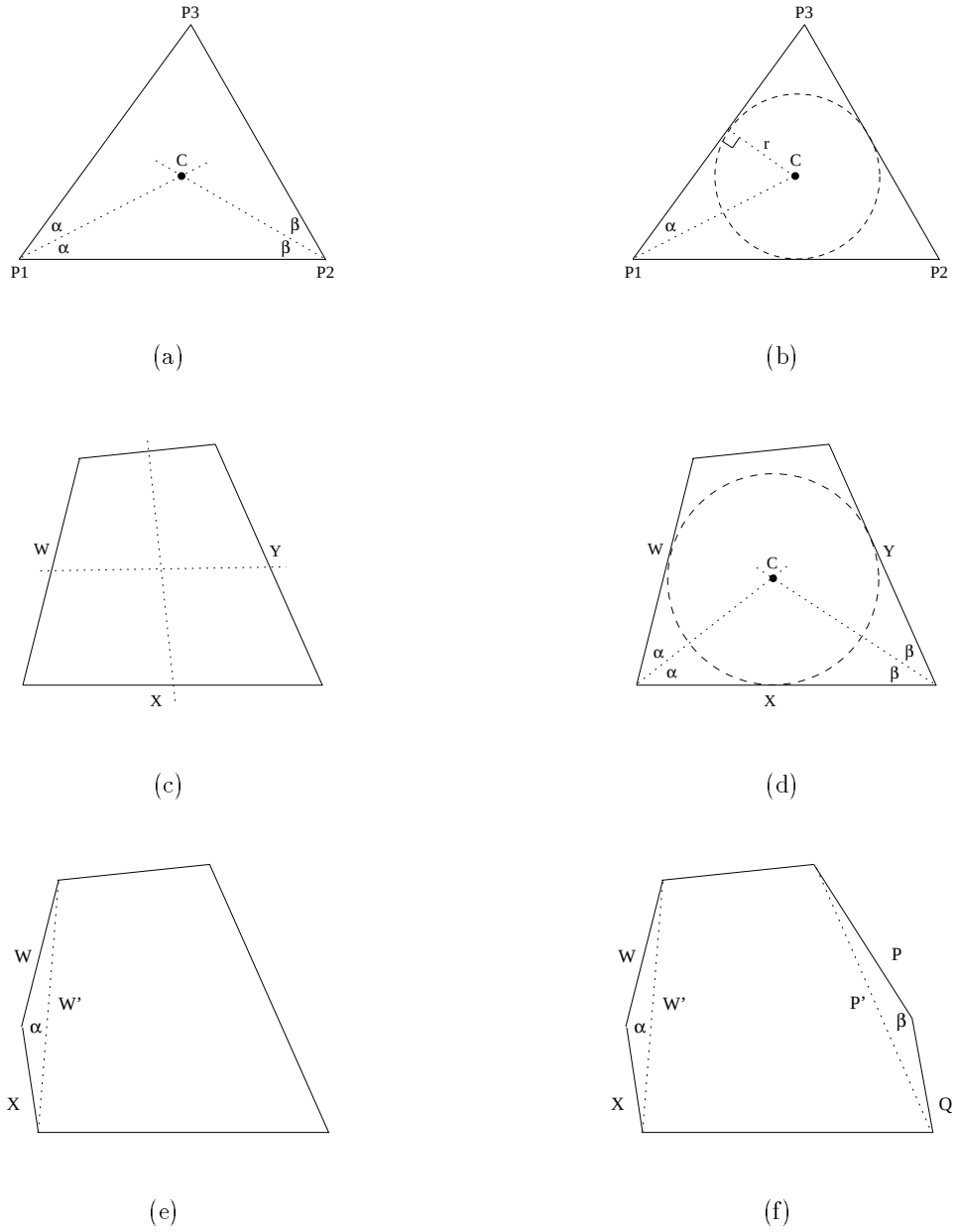


Figure 5.4: An inscribed circle placed within an arbitrary polytope: (a) within a triangle, the circle C is defined by the intersection of any two angle bisectors; (c) the radius of the circle r is calculated as $r = \overline{P_1 C} \sin \alpha$; (c) within a quadrilateral, the longest edge X of the longest opposite sides bisector is identified; (c) the center of the circle C is defined by the intersection of the two bisectors $\angle WX$ and $\angle XY$; (e) within a pentagon, the sides W, X that form the largest interior angle $\angle WX$ are combined as W' to form a quadrilateral; (b) the sides of the hexagon W, X and P, Q that form the two largest interior angles $\angle WX$ and $\angle PQ$ are combined as W' and P' to form a quadrilateral

The center of the circle C is defined by the intersection of the two angle bisectors of $\angle WX$ and $\angle XY$ (Figure 5.4d). The radius r of the circle is computed in a manner similar to the three-sided triangle case.

Case 3: Five-sided pentagon or six-sided hexagon

Five and six-sided convex polytopes are initially reduced to a four-sided quadrilateral. We can then use the solution to Case 2 above to find a reasonable approximation of the largest inscribed circle. For example, in a five-sided pentagon we find the two edges W and X that form the largest interior angle $\alpha = \angle WX$. These edges are replaced with a new edge W' (Figure 5.4e). The largest circle inscribed within the resulting quadrilateral is used to approximate a solution to the original five-sided pentagon. A similar technique is used for six-sided hexagons; we replace the edges W, X and P, Q that form the two largest interior angles $\alpha = \angle WX$ and $\beta = \angle PQ$ with two new edges W' and P' (Figure 5.4e). As in the five-sided case, we estimate the maximum circle by inscribing within the resulting quadrilateral.

5.3 Segmenting Colour Regions

Although colour models allow us to describe colours in an unambiguous manner, in practice colours are often identified by name. This technique is not precise, because it relies on a common colour vocabulary and an agreement on what to call different colours. In spite of this, it is useful to be able to name individual colours. Identifying colours by name is a method of communication that everyone understands. Names can also be used to divide a colour gamut into categories of similar colours. That is, given a set of n names $N_1, \ldots, N_n$, we can divide a colour region into n categories $C_1, \ldots, C_n$ such that all the colours in category C_i are best described by the name N_i.

The National Bureau of Standards developed the ISCC-NBS colour naming system [NBS,

1976] to try to provide a standard method for choosing colour names. English terms along the three dimensions hue, lightness, and saturation are used to represent different colours. Words from each dimension are combined in various ways to produce 267 different colour names (*e.g.,* dark red, strong blue, greenish blue, and so on).

One problem with the ISCC-NBS model is the lack of a systematic syntax; this was addressed during the design of a new Colour-Naming System (CNS) [Berk et al., 1982; Kaufman, 1986]. The CNS was based in part on the ISCC-NBS model. It uses the same three dimensions of hue, lightness, and saturation. However, the rules used to combine words from these dimensions are defined in a formal BNF syntax. Berk compared the RGB, HSV, and CNS colour models by asking observers to use each system to name a set of colours. Observers were most accurate when they used the CNS model. Both Berk and Kaufman describe a method of representing CNS names (as discrete points) within the Munsell colour model.

An extension of CNS called the Colour-Naming Method (CNM) was proposed by Tominaga [Tominaga, 1985]. The CNM uses a systematic syntax similar to the one described in the CNS model. In addition, colour names from the CNM can be mapped to colour ranges in the Munsell colour model. Names in the CNM are specified at one of four accuracy levels called fundamental, gross, medium, or minute classification. Names from a higher accuracy level correspond to smaller colour regions in Munsell. A brief description of the algorithm used to convert from a name to a Munsell colour range is provided in the paper.

In psychology, colour naming experiments are used to divide a colour gamut into named regions. First, the colour names to be used during the experiment are chosen. The colour gamut to be named is divided into representative colours. Observers view these colours one after another and choose from the group of colour names the most appropriate name for each. Experiments of this type are complicated for a number of reasons. The colour gamut has to

be divided into a reasonable number of different colours, to ensure that accurate category boundaries are reported. Even a 2D slice through a colour model may need to be divided into hundreds or sometimes thousands of representative colours. A large number of observers are also required, to ensure that the choice of names is not biased in some unusual manner. Different observers will disagree on the name for an individual colour, so some method must be devised to choose a single name in these cases. Finally, some skill is required in picking the initial group of colour names. A set of names that does not cover completely the colour gamut will force users to name certain colours in an ad-hoc fashion (*e.g.,* asking observers to name blues from the RGB colour cube using only the names red, green, and yellow).

The CNS, CNM, and ISCC-NBS models are all designed to provide a set of names that can be used to accurately identify colours from the visible colour domain. The CNS and CNM position their names within the Munsell colour model. In particular, the CNM can be described as a "renaming" of Munsell patches, since each name from the CNM is mapped to a colour range in Munsell. Unfortunately, it is not obvious how to use these results to automatically segment colour regions into a small number of colour categories. Perceptually balanced colour models like Munsell and CIE LUV were created using controlled psychological experiments. Rather than building a new colour naming language or performing our own colour naming tests, we decided to use the built-in properties of Munsell and CIE LUV to provide names for individual colours. We wanted an algorithm that was:

- *automatic:* the algorithm should automatically segment a colour region into a set of colour categories; individual colours would then be assigned the name of the category in which they lie;
- *accurate:* the names for each category must be "descriptive", that is, given the set of colour names being used, observers should agree the name chosen by the algorithm represents correctly the colours being named;

- *stable:* the algorithm should build stable, well defined boundaries between neighbouring categories;
- *perceptually controlled:* the algorithm should consider perceived colour differences during naming

Method

Our technique assumes the target colour to be named is specified using CIE XYZ. Algorithms exist to convert colours from most colour models into an (x, y, Y) value. The target colour is then mapped from CIE XYZ into the Munsell colour space. The Munsell hue dimension uses colour names to identify different hues; this dimension is used to name the target colour.

There are no simple functions to convert a colour from CIE XYZ to Munsell. Some mathematical algorithms have been proposed [Miyahara and Yoshida, 1988]. The CIE suggests using CIE Lab values to obtain Munsell hue, value, and chroma [CIE, 1976] as:

$$\begin{aligned} H &= tan^{-1}(b^*/a^*) \\ V &= L^*/10 \\ C &= (a^{*2} + b^{*2})^{\frac{1}{2}} \end{aligned} \tag{5.1}$$

Unfortunately, these methods are complicated and sometimes inaccurate for certain regions of CIE XYZ or Munsell. We decided to use a much simpler method of mapping, namely table lookup. The Munsell values listed in Wyszecki and Stiles [1982] are specified as (x, y, Y) values. The Munsell patch closest to the target colour is used to represent the target. Given the target colour (x_t, y_t, Y_t) and a Munsell patch (x_M, y_M, Y_M), the distance between the two is simply:

$$d = \sqrt{(x_t - x_M)^2 + (y_t - y_M)^2 + (Y_t - Y_M)^2} \tag{5.2}$$

One problem with this technique is the lack of perceptual balance in CIE XYZ. The Munsell patch that gives the smallest d is closest in Euclidean distance, but it may not be closest in perceptual difference. To overcome this, we computed the distance between target and patch in CIE LUV. The Munsell patch colours were converted and stored as (L^*, u^*, v^*) values. Target colours were mapped from CIE XYZ to CIE LUV in a similar manner. Given the target colour (L_t^*, u_t^*, v_t^*) and a Munsell patch (L_M^*, u_M^*, v_M^*), the perceived difference between the two is now:

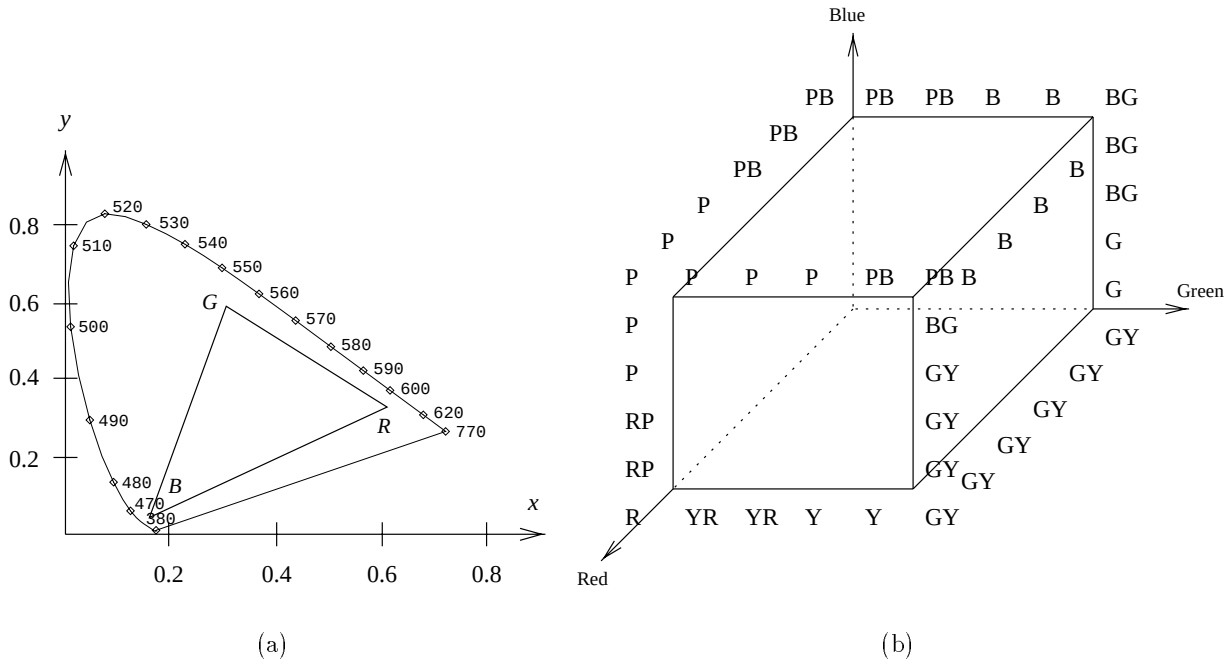
$$d = \sqrt{(L_t^* - L_M^*)^2 + (u_t^* - u_M^*)^2 + (v_t^* - v_M^*)^2} \tag{5.3}$$

This allows us to match the target colour to the Munsell patch that is closest in perceptual difference. The hue name of the given patch is then assigned to the target colour.

Results

When we designed our algorithm, one property we wanted to try to provide was stability, namely that category boundaries were well defined, and that names attached to a smoothly changing set of colours also changed smoothly. Individual Munsell patches are represented as points when they are moved into the CIE LUV coordinate system. Because this is a continuous mapping, the topology of the Munsell colour model is preserved; patches that were adjacent in Munsell are also adjacent (as points) in CIE LUV.

Target colours are associated with Munsell patches using a Euclidean distance metric. Consider the region around a Munsell patch M_i such that all points in the region are closer to M_i than to any other patch M_j. Any target colour that falls within this region is associated with M_i. A 3D Voronoi diagram in CIE LUV of the points representing Munsell patches builds exactly these regions for every M_i [Okabe et al., 1992]. Polyhedrons in a 3D Voronoi diagram are guaranteed to be convex. This means a straight line in CIE LUV will never pass



(a) (b)

Figure 5.5: Silicon Graphics monitor gamut and corresponding RGB colour cube: (a) the the monitor's gamut is defined by the chromaticities of its triads to be a triangle in the CIE chromaticity horseshoe; (b) names for colours along nine of the 12 RGB colour cube edges

through a patch's Voronoi region more than once. Because the Voronoi regions of connected Munsell patches are themselves connected in CIE LUV, the names for colours along a line will change in a smooth, stable manner.

In order to assess the accuracy of our technique, we named known colour regions from the RGB, HSV, and CIE LUV colour models. We began with the RGB colour cube. Monitor RGB values from our Silicon Graphics workstations were converted to CIE XYZ, and then into CIE LUV to be named. The results are shown in Figure 5.5b. Colour names change smoothly along the cube's edges from corner to corner. The names provided appear to be correct, given that the monitor's RGB colour gamut covers only a subset of all visible colours (Figure 5.5a).

One apparent problem involves the RGB cube's white corner, which is labelled purple-

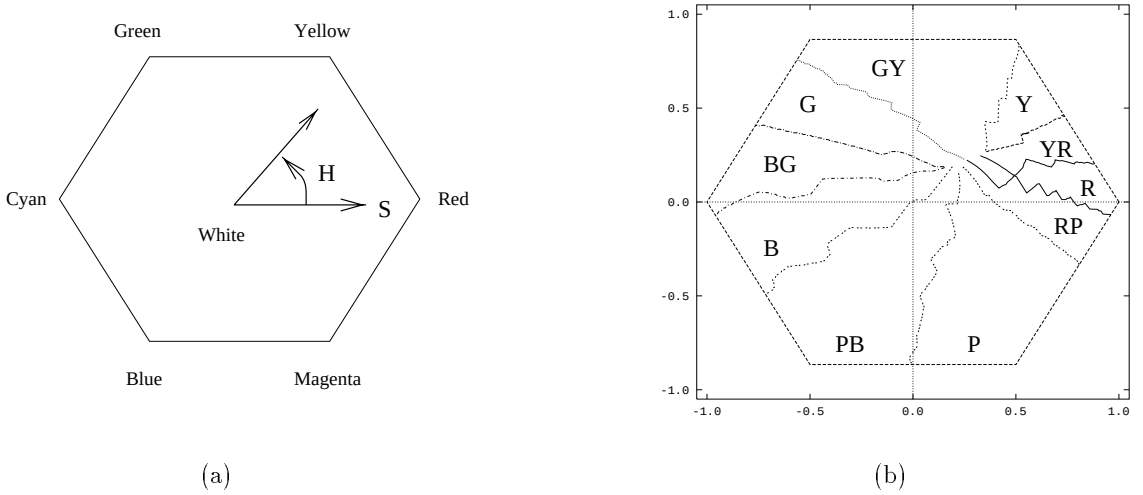
Monitor RGB	*Name*	*Munsell Patch*
(0.0, 0.0, 0.0)	Black	Black 0/0
(0.1, 0.1, 0.1)	B	5B 4/2
(0.2, 0.2, 0.2)	PB	5PB 5/2
(0.3, 0.3, 0.3)	B	5B 6/2
(0.4, 0.4, 0.4)	B	5B 7/2
(0.5, 0.5, 0.5)	B	5B 7/2
(0.6, 0.6, 0.6)	B	5B 8/2
(0.7, 0.7, 0.7)	PB	5PB 8/4
(0.8, 0.8, 0.8)	B	5B 9/2
(0.9, 0.9, 0.9)	PB	5PB 9/4
(1.0, 1.0, 1.0)	PB	5PB 9/4

Table 5.1: Names attached by our algorithm to colours along the black-white line in monitor RGB colour cube; colour constancy ensures that white, black, and grey values appear correct in the context of other monitor colours

blue rather than white. The Munsell patch matched to the white corner was 5PB 9/4, which is very close to white (Munsell patch 5PB 9/4 is two chroma "steps" from 5PB 9/0, since Munsell values from Wyszecki and Stiles are specified at even chroma values). A similar problem occurs during the naming of the cube's black corner (matched to Munsell patch 5GY 1/2). In the Munsell colour model, "white", "black", and "grey" are special cases, and are not used as hue names. We could solve this problem in certain situations by adding the colour names white, black, and grey. These correspond to Munsell colours with chroma zero and value ten (white), chroma zero and value zero (black), and chroma zero and value between one and nine (grey). Table 5.1 shows the Munsell patches and colour names we obtain for greys along the RGB cube's black-white diagonal.

The RGB cube's white corner is still named PB (although the black corner is now named "black"). This is because the monitor's white corner is not in the same location as the white point in the Munsell colour model. Colour constancy makes the white value appear correct

in the context of the other monitor colours. This last point also explains why corners of the colour cube are not named exactly as expected (*e.g.,* both the yellow and the green corners are named GY). Our naming technique is in fact showing where the monitor's gamut falls within the region of visible colours.



(a) (b)

Figure 5.6: Example of segmenting an HS-slice through the HSV hexcone at $V = 1$; (a) an HS-slice through the HSV hexcone, showing the primary colours at the corners of the hexcone, and the hue and saturation dimensions; (b) a segmented HS-slice at $V = 1$, lines represent the boundaries between the ten named categories

We continued investigating our naming technique by trying to categorize colours from a 2D slice through the HSV hexcone. The HSV colour model represents colours using the three "dimensions" hue, saturation, and value. The HSV hexcone is a projection of the RGB colour cube along the white-black diagonal. This means the colour gamut of a monitor's HSV hexcone is the same as the gamut of the monitor's RGB cube.

The $V = 1$ plane of the hexcone contains a monitor's fully saturated red, green, blue, yellow, cyan, and magenta (Figure 5.6a). Our naming algorithm was used to partition this plane into the ten named categories shown in Figure 5.6b. These regions correspond closely to the names that are normally attached to the hexcone's corners. Each category boundary is

well defined, and none of the boundaries intersect with one another. As with the RGB colour cube, the white point at the center of the hexcone falls within the purple-blue category. Our algorithm identifies what it would call white in the upper-right quadrant (somewhere in the region where the colour boundary lines converge). Again, this is due to the location of the monitor's gamut within the visible colour region.

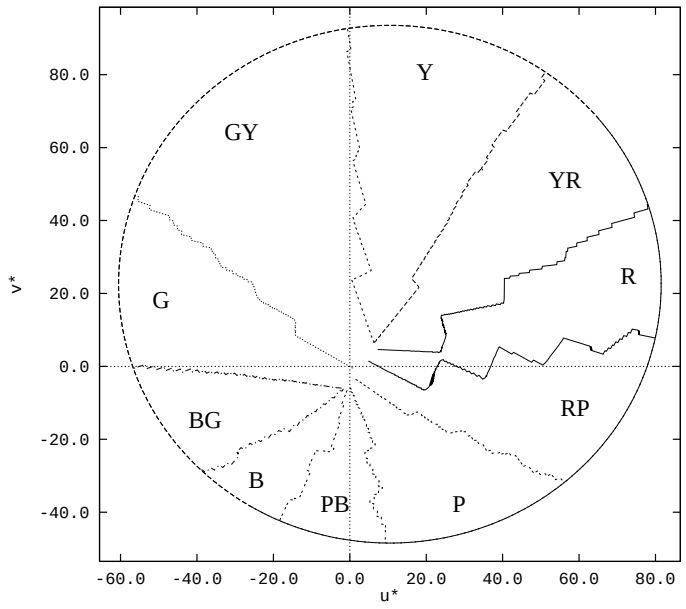


Figure 5.7: Example of segmenting a u^*v^*-slice through CIE LUV at $L^* = 71.6$; lines represent the boundaries between the ten named categories

We concluded our tests by trying to divide a 2D slice through the CIE LUV colour space. Unlike RGB and HSV, CIE LUV is not constrained by a set of monitor triad chromaticities. The CIE LUV colour model is capable of representing any colour from the visible frequency domain. We expected to see a much closer match between the white point in CIE LUV and the region our algorithm identifies as containing white, since the CIE explicitly calibrated CIE LUV to correspond closely to Munsell.

Figure 5.7 shows a circular u^*v^*-slice through the CIE LUV model at a fixed luminance of

$L^* = 71.6$. The circle was centered at $(u^*, v^*) = (10.49, 22.54)$ and had a radius of $71.0\Delta E^*$. The white point in CIE LUV is located at $(u^*, v^*) = (0, 0)$. As with the HSV example, each category boundary is clearly defined. Boundaries do not cross one another. Moreover, the colour boundary lines terminate as expected at the region that contains the CIE LUV white point.

5.4 Evaluating Colour Categories

Results from our automatic segmentation algorithm suggest it is accurate and stable for a wide range of colour models and visible colours. Observers agreed that the names attached to individual colours were appropriate given the set of names available through the Munsell hue dimension. This does not mean that observers would have chosen exactly the same names used by the algorithm to describe each colour, however. For example, most people do not use the name "purple-blue" to describe a colour. They are more likely to use names like dark blue, grey blue, or indigo. In order to measure colour category effects, we needed a method for segmenting a colour gamut into user-named categories. We used our algorithm to automatically divide the colour gamut into an initial starting state. From there, we picked one or more representative colours from each category and ran a limited set of colour naming tests. Results from these experiments allow us to specify the perceptual overlap, strength, and user-chosen name of each category.

Method

We decided to try to assess the categories along the circumference of the u^*v^*-circle shown in Figure 5.7. Our segmentation algorithm was used to divide the circle's boundary into ten initial categories. We needed to pick a representative colour from each category to use

during our naming experiment. The colour at the center of the category was chosen for this purpose (Figure 5.8).

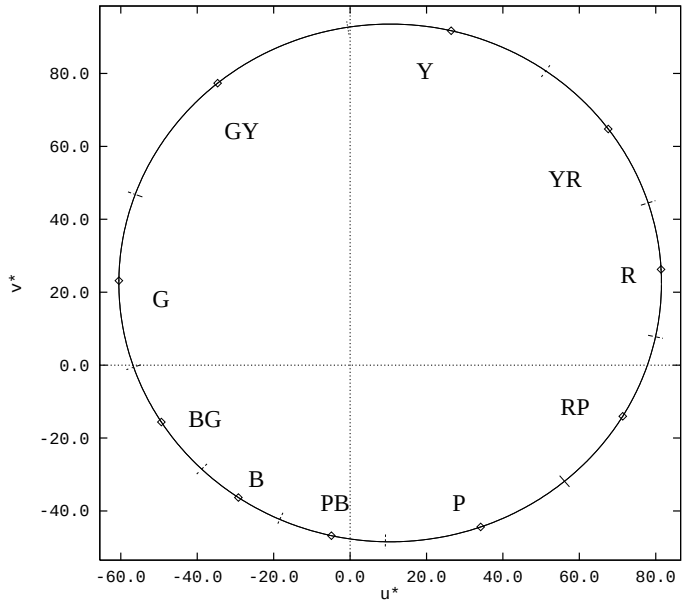


Figure 5.8: A u^*v^*-slice through CIE LUV at $L^* = 71.6$; ticks along the the circle mark the boundaries between the ten named categories, points are the representative colours for each category

Thirty-eight observers with normal or corrected acuity volunteered to participate in our experiment. Each observer was tested to ensure they were not colour blind. They were then asked to name each of the ten representative colours. Observers were told to give a simple name that accurately described each colour. No restrictions were placed on the names they were allowed to use.

Our experiment was conducted on a Macintosh computer with a sixteen inch colour monitor and video hardware with an eight bit colour lookup table. The software used was designed and written specifically to run vision experiments [Enns and Rensink, 1991]. The chromaticities of the monitor's triads and maximum-intensity red, green, and blue were measured to build an accurate CIE LUV to monitor RGB conversion matrix.

During the experiment the representative colours were shown to each observer in a random order. Each colour was displayed as a rectangular patch that filled approximately 90% of the screen. The border around the outside of the patch was coloured light grey, and had a luminance $L^* = 71.6$ equal to the luminance of the colour patches. Observers were allowed to view the patch for as long as they wanted before providing a name. Observers were told that they would not be shown the same colour more than once. The names provided by each observer were recorded for later analysis.

Results

Observers provided us with a wide range of different colour names. We compressed these results by combining names that described the same colour or combination of colours. For example, the names "purple", "violet", and "mauve" from the P category were combined under the name "purple". The names "aqua", "turquoise", "blue green", "green blue", and "sea green" from the BG category were combined under the name "aqua". Figure 5.2 shows the names chosen for each of the ten representative colours. The frequency of each name is also provided. A name's frequency is the percentage of observers who chose that name to describe the given colour patch.

Even a brief review of the results in Table 5.2 leads to a number of interesting conclusions. For example, both the G and GY categories were almost exclusively named "green" (with frequencies of 100% and 97.4%, respectively). This suggests that, in terms of user-chosen names, G and GY should be collapsed into a single "green" category. The B and PB categories (both named "blue" with frequencies 89.4% and 92.1%, respectively) might also be collapsed into a single "blue" category. The R category appears to lack saturation; combining the frequencies for the names "red" and "pink" (unsaturated red) results in near-total coverage of 97.3%. Similarly, the Y category lacks luminance; a combination of the

	purple	*magenta*	*pink*	*red*	*orange*	*brown*	*yellow*	*green*	*aqua*	*blue*	*other*
P	86.9%	2.6%	5.2%	–	–	–	–	–	–	–	5.2%
RP	15.7%	28.9%	55.3%	–	–	–	–	–	–	–	–
R	–	–	26.3%	71.0%	–	–	–	–	–	–	2.6%
YR	–	–	–	5.3%	86.8%	7.9%	–	–	–	–	–
Y	–	–	–	–	2.6%	44.7%	47.4%	–	–	–	5.2%
GY	–	–	–	–	–	–	–	97.3%	–	–	2.6%
G	–	–	–	–	–	–	–	100.0%	–	–	–
BG	–	–	–	–	–	–	–	26.3%	57.8%	15.8%	–
B	–	–	–	–	–	–	–	–	7.9%	89.4%	2.6%
PB	5.2%	–	–	–	–	–	–	–	–	92.1%	2.6%

Table 5.2: Frequency of user-chosen names for each of the ten named category, represented by percentage of observers who chose each name to describe the given colour

names "yellow" and "brown" (dull yellow) results in a total frequency of 92.2%.

Other categories appear to cover a range of user-chosen names. The RP category is split between "pink" (or unsaturated red, 55.3%), "purple" (15.7%), and "magenta" (or red-purple, 28.9%). Observers named this category using one or both of its primary components (either red or purple). The BG category was named either "green" (26.3%), "aqua" (or blue-green, 57.8%), or "blue" (15.8%). Observers were told after the experiment that the Munsell name for this colour was "blue-green". Most observers agreed that "blue-green" was a descriptive and accurate name (in spite of the fact that almost no one chose it for themselves). This suggests that Munsell names are not necessarily poor, rather they may be uncommon in most people's vocabulary.

5.5 Perceptual Overlap

Our results show clearly that certain neighbouring categories overlap in the names chosen to represent the category. Kawai et al. [1995] suggest that a user's ability to detect a colour

"target" will decrease dramatically when background elements use colours from the target's category. We could logically extend this to imply that a similar decrease in performance will occur if background colours are chosen from neighbouring but overlapping categories. In order to predict (and avoid) this type of visual interference, we need a method to measure perceived category overlap. This type of overlap depends on:

- the range of user-chosen names assigned to a given category
- the frequency of a user-chosen name
- how the ranges of neighbouring categories overlap

As an example, consider the R and YR categories, which have user-chosen name ranges:

	pink	*red*	*orange*	*brown*
R	26.3%	71.0%		
YR		5.3%	86.8%	7.9%

Colours from R and YR overlap only at the "red" name; their overlap is not that strong, since the frequency of "red" for the YR category is low. We computed the amount of overlap by multiplying the frequencies for the common name. This gives an R-YR overlap of $71.0\% * 5.3\% = 0.0376$. A closer correspondence of user-chosen names for a pair of categories results in a much larger overlap. For example, given the user-chosen name ranges for GY and G:

	green	*other*
GY	97.4%	2.6%
G	100.0%	

the GY-G overlap is $97.3\% * 100.0\% = 0.973$. Colours that overlap over multiple names are combined using addition, for example, BG and B have ranges:

	green	*aqua*	*blue*	*other*
BG	26.3%	57.8%	15.8%	
B		7.9%	89.4%	2.6%

for a BG-B overlap of $(57.8\% * 7.9\%) + (15.8\% * 89.4\%) = 0.187$. When we use this algorithm to compute the overlap between each of the ten named categories, we obtain the overlap table shown in Table 5.3. Values from this table can be used to predict when certain colours might be difficult to distinguish from one another.

	R	YR	Y	GY	G	BG	B	PB	P	RP
R	—	.038	0	0	0	0	0	0	.014	.145
YR	.038	—	.058	0	0	0	0	0	0	0
Y	0	.058	—	0	0	0	0	0	0	0
GY	0	0	0	—	.973	.256	0	0	0	0
G	0	0	0	.973	—	.263	0	0	0	0
BG	0	0	0	.256	.263	—	.187	.146	0	0
B	0	0	0	0	0	.187	—	.823	0	0
PB	0	0	0	0	0	.146	.823	—	.045	.008
P	.014	0	0	0	0	0	0	.045	—	.173
RP	.145	0	0	0	0	0	0	.008	.173	—

Table 5.3: Table showing the degree of user-name overlap between representative colours from each of the ten named categories

A final characteristic that we measured was the strength of each named category. We defined a category to be strong when:

- it has one high frequency user-chosen name
- it has relatively low perceptual overlap with neighbouring categories

Using this definition and results for categories along our CIE LUV colour wheel, we classified YR (with a total perceptual overlap of 0.096) and P (with an overlap of 0.232)

as being strongly named "orange" and "purple". The G-GY categories might be collapsed into a single strong "green" category (overlapping somewhat with BG). Similarly, B-PB might be collapsed into a single strong "blue" category (again overlapping somewhat with BG). Although categories like R and YR have low perceptual overlap, there was no common agreement on a single name to describe them. Users were evenly split between "red" and "pink" for the R category, and "yellow" and "brown" for the Y category.

5.6 Experiment 1: Distance and Separation

We began our investigation of colour selection by controlling colour distance and linear separation, but not colour category. Although Kawai's results on colour category are intriguing, researchers in psychology are still divided on the question of whether the way in which people name colours has an affect on the perceptual division of a colour space into colour regions. We deliberately ignored colour category to see what effect this would have on our colour target search task. Results from our experiment showed inconsistent performance. When we considered colour categories during colour selection, we were able to predict and avoid this visual interference effect. We were also able to increase the number of colours we could simultaneously display, which is an important factor to consider. This shows that, at least during target detection, how users name colours can influence the perceived difference between the colours.

As discussed in the introduction to this chapter, we proceeded under the assumption that a user might choose to search for any one of the different types of data element at any given time. Our requirement meant that we could not predefine which elements were targets and which were non-targets. The colour selection technique had to allow for rapid and accurate identification of any of the elements being displayed.

We ran four studies to investigate the tradeoff between the number of colours displayed and the time required to determine the presence or absence of a target element. Table 5.5 lists the names and the monitor RGB values of all the colours used in the studies. Each study displayed a different number of unique colours:

- *three-colour study:* each display contained three different colours (*i.e.,* one colour for the target and two for the non-targets, Figures 5.9a–5.9b)
- *five-colour study:* each display contained five different colours (*i.e.,* one for the target and four for the non-targets, Figures 5.9c–5.9d)
- *seven-colour study:* each display contained seven different colours (*i.e.,* one for the target and six for the non-targets, Figures 5.9e–5.9f)
- *nine-colour study:* each display contained nine different colours (*i.e.,* one for the target and eight for the non-targets, Figures 5.9g–5.9h)

Every colour in a given study was tested as a target. For example, the three-colour study was run three times, first with an R element acting as a target (and GY and PB elements acting as non-targets), next with a GY target (and R and PB non-targets), and finally with a PB target (and R and GY non-targets). Faster search times for certain targets would have to be explained in terms of colour distance, linear separation, or colour category. Each of the four studies were themselves divided in the following manner:

- half of the displays were randomly chosen to contain an element that used the target colour; the other half did not
- one-third of the displays contained a total of 17 elements (one target and 16 non-targets if the target was present, or 17 non-targets if the target was absent); one-third of the

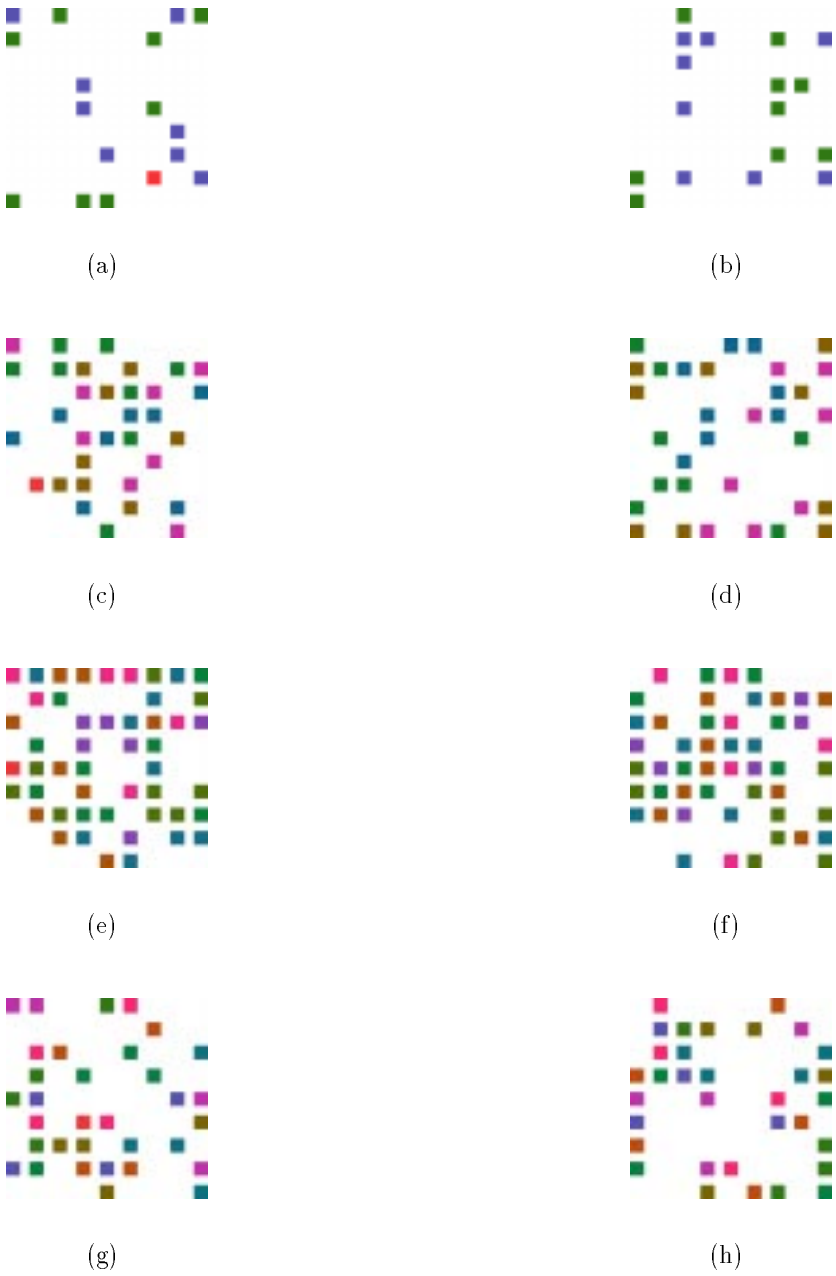


Figure 5.9: Example displays from each of the four studies, in each figure the target is a red element: (a)–(b) three-colour study, 17 elements, target present in (a), absent in (b); (c)–(d) five-colour study, 33 elements, target present in (c), absent in (d); (e)–(f) seven-colour study, 49 elements, target present in (e), absent in (f); (g)–(h) nine-colour study, 33 elements, target present in (g), absent in (h)

displays contained a total of 33 elements (either one target and 32 non-targets, or 33 non-targets); one-third of the displays contained a total of 49 elements (either one target and 48 non-targets, or 49 non-targets)

- elements in each display were randomly located in an underlying 9×9 grid that covered the entire viewing area of the monitor

Colours for each of the four studies were chosen such that the colour distance between pairs of colours and the linear separation for each colour were fixed to constant values (as described below). Our results showed that detection was rapid and accurate for all colours from both the three-colour and five-colour studies. Results from the seven-colour and nine-colour studies were mixed; some colours gave better performance than others. This difference was explained by examining the colour regions occupied by each of the colours.

Method

We began the colour selection process by obtaining the chromaticities of our monitor's triads. We also measured the luminance of the monitor's maximum intensity red, green, and blue with a spot photometer.

We chose colours with the same perceived intensity, because previous research has shown that random variation in intensity can interfere with an observer's ability to perform visualization tasks based on colour [Callaghan, 1984]. All the colours were chosen from a single u^*, v^*-slice through the CIE LUV colour space at $L^* = 67.1$. We wanted to maximize the number of available colours, while still maintaining control over colour distance and linear separability. To do this, we computed the monitor's gamut in the $L^* = 67.1$ slice. We then found the largest circle inscribed within the gamut (Figure 5.10).

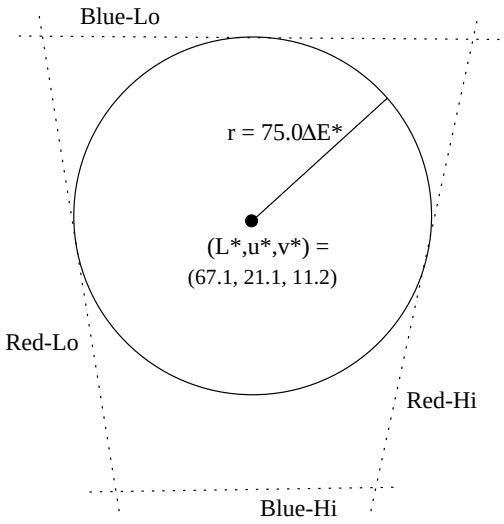


Figure 5.10: An diagram of the monitor's gamut at $L^* = 67.1$, along with the maximum inscribed circle centered at $(67.1, 21.1, 11.6)$, radius $75.0\,\Delta E^*$

Given this largest inscribed circle, we chose colours that were equally spaced around its circumference. For example, during the five-colour study we chose colours at positions 14°, 86°, 158°, 230°, and 302° counterclockwise rotation from the x-axis (Figure 5.1). This method ensured that neighbouring colours had a constant colour distance. It also ensured that any colour acting as a target had a constant linear separation from every other (non-target) colour. A similar technique was used to select colours for the three-colour, seven-colour, and nine-colour studies. This gave us the colour distances d and linear separations l shown in Table 5.4.

Study	d	l
3-colour	$129.9\,\Delta E^*$	$112.5\,\Delta E^*$
5-colour	$88.2\,\Delta E^*$	$51.9\,\Delta E^*$
7-colour	$65.1\,\Delta E^*$	$28.4\,\Delta E^*$
9-colour	$51.3\,\Delta E^*$	$17.6\,\Delta E^*$

Table 5.4: The constant colour distance d and linear separation l for each of the four studies; distances are measured in ΔE^* units

Our experiments were conducted on a Macintosh computer with a fourteen inch colour monitor and video hardware with an eight-bit lookup table. The software used was designed and written specifically to run vision experiments [Enns and Rensink, 1991]. Thirty-eight users with normal or corrected acuity participated as observers during our studies. Observers were tested to ensure they were not colour blind, using the Insight-2 Colour Vision Test. The Insight-2 software is designed to run on Macintosh computers; its colour vision test is based on the Farnsworth-Munsell 100-hue and dichotomous colour vision experiments. After the colour vision test, each observer was asked to complete one or more experiment subblocks. A subblock consisted of 360 displays testing a single colour target from one of the four studies. A total of 66 subblocks were run, divided to roughly balance the number of observers who completed each study (Table 5.5).

The procedure and task of each experiment were introduced to the observers before they began a testing session. For example, before starting a subblock from the seven-colour experiment observers were shown a sample display frame. The colour of the target element was identified within the display. Observers were told that during the experiment they would have to determine whether an element with the given target colour was present or absent in each display they were shown. Observers were asked to answer as quickly as possible, but also to ensure that they answered correctly. They were then shown how to enter their answer (either "present" or "absent"). This was done by typing a letter on the keyboard. Observers were allowed to choose one letter that corresponded to "present", and another that corresponded to "absent".

Observers were given an opportunity to practice each subblock before they began the actual experiment. A practice session consisted of thirty-six trials presented in a random order. The target colour used during the practice session was the same as the target colour used during the experiment subblock. The number of colours present in each display was also the same as in the experiment subblock (*i.e.,* practice sessions during the seven-colour

Study	Target Colour	Monitor (R,G,B)	Number of Subjects	Number of Trials	Average Error
3-colour	R	(243, 51, 55)	4	1440	2.4%
3-colour	GY	(49, 121, 20)	4	1440	1.6%
3-colour	PB	(88, 82, 175)	4	1440	3.4%
5-colour	R	(243, 51, 55)	3	1080	2.6%
5-colour	Y	(134, 96, 1)	3	1080	3.1%
5-colour	GY	(18, 127, 45)	3	1080	2.1%
5-colour	B	(36, 103, 151)	3	1080	1.8%
5-colour	P	(206, 45, 162)	3	1080	1.9%
7-colour	R	(243, 51, 55)	3	1080	5.3%
7-colour	Y	(171, 83, 7)	3	1080	4.9%
7-colour	GY	(78, 114, 8)	3	1080	3.6%
7-colour	G	(9, 128, 58)	3	1080	3.1%
7-colour	BG	(20, 111, 137)	3	1080	3.0%
7-colour	P	(132, 66, 180)	3	1080	3.3%
7-colour	RP	(239, 39, 134)	3	1080	3.0%
9-colour	R	(243, 51, 55)	2	720	3.1%
9-colour	YR	(191, 75, 13)	2	720	5.2%
9-colour	Y	(119, 101, 1)	2	720	3.3%
9-colour	GY	(49, 121, 20)	2	720	13.9%
9-colour	G	(5, 128, 66)	2	720	14.8%
9-colour	BG	(13, 115, 128)	2	720	9.8%
9-colour	PB	(88, 82, 175)	2	720	2.8%
9-colour	P	(189, 49, 170)	2	720	4.1%
9-colour	RP	(249, 38, 116)	2	720	16.1%

Table 5.5: Table listing for each target tested within the four colour studies: the colour name and monitor RGB value of the target, the number of subjects who ran the given subblock, the total number of trials completed, and the average error rate

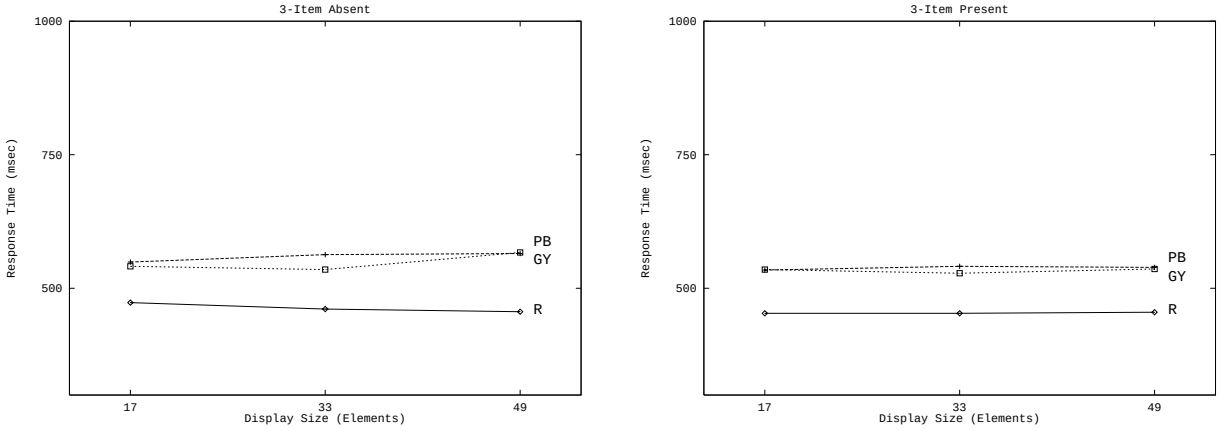
study used displays with seven different colours: one for the target and six for the non-targets). The thirty-six displays were divided equally across display size: one-third of the displays contained 17 elements, one-third contained 33 elements, and one-third contained 49 elements. For each display size, half the trials contained an element that used the target colour and half did not. The practice session was designed to give observers an idea of the experiment procedure and the speed of the trials.

After completing a practice session, observers began the actual experiment subblock. Each subblock consisted of 360 displays. Observers were given an opportunity to rest after every 60 displays. Responses (either "present" or "absent") and the time to respond for each display an observer completed were recorded for later analysis.

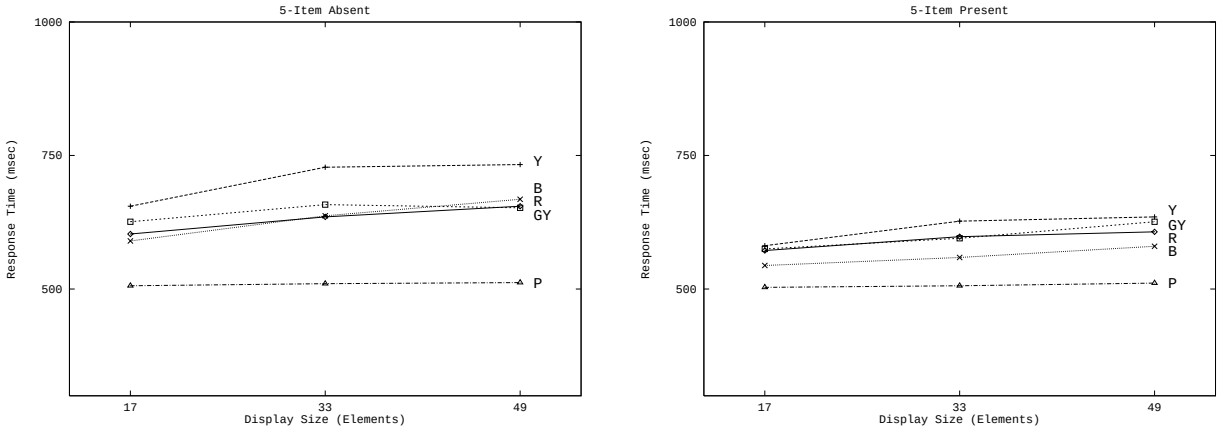
Results

Observers had very little difficulty identifying targets during the three-colour and five-colour studies. Graphs of mean response time across display size were relatively flat for every colour (Figure 5.11). During the three-colour study response times ranged from 453 msec to 532 msec for target present displays, and from 463 msec to 547 msec for target absent displays. Results for the five-colour study were similar, with mean response time ranges of 504 msec to 592 msec for target present displays, and 509 msec to 704 msec for target absent displays. Mean response error for the three-colour and five-colour studies was 2.5% and 2.3%, respectively.

There was a small increase in mean response times when we moved from the three-colour to the five-colour study. This is thought to correspond to the increase in the number of unique colours being displayed. Hick [1952] showed that when users expect an increased number of choices their absolute response times increase logarithmically in the number of choices available. We believe that the increase in the number of unique colour stimulae in



(a)



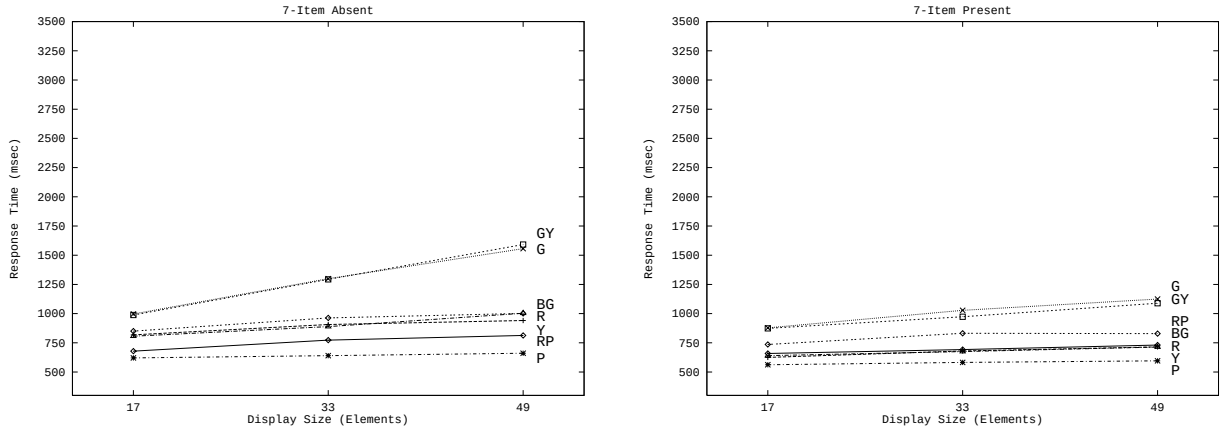
(b)

Figure 5.11: Response time graphs for the three-colour and five-colour studies, in each figure the graph on the left represents displays where the target was absent, while the graph on the right represents displays where the target was present: (a) response time as a function of display size (*i.e.,* total number of elements shown in the display) for each target from the three-colour study; (b) response time as a function of display size for each target from the five-colour study

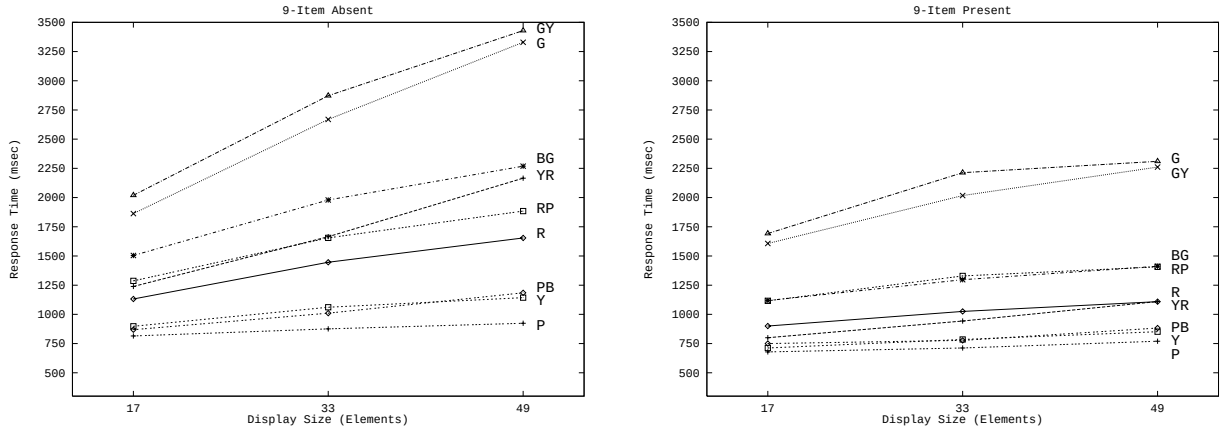
our displays produces a similar effect; this would explain the small y-intercept increase in response times observed during the five-colour study.

Analysis of variance (ANOVA) results showed a significant difference in response time across target colour during the three-colour experiment, $F(2, 4209) = 224.6$, $p < 0.0001$ (main effects with a p-value of less than 5% were considered significant). For the five-colour experiment, ANOVAs identified a significant difference in response time across both target colour and display size ($F(4, 5256) = 194.0$, $p < 0.0001$ and $F(2, 5258) = 49.2$, $p < 0.0001$, respectively). Significant reaction time differences across target colour are probably caused by performance differences in individual subjects, since only a small number of subjects completed trials for a given colour (four subjects per target colour during the three-colour study, and three subjects per target colour during the five-colour study).

Scheffé values suggest that the significance across display size during the five-colour study is due to lower reaction times during the 17 element displays ($p < 0.0001$ for the (17, 33) and (17, 49) display size pairs, $p = 0.03$ for the (33, 49) display size pair). This can be seen visually as a flattening of the response time graphs in Figure 5.11b (particularly for target absent displays) when display sizes increased. As we moved from 33 to 49 element displays, we observed an average mean reaction time increase of $+0.80$ msec per element, a minimum of -0.25 msec per per element (for GY targets during target absent displays), and a maximum of $+1.98$ msec per element (for GY targets during target present displays). We expect similar response time increases for displays larger than 49 elements, although additional experiments would be required to confirm this hypothesis. Based on the absolute response times and per element response time increases, we concluded that users could accurately identify the target in all cases, and that the time required to do so was sufficiently independent of display size for our purposes. This suggests that, even when using five different colours, the visual system can search for any one of the colours in parallel.



(a)



(b)

Figure 5.12: Response time graphs for the seven-colour and nine-colour studies, in each figure the graph on the left represents displays where the target was absent, while the graph on the right represents displays where the target was present: (a) response time as a function of display size for each target from the seven-colour study; (b) response time as a function of display size for each target from the nine-colour study

Target identification became much more difficult for certain colours during the seven-colour and nine-colour studies. Mean response error during the seven-colour study was still low, at approximately 3.3%. The P, Y, R, BG, and RP targets each exhibited relatively flat response time graphs. Mean response time for these elements ranged from 562 msec to 829 msec for target present displays, and from 621 msec to 1003 msec for target absent displays. The G and GY targets, however, gave response times typical of serial search. An increase in the number of elements being displayed brought on a corresponding increase in response time. A linear regression fit to results for the G target produced $t = 695.1 + 7.7n$, $r = 0.23$ and $t = 745.6 + 17.5n$, $r = 0.54$ for target present and target absent displays, respectively. A similar calculation for the GY target produced $t = 764.3 + 6.7n$, $r = 0.19$ and $t = 684.0 + 18.8n$, $r = 0.46$. The reaction time increase per additional element for target-absent displays (approximately 17 msec per additional element for G, and 19 msec per element for GY) was roughly twice that for target-present displays (8 msec and 7 msec per element for G and GY, respectively). Observers had to search through, on average, half the elements before they found the target in target-present displays. In target-absent displays, however, they had to search through all the elements to confirm that no target existed. This explains why per item search time increased roughly twice as fast for target-absent displays.

A similar set of results was obtained during the nine-colour study. Overall mean response error increased to 8.1%; it was lowest for the R, Y, PB, and P targets (3.1%, 3.3%, 2.8%, and 4.1%, respectively) and highest for the GY, G, and RP targets (13.9%, 14.8%, and 16.1%, respectively). The Y, PB, and P targets displayed relatively flat response time graphs during target-present displays (although response time did increase with display size during target-absent displays). The remaining targets showed some form of serial search. The effect was weakest for the R and YR targets, and strongest for the G and GY targets.

5.7 Colour Category Integration

Results from our four studies showed that controlling colour distance and linear separation alone is not enough to guarantee consistently good (or consistently bad) target identification for every colour. Results from Kawai et al. suggest that colour category can also have a strong effect on the amount of time required to identify a colour target. We decided to see whether colour category results could explain the asymmetric response times we observed during the seven-colour and nine-colour studies. We used our colour segmentation technique to place individual colours within a named colour region.

In the seven-colour study the P, Y, R, BG, and RP targets gave good performance. The G and GY targets gave poor performance. An examination of the perceptual overlap table (Table 5.3) suggests the following explanation:

- targets R, Y, P, and RP have a weak overlap with the other colours used during the study: Y has no overlap to the other colours, R and RP have an overlap of 0.145, and P and RP have an overlap of 0.173
- target BG has a moderate overlap of 0.256 and 0.263 with GY and G, respectively
- targets G and GY have a moderate overlap with BG, and a very strong overlap of 0.973 with one another

For each target in the seven-colour study, its perceptual overlap with the other colours corresponded closely to its mean response time. We can measure this correspondence by computing Spearman's correlation coefficient on the rank order of our colours in terms of total overlap and mean response time.

Table 5.6 sums the overlap measures for each of the seven colours we used. The rank order of total overlap from lowest to highest is Y, R, P, RP, BG, GY, and G. Ranking our

	R	Y	GY	G	BG	P	RP
R	—	0	0	0	0	.014	.145
Y	0	—	0	0	0	0	0
GY	0	0	—	.973	.256	0	0
G	0	0	.973	—	.263	0	0
BG	0	0	.256	.263	—	0	0
P	.014	0	0	0	0	—	.173
RP	.145	0	0	0	0	.173	—
Total	.159	0	1.229	1.236	.519	.187	.318

Table 5.6: Perceptual overlap table, showing individual and total overlap values for each of the colours used during the seven-colour study

colours based on total mean response time (including both present and absent trials across all three display sizes) from smallest to largest gave an order of P, R, Y, BG, RP, GY, G. The Spearman correlation between these rankings is $r_{rank} = 0.821$, confirming that higher mean response times for a given target correspond to a higher colour category overlap.

A category overlap–response time correlation can also be observed during the nine-colour study:

- targets P, Y, YR, and PB have a weak overlap with the other colours used during the study (PB and BG have a similarity of 0.146, and P and RP have an overlap of 0.173)
- targets R, RP, and BG have a moderate overlap with the other colours (RP has an overlap of 0.145 and 0.173 to R and P, respectively, while BG has an overlap of 0.256 and 0.263 to G and GY, respectively)
- as in the seven-colour study, G and GY have a moderate overlap with BG, and a very strong overlap with one another

The Spearman correlation between the colours' total similarity and mean response time

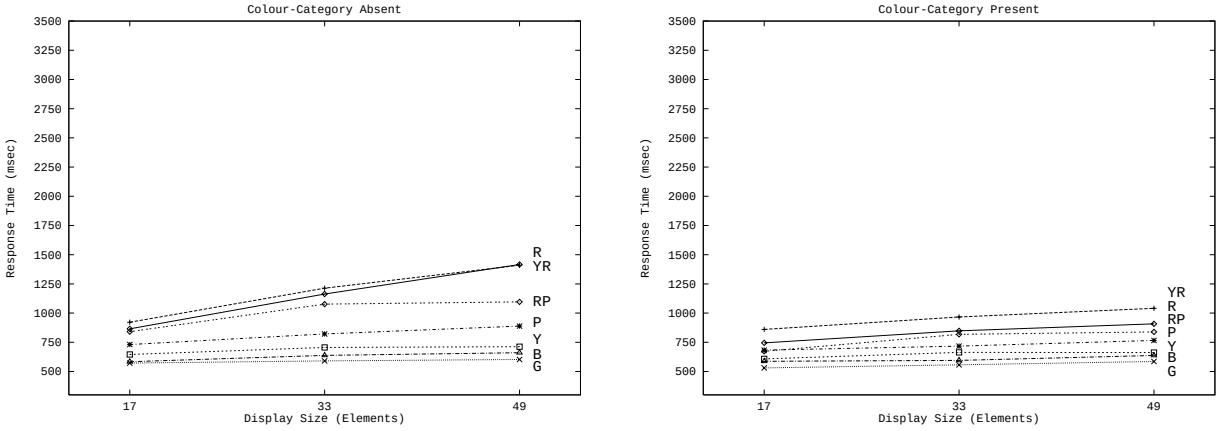


Figure 5.13: Response time graphs for the first colour-category study, the graph on the left represents displays where the target was absent, while the graph on the right represents displays where the target was present

rankings was $r_{rank} = 0.762$. Results for YR and BG are somewhat anomalous; values from the overlap table suggest we should have observed better performance for the YR target compared to the BG target. Results for the other colours correspond closely to their perceptual overlap measures.

5.8 Experiment 2: Colour Category

Results from investigating colour category integration might imply that effective colours can be selected by controlling colour category alone. This was tested by selecting seven colours from colour regions that had low overlap with one another. An examination of the similarity table in Figure 5.2 shows that colours chosen from the R, YR, Y, G, B, P, and RP regions satisfy our restriction; the largest overlap occurs between RP-R (0.145) and RP-P (0.173). The colours we chose were exactly those used as representative colours during the colour naming experiments (Figure 5.8). Because these were the colours used to obtain the user-chosen names for each region, their overlap with one another corresponds exactly to the values in the similarity table. A single observer completed 360 displays for each target.

	CW d (in ΔE^* units)	CCW d (in ΔE^* units)	l (in ΔE^* units)
R	42.0	42.0	12.2
YR	42.0	49.8	14.6
Y	49.8	112.6	39.0
G	112.6	67.5	52.9
B	67.5	63.2	29.1
P	63.2	48.6	20.4
RP	48.6	42.0	14.2

Table 5.7: The exact distance in CIE LUV between a colour, its clockwise neighbour (CW d), and its counter-clockwise neighbour (CCW d), as well as the linear separation from the other colours when it acted as a target

Mean response error was 3.8%, which shows responses were accurate. Although response times are somewhat better than those from the original seven-colour study, several colours (in particular R, YR, and RP) still exhibit poor search performance. Linear regression fits suggest the R target had a per element response time increase of 5.1 msec and 17.2 msec for target present and target absent displays, respectively. Results for the P target were 5.7 msec and 15.2 msec per element; for the RP target they were 5.1 msec and 7.8 msec per element. This can be explained by examining the distance between neighbouring pairs of colours, and the linear separation for each colour when it acted as a target (Figure 5.7). Colours that gave the worst search performance had the smallest neighbour distances and linear separation (R, YR, and RP); in fact, the R, YR, and RP targets had a linear separation that was smaller than the one used during the nine-colour study. Colours that gave the best search performance had the largest neighbour distances and linear separation (G, B, and Y).

5.9 Experiment 3: Colour Selection

Results from our colour category experiment show that category alone cannot be used to ensure consistently good identification based on colour. Colour distance and linear separation need to be considered, because they do have an effect on search performance. We used a more systematic colour selection technique to choose another set of seven colours:

- a single green was chosen from the combined G-GY colour region; observers see the entire region as green, which means it can be used for only a single colour
- the clockwise neighbour of our green was a yellow, chosen to lie on the border between the GY and Y colour regions
- the counterclockwise neighbour of our green was a blue-green, chosen to lie in the center of the BG colour region (we did not use the colour from the border between G and BG, because it was too difficult to differentiate from our green)
- the remaining four colours (chosen from the YR, R, RP, and PB colour regions) were at equal steps between our yellow and our blue-green

This gave us a constant neighbour distance d and linear separation l ($59.4\,\Delta E^*$ and $24.6\,\Delta E^*$, respectively) between the Y, YR, R, RP, PB, and BG colours (G had a larger d and l than the other colours). Results from displays using these colours as targets are shown in Figure 5.14. Mean response error was 5.6%. Response time graphs for all seven colours are much flatter than in the original seven-colour study, although G and Y still give mixed results during target-absent displays (response time increases as we moved from 33 to 49 element displays of 3.5 msec and 5.5 msec per element, respectively). We could have further differentiated these elements by choosing a yellow from the center of the Y colour region (rather than at the GY-Y border). This might have resulted in poorer performance for other

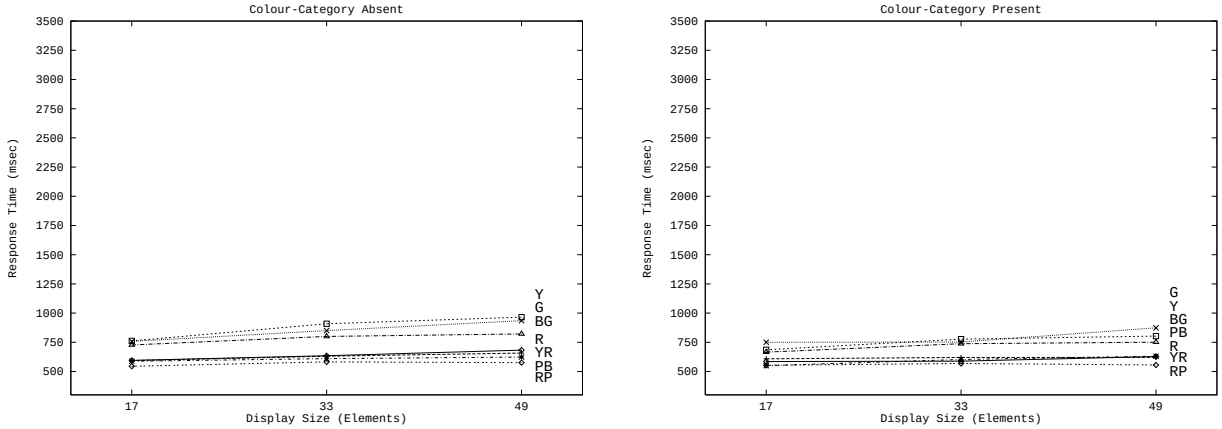


Figure 5.14: Response time graphs for the second colour-category study, the graph on the left represents displays where the target was absent, while the graph on the right represents displays where the target was present

targets due to the reduction in colour distance d and linear separation l between our Y, YR, R, RP, PB, and BG colours, however. It appears that seven isoluminant colours is the maximum we can display at one time, while still allowing rapid and accurate identification of any one of the colours.

Chapter 6

Real-Time Visualization

The techniques we have discussed thus far are all based on static display frames. A static frame is shown to a user in isolation, usually for a fixed exposure duration. After that time the display is cleared, and the user tries to answer questions based on visual features that were present in the display (*e.g.,* target detection, boundary detection, region identification, or estimation). Work to date in preattentive processing has focused on finding features and tasks that occur in exactly these kinds of static displays.

An obvious drawback to static frames is their maximum resolution, which is limited by the maximum screen size in pixels. We decided to try to extend our static techniques to a dynamic visualization environment. A dynamic environment displays a sequence of frames to the user one after another. Each frame is shown for a fixed period of time, after which it is replaced by the next frame in the sequence. An obvious question to ask is: If I can perform tasks in 200 msec on a static frame, can I perform the same tasks on a sequence of frames displayed at five frames per second?

We believe dynamic visualization techniques can be used to address the problem of large datasets. Although we did not explicitly study methods for decomposing a dataset into a sequence of two-dimensional display frames, many datasets can be easily modified to satisfy this requirement. For example, time-varying datasets can be subdivided along the time axis.

Three-dimensional volumetric models can be decomposed into a stack of two-dimensional slices. If a dataset can be reduced in this manner, it can be analysed using our dynamic visualization techniques. Recall that our definition of a large dataset is one containing on the order of one million elements. Suppose a one million element dataset is reduced to 2,500 individual display frames, each of which contains 400 elements. If each frame could be displayed and analysed in 200 msec, the entire dataset could be viewed in 500 seconds (or just over eight minutes). Assuming users spend only 20% of the time browsing the dataset (and 80% performing other visualization and analysis tasks), they could still view the entire dataset in about 45 minutes.

We approached real-time multivariate visualization by defining a set of requirements that we feel are inherent to this class of problem:

- *multidimensional data:* the technique should be able to display multidimensional data in a two-dimensional environment, the computer screen
- *shared data:* the technique should display independent data values simultaneously; a single user could choose to examine various relationships, or multiple users could simultaneously examine independent data values
- *real-time data:* the technique must function in a real-time environment, where frames of data are continuously generated and displayed one after another
- *speed:* the technique should allow users to rapidly obtain useful and nontrivial information; here, "rapidly" means in less than 250 msec per data frame
- *accuracy:* information obtained by the users should accurately represent the relationship being investigated

Using an approach that extends our previous work on static visualization, we decided to

use preattentive processing to assist with real-time (dynamic) multivariate data visualization. We hypothesized that important aspects of preattentive processing will extend to a real-time environment. In particular, we believe real-time visualization techniques based on preattentive processing will satisfy the five requirements listed above. A visualization tool that uses preattentive features will allow viewers to perform rapid and accurate visual tasks such as grouping of similar data elements (boundary detection), detection of elements with a unique characteristic (target detection), and estimation of the number of elements with a given value or range of values, all in real-time on temporally animated data frames. We tested this hypothesis using behavioral experiments that simulated our preattentive visualization tools. Analysis of the experimental results supported our hypothesis for boundary and target detection. Moreover, interference properties previously reported for static preattentive visualization were found to apply to a dynamic environment.

6.1 Experiment 1: Boundary Detection

Through experimentation we sought to determine whether or not research in preattentive processing can help design more useful and intuitive scientific visualization tools. Specifically, we investigated whether preattentive tasks and interference effects extend to a real-time visualization environment, where frames of data are displayed one after another. Our first experiment addressed two general questions about preattentive features and their use in our visualization tools.

- *Question 1:* Is it possible for subjects to detect a data frame with a horizontal boundary within a sequence of random frames? If so, what features allow this and under what conditions?

- *Question 2:* Do Callaghan's feature preference effects apply to our real-time visualization environment? Specifically, does random hue interfere with form boundary detection within a sequence of frames? Does random form interfere with hue boundary detection within a sequence of frames?

These questions were designed to address the requirements described in the introduction to this chapter. Detection of boundaries and groups is one example of a common data analysis task. If preattentive features can be used to help perform the task, we could employ this technique to rapidly explore large amounts of data. Real-time applications could also use this technique for effective real-time visualization. Evidence that boundary detection and corresponding interference effects occur as expected in a real-time environment would imply that other preattentive tasks (*e.g.,* target detection, counting, and estimation) might also extend naturally. The ability to encode multiple unrelated data values in a single display would allow users to visualize multidimensional datasets, or to "share" the display, but only in cases where no interference occurs.

We decided to examine two preattentive features, hue and form. This was done by running experiments that displayed 14×14 arrays of coloured circles and squares (Figures 6.1 and 6.2). These features are commonly used in existing visualization software. Both hue and form have been shown to be preattentive by Triesman, Julész, and others [Julész and Bergen, 1983; Triesman, 1985]. Moreover, Callaghan's research has shown that hue exhibits a strong interference effect over form during certain preattentive tasks [Callaghan, 1989]. All of this suggests that studying how hue and form interact in a preattentive environment is both an important and an interesting question.

Method

We had to pick two hues and two forms (or shapes) to use during our experiments. We chose a circle and a square for our two forms. These shapes have been shown to be preattentive during numerous target and boundary identification experiments. Our hues were chosen from the Munsell colour space. Because Munsell is a perceptually balanced colour space, it can be used to choose hues that are isoluminant. This is necessary, because intensity itself is a preattentive feature, and therefore must be equal for both hues. The exact hues we used were a red (Munsell 5R 7/8) and a blue (Munsell 5PB 7/8). Previous experiments ensured that the perceived difference between these two hues was large enough to be preattentively detected. Healey et al. [1993] describes how this was done.

The experiment was split into two subsections B_1 and B_2 of 200 trials each. The first subsection tested a subject's ability to detect a horizontal boundary defined by hue (*i.e.,* red and blue). The second subsection tested a subject's ability to detect a horizontal boundary defined by form (*i.e.,* circle and square). Each trial was meant to simulate searching for a horizontal boundary while visualizing real-time data. A trial consisted of 18 separate data frames displayed to the subject one after another. Each frame contained 196 elements, and was shown for a fixed amount of time (between 50 and 150 msec) that was chosen before the trial started. After viewing a trial, users were asked to indicate whether a frame containing a horizontal boundary had been present or absent. For "boundary present" trials, one of the 18 data frames was randomly chosen to contain a horizontal boundary. The remaining frames displayed a random pattern of features (with no horizontal boundary present). In "boundary absent" trials, all 18 frames displayed a random pattern of features; no frame contained a horizontal boundary.

Trials in each subsection were divided equally between control trials, where a secondary feature was fixed to a specific constant value, and experimental trials, where a secondary

feature varied randomly from element to element. This allowed us to test for feature interference. Better performance in control trials versus experimental trials would suggest that using a secondary feature to encode an "irrelevant" data value interfered with a subject's boundary detection ability. We tested for both form interfering with hue boundary detection and hue interfering with form boundary detection. This experiment design gave us the following six subsections:

1. *hue-circle control*, horizontal boundary defined by hue, all elements are circles (Figures 6.1a – 6.1b).

2. *hue-square control*, horizontal boundary defined by hue, all elements are squares (Figures 6.1c – 6.1d).

3. *hue-form experimental*, horizontal boundary defined by hue, half the elements are randomly chosen to be circles, half to be squares (Figures 6.1e – 6.1f).

4. *form-red control*, horizontal boundary defined by form, all elements are red (Figures 6.2a – 6.2b).

5. *form-blue control*, horizontal boundary defined by form, all elements are blue (Figures 6.2c – 6.2d).

6. *form-hue experimental*, horizontal boundary defined by form, half the elements are randomly chosen to be red, half to be blue (Figures 6.2e – 6.2f).

Six subjects (five males and one female, aged 21 to 33) with normal or corrected acuity and normal colour vision volunteered to be tested. The experiments were conducted in the Computer Science Department's computer graphics laboratory, using a Silicon Graphics workstation equipped with a 21-inch colour display. The software used to conduct the experiments was written specifically to investigate preattentive visualization techniques. It used

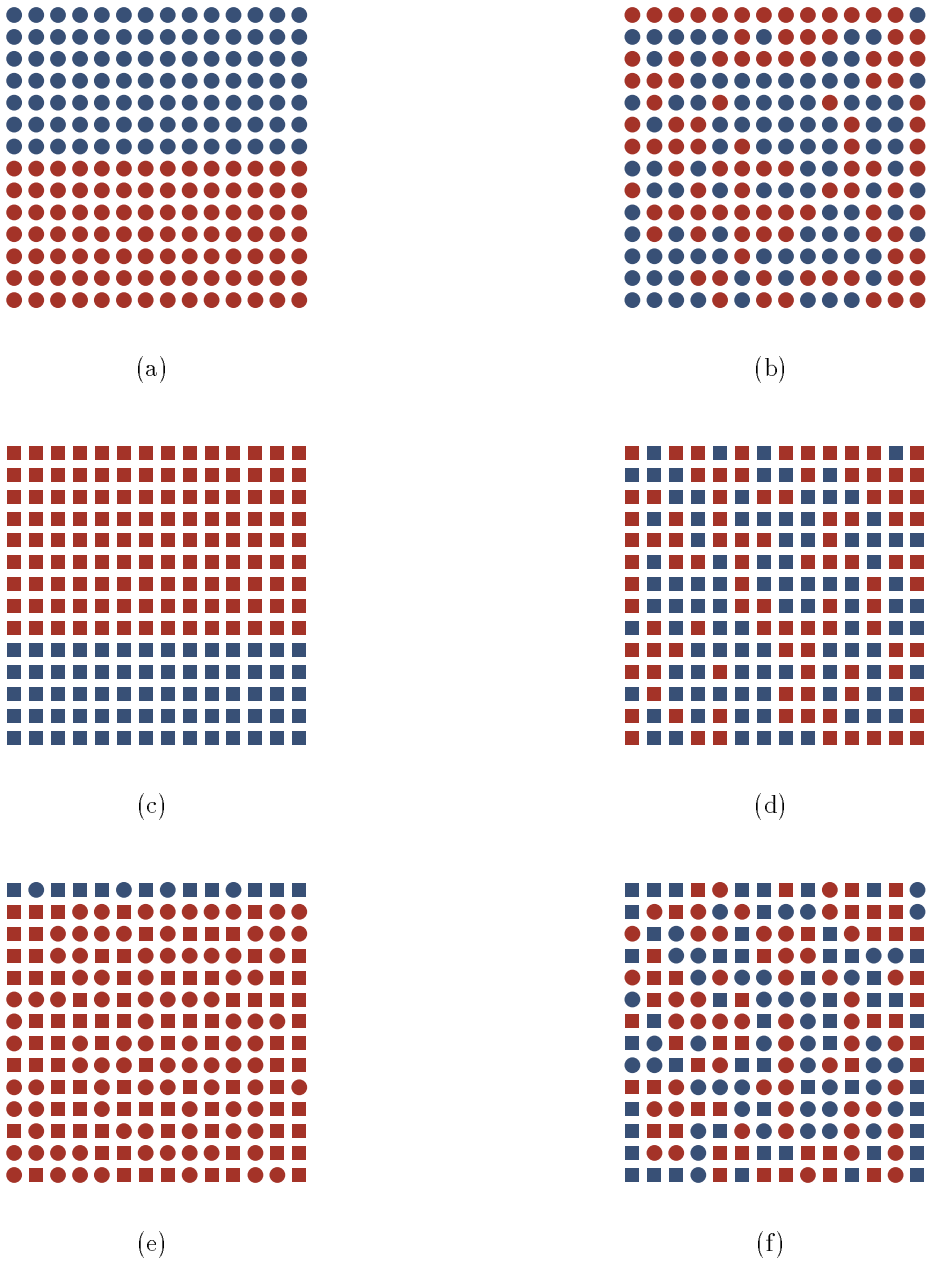


Figure 6.1: Example data frames from subsection B_1 of the boundary detection experiment (hue boundary detection): (a) control trial with all circles, horizontal boundary present; (b) control trial with all circles, boundary absent; (c) control trial with all squares, horizontal boundary present; (d) control trial with all squares, boundary absent; (e) experimental trial with random form, horizontal boundary present; (f) experimental trial with random form, boundary absent

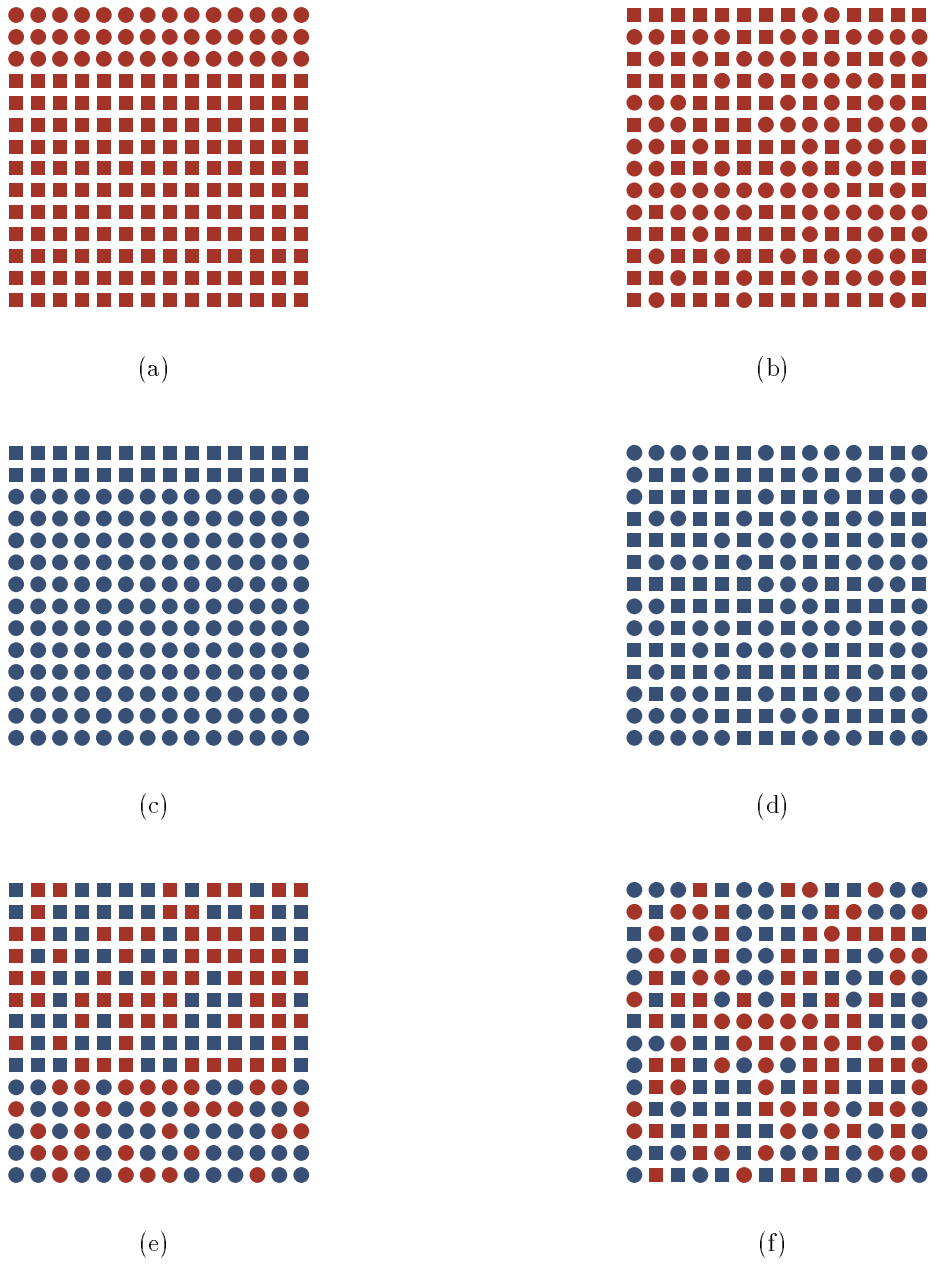


Figure 6.2: Example data frames from subsection B_2 of the boundary detection experiment (form boundary detection): (a) control trial with all red, horizontal boundary present; (b) control trial with all red, boundary absent; (c) control trial with all blue, horizontal boundary present; (d) control trial with all blue, boundary absent; (e) experimental trial with random hue, horizontal boundary present; (f) experimental trial with random hue, boundary absent

the display's vertical refresh to ensure accurate millisecond timing. Each subject completed both subsections of the experiment with three different frame exposure durations: 50 msec, 100 msec, and 150 msec.

At the beginning of the experiment, subjects were shown a sample display frame. The experiment procedure and task were explained. Subjects were also shown how to enter their answers (either "present" or "absent") using the keyboard. Subjects began both subsections of the experiment with a set of practice trials. This consisted of 40 trials, 20 control trials split evenly between the two types of controls (*i.e.,* ten trials with all circles and ten trials with all squares for subsection B_1, ten trials with all red and ten trials with all blue for subsection B_2) and 20 experimental trials. Ten control trials and ten experimental trials contained a horizontal boundary; the remaining trials did not. Exposure duration for practice trials was 100 msec per frame. The practice trials were designed to give the subjects an idea of the speed of the trials and the experiment. Trials were displayed one after another, and subjects were asked whether a horizontal boundary had been present or absent after each trial. If a subject responded correctly, a plus sign was shown following the response. If a subject responded incorrectly, a minus sign was shown. Feedback (plus or minus) was displayed in the center of the screen for 400 msec, at a size of approximately twice that of a single data element (1.2 cm or subtending a visual angle of 1.1° at 60 cm).

Next, subjects completed the two experiment subsections B_1 and B_2. Each subsection consisted of 100 control trials and 100 experimental trials. Fifty control trials and 50 experimental trials contained a horizontal boundary; the remaining trials did not. The 200 trials from each subsection were presented to the subjects in a random order. Subjects were provided with an opportunity to rest after every 50 trials. Feedback (plus or minus) was displayed after every subject response. Subjects completed both subsections three times using three different exposure durations: 50 msec per frame, 100 msec per frame, and 150 msec per frame. Frames with a given exposure duration were presented to subjects as a separate

group (*e.g.,* a subject completed all the 100 msec frames, followed by the 50 msec frames, then finished with the 150 msec frames). The ordering of the three exposure duration groups was random for each subject.

Results

The primary dependent variable examined was percentage error. Error was zero for trials where subjects responded correctly, and one for trials where they responded incorrectly. We began our analysis by dividing trials across the following experimental conditions, averaging response errors for each subject, then computing a mixed-factors ANOVA on the results:

- feature type; *hue* if a difference in hue defined the horizontal boundary, *form* if a difference in form defined the horizontal boundary
- trial type; *control* if the secondary feature was fixed to a constant value, *experimental* if it varied randomly from element to element
- block; BK_1 if a trial came from the first 100 trials the subject completed, BK_2 if it came from the last 100 trials
- exposure; *50*, *100*, or *150* msec, depending on a trial's display duration
- location of boundary frame (for experimental trials that contained a boundary frame); *front* if the boundary frame appeared during the first nine frames shown the the subject, *back* if it appeared during the last nine frames

Main effects with a p-value of less than 5% were considered significant. Results from the ANOVA suggested the following conclusions:

- rapid and accurate boundary detection can be performed using either hue or form; errors increased when exposure duration fell below 100 msec for both hue and form boundary detection
- form did not interfere with a subject's ability to detect a hue boundary at either 150 or 100 msec
- hue interfered with a subject's ability to detect a form boundary at both 150 and 100 msec
- accuracy was greater for hue than form, with this difference growing as exposure duration decreased
- there was no preference for the frame location of the target during either hue or form boundary detection

Figure 6.3 shows combined subject data for subsections B_1 (hue boundary detection) and B_2 (form boundary detection). The results indicate that hue boundary detection was quite accurate at all three exposure durations, with the most errors (about 13%) occurring in the experimental trials at 50 msec. Although errors for form boundary detection were uniformly higher than for hue boundary detection, subjects were still quite accurate at 100 msec (approximately 9%). Past that point, error rapidly approaches the chance limit of 50%, with an error rate of 36% at 50 msec.

Errors were generally higher for the form task than for the hue task, with $F(1, 10) = 46.51$, $p = 0.001$. The feature type by exposure duration interaction of $F(2, 20) = 45.54$, $p = 0.001$ was also significant. Additional F-values were computed to see how error varied across feature type (*i.e.,* hue and form) during the three exposure durations. In two of the three individual comparisons, hue accuracy was significantly greater than form accuracy (p-values ranged from 0.02 to 0.001). The one exception concerned 150 msec trials, where

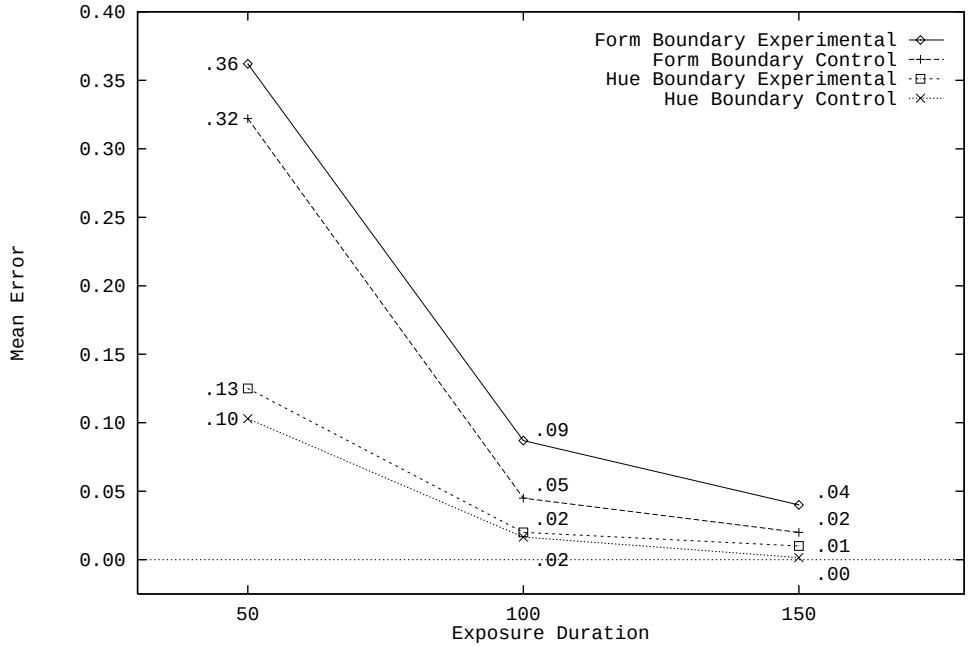


Figure 6.3: Graph of proportional error as a function of exposure duration for hue and form boundary trials; numbers represent exact proportional error values for each data point

$F(1,5) = 2.04$, $p = 0.19$. Differences in accuracy increased as exposure duration decreased, suggesting that the perceived difference between our two hues was larger than the perceived difference between a circle and a square.

ANOVA results of $F(1,10) = 16.34$, $p = 0.02$ showed a significant difference in errors between control and experimental trials. A feature type by trial type interaction of $F(1,10) = 3.56$, $p = 0.09$ suggested interference was present during both hue and form boundary detection. Moreover, a trial type by exposure duration interaction of $F(1,10) = 1.10$, $p = 0.35$ indicated interference at all three exposure durations. Simple t-tests comparing control and experimental trials across exposure duration showed weak interference (at a significance level of 10%) during form boundary detection for 100 and 150 msec trials. Thus, the hue-on-form interference effect must be considered small, albeit consistent. Corresponding results for hue

boundary detection found weak interference (at a significance level of 10%) for 50 msec trials. This is similar to Callaghan's [1989, 1990] static boundary detection experiments, although weak hue interference during form boundary detection was not reported in her results.

There was a significant exposure duration effect, $F(1, 10) = 197.66$, $p = 0.001$. Individual F-values for the four conditions shown in Figure 6.3 (form boundary experimental, form boundary control, hue boundary experimental, and hue boundary control) were all $p = 0.001$. Fisher's protected least significant difference (PLSD) values identified significant differences between exposure durations $(50, 100)$ and $(50, 150)$ in all four conditions, but not between $(100, 150)$. We concluded that the high F-values were due to relatively higher errors during the 50 msec trials.

Finally, the results showed no boundary frame location preference for either hue ($F(1, 5) = 2.37$, $p = 0.18$) or form ($F(1, 5) = 0.05$, $p = 0.83$) boundary detection. Moreover, there was no consistent effect of trial block. Whereas errors actually increased from BK_1 to BK_2 in the hue condition, $F(1, 5) = 17.44$, $p = 0.01$, they decreased (non-significantly) over time in the form condition, $F(1, 5) = 5.51$, $p = 0.07$. There were no other significant interactions of the block factor with other factors of interest. We can draw no conclusions about the effects of practice or fatigue without performing additional experiments.

6.2 Experiment 2: Target Detection

We continued our investigation of real-time preattentive visualization by studying temporal target detection. Our second experiment addressed two additional questions about preattentive features and their use in our visualization tools.

- *Question 1:* Is it possible for subjects to detect a data frame containing a unique target element in a sequence of random frames? If so, what features allow this and under what conditions?

- *Question 2:* Does any interference occur when viewing a dynamic sequence of data frames? Specifically, does random hue interfere with form target detection? Does random form interfere with hue target detection?

As with temporal boundary detection, these questions are specifically designed to address our visualization requirements. The ability to perform target detection using preattentive features would provide further justification for their use in exploring large datasets or visualization results from real-time applications. Our experiments also searched for any new types of interference that might occur as a result of viewing a dynamic sequence of data frames during visualization.

Method

We chose to test the same two visual features (hue and form) used during the boundary detection experiments. Target detection experiments consisted of frames containing 125 elements (Figures 6.4 and 6.5). The position of the elements was held constant in every frame. Hue and form were the same as in the boundary detection experiments, specifically a red (Munsell 5R 7/8) and a blue (Munsell 5PB 7/8) hue, a circle and a square form.

As in the first experiment, temporal target detection was split into two subsections T_1 and T_2 of 200 trials each. The first subsection tested a subject's ability to detect a target element defined by hue. The second subsection tested a subject's ability to detect a target element defined by form. Each trial was meant to simulate searching for a target element while visualizing real-time data. A trial consisted of 18 separate data frames, which were

displayed to the subject one after another. Each frame was shown for a fixed amount of time (either 50 or 100 msec) that was chosen before the trial started. After viewing a trial, users were asked to indicate whether a frame containing the target element had been present or absent. For "target present" trials, one of the 18 data frames was randomly chosen to contain the target element. The remaining frames did not contain a target element. In "target absent" trials, none of the 18 frames contained a target element.

As with boundary detection, we tested for feature interference by dividing each subsection into control and experimental trials. In control trials, the secondary feature was fixed to a specific constant value; in experimental trials, it varied randomly from element to element. We tested for both form interfering with hue target detection, and hue interfering with form target detection. This gave us the following six subsections:

1. *hue-circle control*, target element is a red circle, all distractors are blue circles (Figures 6.4a – 6.4b).

2. *hue-square control*, target element is a red square, all distractors are blue squares (Figures 6.4c – 6.4d).

3. *hue-form experimental*, target element is a red circle, half the distractors are randomly chosen to be blue circles, half to be blue squares (Figures 6.4e – 6.4f).

4. *form-red control*, target element is a red circle, all distractors are red squares (Figures 6.5a – 6.5b).

5. *form-blue control*, target element is a blue circle, all distractors are blue squares (Figures 6.5c – 6.5d).

6. *form-hue experimental*, target element is a red circle, half the distractors are randomly chosen to be red squares, half to be blue squares (Figures 6.5e – 6.5f).

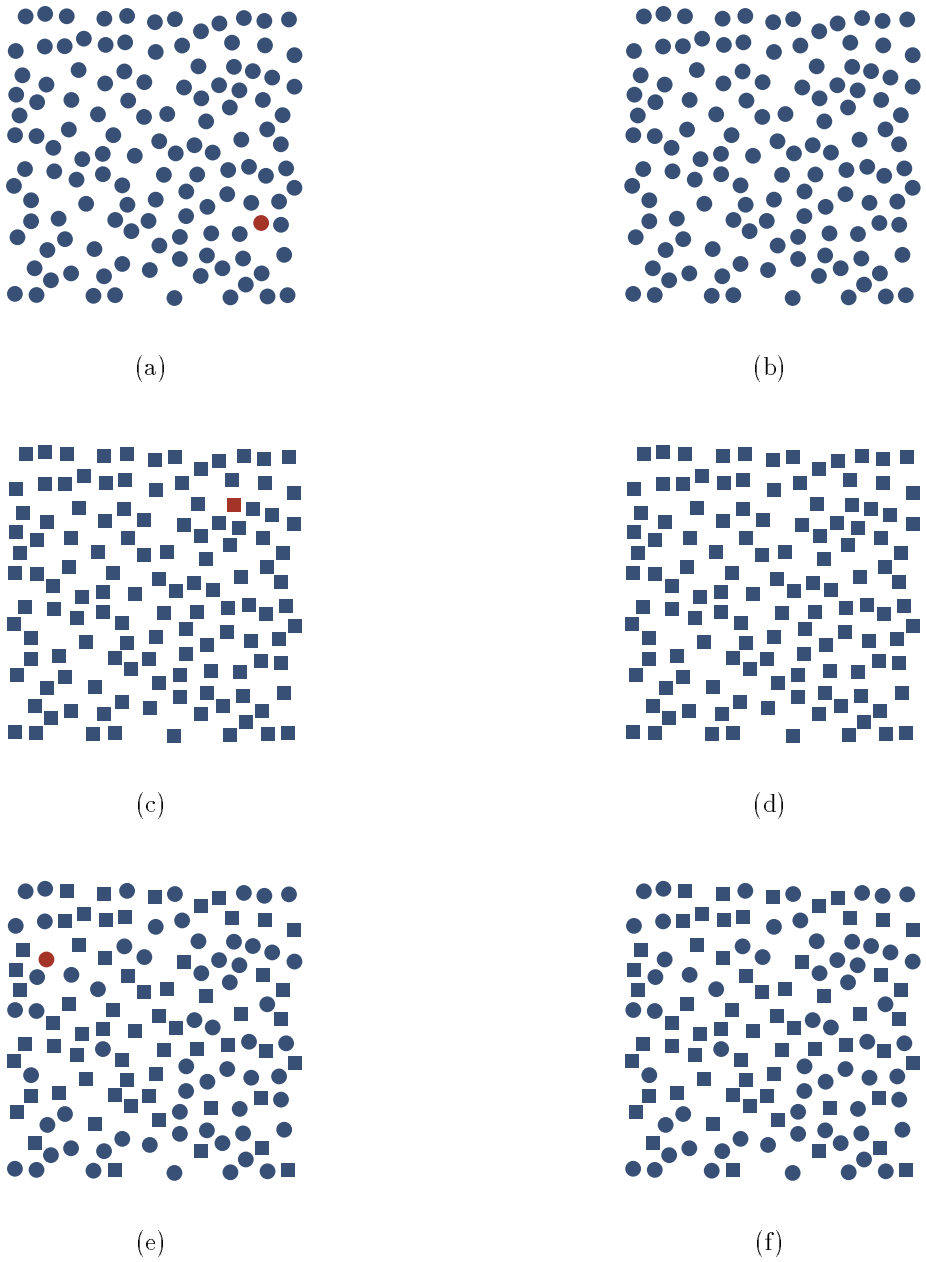


Figure 6.4: Example data frames from subsection T_1 of the target detection experiment (hue target detection): (a) control trial with all circles, red target present; (b) control trial with all circles, target absent; (c) control trial with all squares, red target present; (d) control trial with all squares, target absent; (e) experimental trial with random form, red target present; (f) experimental trial with random form, target absent

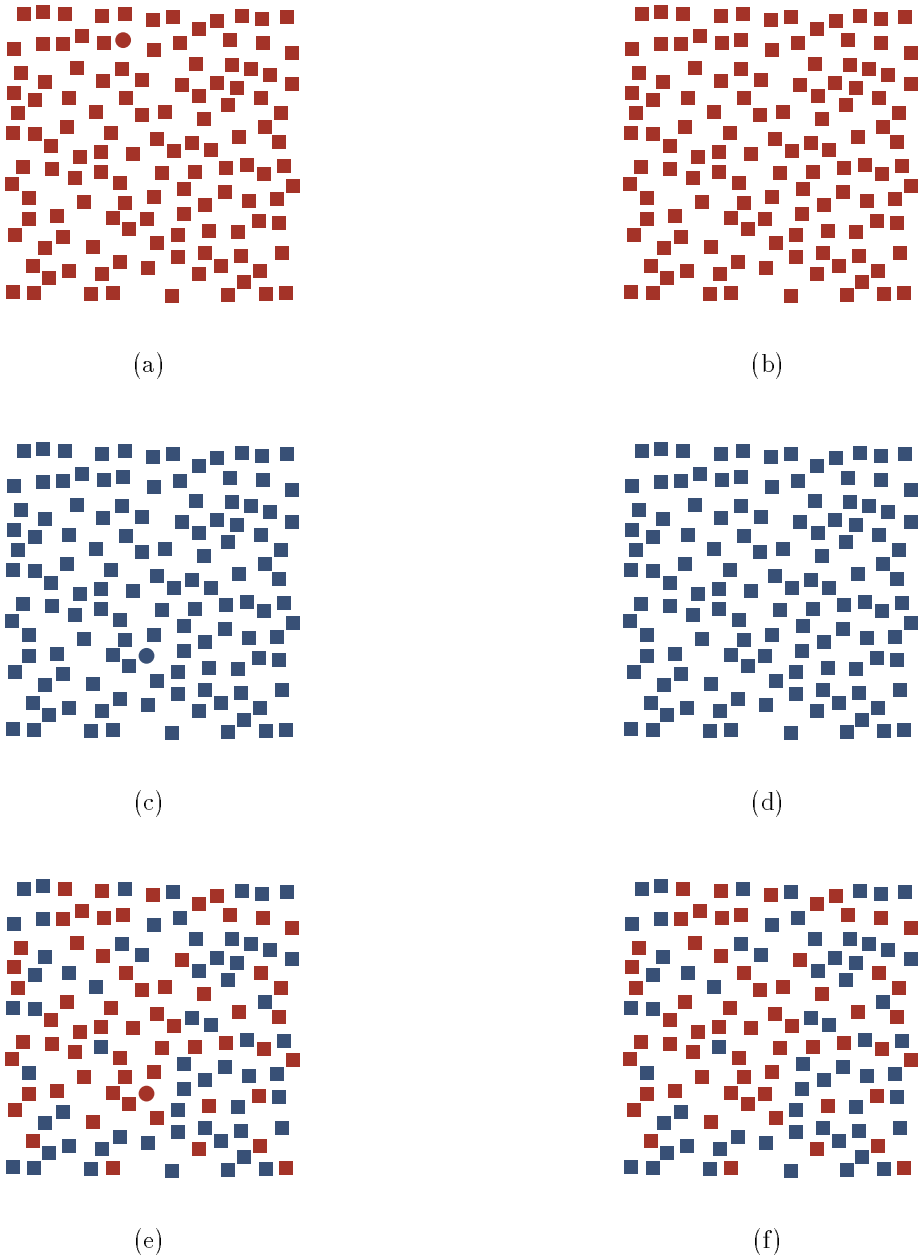


Figure 6.5: Example data frames from subsection T_2 of the target detection experiment (form target detection): (a) control trial with all red, circle target present; (b) control trial with all red, target absent; (c) control trial with all blue, circle target present; (d) control trial with all blue, target absent; (e) experimental trial with random hue, circle target present; (f) experimental trial with random hue, target absent

Six subjects (five males and one female, aged 21 to 33) with normal or corrected acuity and normal colour vision were tested. Five of the six subjects also participated in Experiment 1. At the beginning of the experiment, subjects were shown a sample display frame. The experiment procedure and task were explained. Subjects were shown how to enter their answers (either present or absent) using the keyboard. Subjects began both subsections of the experiment with a set of practice trials similar to those for the boundary detection experiments. Exposure duration for practice trials was 100 msec per frame. Trials were displayed one after another. Subjects were asked whether a target element had been present or absent for each trial. Correct or incorrect responses were signalled by a plus or a minus sign.

Next, subjects completed the two experiment subsections T_1 and T_2. Each subsection consisted of 100 control trials and 100 experimental trials. Fifty control trials and 50 experimental trials contained a target element; the remaining trials did not. The 200 trials from each subsection were presented to the subjects in a random order. Subjects were provided with an opportunity to rest after every 50 trials. Feedback (plus or minus) was displayed after every response. Subjects completed both subsections two times using two different exposure durations: 100 msec per frame and 50 msec per frame. The ordering of the two exposure duration groups was random for each subject.

Results

The primary dependent variable was again percentage error. A mixed-factors ANOVA was computed across the same conditions used for analysing boundary detection. The only difference was the number of possible values for exposure duration: *50* or *100* msec, depending on the trial's display duration. Results from the ANOVA can be summarized as follows:

- rapid and accurate target detection could be performed using hue at both 50 and 100 msec exposures
- similar rapid and accurate target detection based on form was possible only when hue was held constant
- form variations did not interfere with the ability to detect a hue-defined target
- hue variations did interfere with the ability to detect a form-defined target
- there was no preference for the frame location of the target during either hue or form target detection

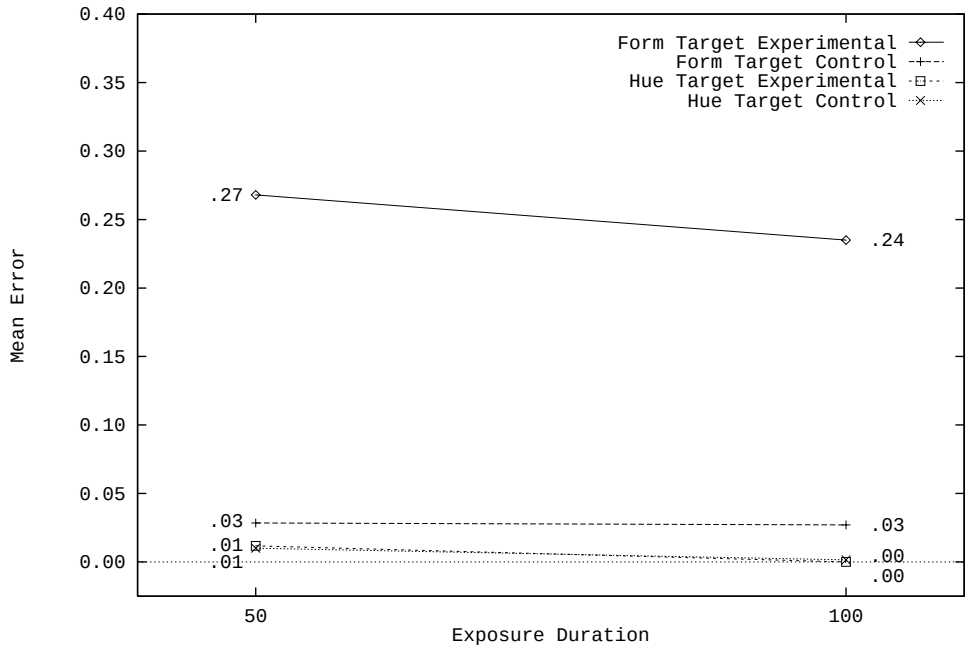


Figure 6.6: Graph of mean error as a function of exposure duration for hue and form target trials; numbers represent exact proportional error values for each data point

Figure 6.6 graphs combined subject data for subsections T_1 (hue target detection) and T_2 (form target detection). The results show that hue target detection is very accurate

at both exposure durations for control and experimental trials (a maximum of 1% error). In contrast, form target detection was very accurate for control trials (3% error), but not for experimental trials (24% error at 100 msec and 27% error at 50 msec). There were no significant effects of exposure duration in any of the four conditions.

A feature type by trial type interaction of $F(1,10) = 62.52$, $p = 0.001$ suggests interference during one of the two target detection tasks. The difference in errors between control and experimental trials was significant in the form task, $F(1,5) = 103.58$, $p = 0.001$, but not in the hue task, $F(1,5) = 1.62$, $p = 0.26$. There was no feature type by trial type by exposure duration interaction, $F(1,10) = 0.39$, $p = 0.55$. Thus, as with our boundary detection task and Callaghan's [1989, 1990] static displays, random hue interferes with form target detection at both exposure durations, while random form does not interfere with hue target detection at either exposure duration. This provides further evidence for concluding that the perceived difference that can be obtained by using two hues is larger than the difference obtained from a circle and a square.

Figure 6.6 shows that errors were generally higher for the form task than for the hue task. ANOVA results supported this, with $F(1,10) = 46.51$, $p = 0.001$. There was no feature type by exposure duration interaction, $F(1,10) = 0.10$, $p = 0.76$. This means hue target detection was easier than form target detection at both exposure durations. Combined with hue interference during form target detection, this suggests that hue should be used as a primary feature when searching multidimensional data elements for a specific target.

As with the boundary detection task, there were no frame location effects for either hue, $F(1,5) = 1.26$, $p = 0.31$, or form, $F(1,5) = 1.04$, $p = 0.35$. The effects of trial block were again mixed. Errors were lower in BK_1 ($\overline{x} = 0.06$) than in BK_2 ($\overline{x} = 0.14$), $F(1,5) = 56.81$, $p = 0.001$ for form targets, but did not differ significantly for hue targets, $F(1,5) = 0.79$, $p = 0.41$. There were no significant interactions of trial block with other

factors of interest.

Chapter 7

Dataset Management

This chapter reviews a number of techniques used to offer different levels of scientific dataset management. Early systems implemented simple selection, filtering, and type conversion operations. Researchers are now combining commercial relational database and visualization tools to build hybrid visualization systems. The advantage of such a system is direct access to the functionality provided by the underlying database. Unfortunately, the relational database model is not well-suited to handling errors, noise, and missing values that often occur in scientific datasets. We review a promising new area of database research, knowledge discovery, and discuss how this technique might be used to focus and compress a dataset before it is visualized. The next chapter (Knowledge Discovery in Visualization) describes how we implemented and extended four known knowledge discovery algorithms. These algorithms were integrated into our visualization environment and then applied to a number of practical applications to test their effectiveness on real-world data and tasks.

Underlying every visualization tool or technique is the dataset to be visualized. Recent developments in computing have created several high volume data sources. Examples include supercomputers, satellites, geophysical monitoring stations, and medical scanners. Much of this data is stored without ever being fully used, due in part to the amount of time required to apply traditional analysis techniques. Many scientists now believe efficient access to and

management of these datasets is an important area of research in scientific visualization. The ability to select, combine, and filter data before it arrives at the visualization tool could significantly reduce the amount of information that needs to be displayed. More advanced techniques would allow users to classify, detect trends and relationships, and build rules that describe dependencies among attributes. Use of this "discovered" information can improve accuracy and dramatically reduce the size of the dataset being visualized.

Various panels and workgroups studying visualization have addressed the data management problem. Lloyd Treinish chaired a panel at ACM SIGGRAPH in 1989, devoted specifically to data management in scientific visualization [Treinish et al., 1989]. The need for efficient data management tools was exemplified by the NASA digital data archive. At that time, the database was approximately six terabytes in size, and was doubling roughly every two years. Treinish described a database where every underlying dataset used the same data format (*e.g.,* Common Data Format or Hierarchical Data Format). Filters that performed tasks such as scaling, projection, discretization, and data conversion sat between the database and various visualization tools. These techniques were implemented in a multidimensional visualization tool used to display NASA satellite and earth science data [Treinish and Goettsche, 1991]. Foley and Campbell discussed the design of data models that efficiently connect visualization tools to their underlying datasets. This was followed by the Intelligent Data Management (IDM) project, designed for NASA's National Space Science Data Centre [Cambell et al., 1989]. IDM was an attempt to develop a generalized set of data management tools that could be used by scientists from any of the space or earth science disciplines supported at NASA. The system uses techniques such as natural language query, expert systems, and inference reasoning to provide users with a simple interface into large multidimensional datasets.

Discussions conducted by Wolff with a number of visualization experts yielded a similar focus on the data management problem [Brown et al., 1988; Wolfe and Franzel, 1988]. Panel

members noted that visualization arose from the need to analyse an ever-increasing amount of data. Graphical displays are one method for improving our understanding of the data. Unfortunately, even the most efficient visualization tool cannot necessarily display large or high-dimensional datasets quickly or effectively. Some form of management through filtering or data compression is needed if users hope to view even a small portion of their databases.

A recent update on the NSF visualization project described current and ongoing research being performed at a number of academic institutions [Rosenblum, 1994]. Although many visual presentation techniques have been studied (*e.g.,* volume visualization, fluid flow, and perceptual visualization), much less work has focused on improving data management. The problems cited by the original panel still persist. Datasets continue to grow in size, and with this our inability to analyse even a small fraction of their contents.

7.1 Database Management Systems

Some researchers have addressed the data management issue in scientific visualization. An obvious approach would be to use a combination of commercial database management systems (DBMS) and visualization packages. However, initial work focused on data management techniques that were independent of the available visualization or database management tools. This was due in large part to the limitations inherent in these systems. An examination of popular visualization tools such as AVS, apE, VIS-5D, and Explorer reveals that these packages provide relatively few data management tools [Upson, 1989; Hibbard and Santek, 1990; Vande Wettering, 1990]. Normally, these systems read data directly from already-existing files. Manipulation of data within the system is limited to operations such as applying regular grids, selection, clipping, and type conversion. Moreover, these operations support only a fixed set of built-in data types. Like visualization tools, DBMSs were

restricted to a fixed set of data types and basic operations. Moreover, relational DBMSs are not well suited to handling scientific datasets. Problems such as size, complexity, and the inability to match the conceptual framework of the dataset to a relational model (*e.g.,* spatial datasets) are difficult to overcome.

Treinish has developed a flexible multidimensional visualization system [Treinish and Goettsche, 1991] designed to help scientists analyse the data they produce. This goal led to a number of system requirements. In particular, Treinish felt that a discipline-independent system should handle arbitrary types of data. In order to meet these requirements, Treinish proposed a construct called the visualization pipeline designed as follows:

- Raw data is stored in a data repository in some abstract, data-independent format (Treinish suggests using the Common Data Format). Users can select the data they want to visualize from the repository.
- The selected data can be passed through a number of filters, in order to convert it into the desired format. These may include scaling and projecting the data, as well as conversion to a specific format or coordinate system.
- If the data is continuous, it may need to be sampled or gridded to a user-selected resolution.
- Finally, the data is visualized or rendered using one of a number of visualization tools provided with the system (*e.g.,* two and three-dimensional plots, contour plots, surfaces, or flow fields).

Treinish stressed the importance of the data repository and the supporting data management facilities (*i.e.,* simple data selection queries, followed by filtering operations like scaling, rotation, projection onto a 2D plane, and gridding). He felt this was key to allowing

discipline-independent visualization. Data selection and manipulation is separated from data visualization. This gives the user the flexibility to work with arbitrary datasets and a variety of visualization techniques. Treinish continued his discussion of data management issues by listing a set of requirements and goals for a generalized data management system that supports scientific visualization [Treinish, 1993]. Included in his discussion were examples of data types, dimensionality, rank, and spatial arrangement, as well as a review of which requirements are addressed or solved by popular visualization packages.

Commercial visualization tools and DBMSs are much more prominent in recent work on data management in visualization. In large part, this is because both are now designed to be extensible. For example, users can write new visualization "functions" using traditional programming languages and integrate them seamlessly into many of the current visualization packages. The same is true for DBMSs like Postgres, Oracle, IRIS, and Orion. These systems support enhanced data models that allow users to define new data types, as well as operations and functions to support them. Recent work has concentrated on writing functions to integrate visualization packages with popular database management programs. This makes the functionality of the DBMS available from within the visualization system.

Kochevar and Ahmed describe a system that allows users to locate and visualize data using a graphical browser built on top of Postgres database files [Kochevar et al., 1993]. Postgres was chosen because it supports the following enhanced database features:

- user-defined data types and user-defined functions that perform operations on them
- "large objects", which represent a collection of bytes whose structure is unknown to Postgres; user-defined functions are required to perform operations on large objects
- an "inverted" file system within Postgres to manage large objects

Datasets are stored as large objects inside Postgres. The key to this method is including

a description of each dataset within the dataset itself. This allows a dataset to describe its structure to other programs, in particular to external visualization tools. An application program called the Visualization Executive is used to browse through the collection of datasets, in a manner similar to using a file browser. When a user selects a specific dataset, a visualization package is spawned, the dataset's description is sent to the package, and the contents of the dataset are displayed. Scripts for each dataset specify the particular visualization tool to use, and the commands needed to start its execution. This information is also stored within the dataset.

The Visualization Executive allows users to view their datasets using any one of a number of popular visualization tools. Any package that supports scripting should be compatible with this system. The loose coupling that integrates Postgres and the visualization tools is possible because of the user-definable data types and functions available in Postgres. Any DBMS that provides similar functionality can be used to support this kind of visualization methodology.

Tioga is an example of a tool that maintains a much tighter connection between the visualization system and its underlying database [Stonebraker et al., 1993]. Tioga also uses Postgres, although any DBMSs that supports user-defined data types, user-defined functions, and multidimensional access methods (*e.g.,* R-trees or grid files) would be sufficient. Tioga provides an environment that allows users to build visualization programs. This is analogous to other popular visualization systems such as AVS and Explorer. Because Tioga is built directly on top of Postgres, however, all of the functionality of a commercial DBMS is available to Tioga users.

Tioga supports the traditional box and arrow method for building visualization systems. Users create their programs in this type of environment using one or more boxes connected together with arrows. Each box represents a data processing operation. Arrows describe how

the data flows from box to box within the visualization system. A box must have at least one input arrow to describe how and when it receives its data. Some boxes use output arrows to send results forward to the next processing step (*e.g.,* boxes that resample or transform data elements). A box is not required to output data, however (*e.g.,* boxes that display results on the screen).

Visualization systems built with Tioga are called recipes, and individual boxes within a recipe are called ingredients. An important property of Tioga is that any user-defined Postgres function is automatically an ingredient, available for use within a recipe. These Postgres-based ingredients can be conceptually divided into two groups: data management functions and browsers. Data management functions perform conventional operations such as selection, projection, and join on the underlying datasets.

Browsers are designed to display screen images when the visualization program runs. In order to support browsers, Tioga requires every visualization program to define an "application coordinate space". This is usually a simple $N-$dimensional coordinate system, where each dimension represents a data attribute that might be displayed on the screen. Database objects can then provide an associated geometry that falls within the application coordinate space. The geometry of a single tuple is a point in $N-$space. A database object with multiple tuples uses an $N-$dimensional polyhedron as its geometry. Users are required to provide a display function for every user-defined data type. A browser provides an object's location to the display function, which must then return a displayable version of the object's value. Browsers can move to any point in the application coordinate space. They also support the ability to travel through application coordinate space, retrieving and displaying information repeatedly until some user-defined condition is met. This was used to "fly" through satellite images of hurricane Hugo by travelling forwards and backwards along a time attribute [Stonebraker et al., 1993].

Both Kochevar and Ahmed [1993] and Stonebraker [1993] describe techniques that act like "glue" to connect existing DBMSs and visualization packages. The systems' data management abilities are still limited to the functionality provided by the DBMS. Since most DBMSs are relational in nature, they have the same problems as their predecessors when dealing with certain types of data. Spatial and temporal datasets do not fit well within the traditional relational approach. Data with noise, nondeterministic values, missing values, or errors are also difficult to store in a relational DBMS. Perhaps more importantly, even enhanced DBMSs still have the same set of built-in operations (*e.g.,* select, project, and join). New techniques to support time, statistical summaries, classification, and knowledge discovery are not included and are difficult to provide, even with the ability to design new data types and operations.

7.2 Knowledge Discovery

Knowledge discovery or data mining, as it is sometimes referred to, is a relatively new area of database research [Silbershatz et al., 1990; Frawley et al., 1991]. Knowledge discovery is defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data". This is done by combining a variety of database, statistical, and machine learning techniques. Although knowledge discovery uses research from all of these areas, it is not a simple subset of any one of them. For example, traditional statistical techniques are not well suited to the structured, incomplete, and dynamic datasets used in knowledge discovery. Moreover, the size of the datasets is often beyond the ability of statistics to analyze in an efficient manner. Because of this, a large gap is developing between the amount of data being generated and the amount being used or understood.

Knowledge discovery methods are expected to read and process raw data stored in a

database or data repository. The raw data has a number of inherent properties that must be addressed by any knowledge discovery algorithm. Data stored in a database is dynamic. It often has attributes that are irrelevant to the relationship being investigated. There is a measure of uncertainty associated with an attribute (*e.g.,* due to inaccuracies in measurement or recording of values). Tuples may contain missing entries, noise, or errors. Entire attributes may be missing from the database (*i.e.,* the given attributes do not differentiate the data uniquely). These properties often preclude the use of already-existing machine learning or statistical techniques.

Based on this description, knowledge discovery algorithms satisfying the following four conditions are sought:

1. A high-level language is used to represent discovered knowledge.
2. The discovered knowledge is interesting, nontrivial, and novel.
3. The discovered knowledge is reasonably accurate.
4. The knowledge discovery algorithms are efficient.

Because knowledge discovery uses techniques similar to those in statistics, machine learning, expert systems, and database management systems, it is useful to describe some of the differences between these disciplines.

- *statistics*

  is not well suited for structured, incomplete, or dynamic data; data driven, it cannot incorporate domain knowledge; results are difficult for non-experts to analyse; there is no guidance on which statistics to compute, and which results to analyse

- *database management systems*

  can extract useful, interesting, nontrivial results but are limited to the attributes contained in the database; they cannot determine which calculations to perform to obtain new information "buried" in the original datasets

- *expert systems*

  often require data of much higher quality and integrity; usually only important cases are covered

- *machine learning*

  data from a database is usually dynamic (*i.e.,* constantly changing or growing), incomplete, noisy, and large compared to machine learning datasets

Discovered knowledge comes in a variety of forms. Inter-field patterns show field dependencies within a single tuple. Inter-record patterns cluster tuples into groups with common characteristics. This can be used to summarize data, or to show trends over time. Techniques can also be divided into quantitative methods that relate numeric field values, and qualitative methods that find logical relationships in the dataset. Examples of tasks performed by different algorithms include clustering and pattern recognition, determining rules to identify group membership, summarizing, discrimination, and comparison.

Various forms of knowledge discovery are currently being used in a number of practical application environments. American Airlines uses knowledge discovery techniques to search for trends in their frequent flyer database. Banks search their loans database to try to improve methods for screening potential borrowers. General Motors has a large research group applying knowledge discovery methods to automobile trouble reports; the results are used to develop diagnostic expert systems for various car models. Food manufacturers use supermarket check-out scanner data to measure promotion effects and search for shopper patterns.

Current research in knowledge discovery is focusing on a number of different problems. Some techniques attempt to incorporate expert user or domain knowledge into discovery algorithms. Efficient algorithms are essential if they are to be applied to very large datasets. Incremental methods that update themselves as the dataset changes over time are also important. Researchers are studying the ability to integrate results from various techniques, and to allow user interaction during the discovery process.

7.3 Managing Large Multidimensional Datasets

We hypothesise that knowledge discovery techniques can be used to improve the efficiency of visualizing large multidimensional datasets. The use of knowledge discovery in scientific data management has been addressed briefly by Cambell and others [Cambell et al., 1989]. There are no descriptions of visualization tools that use knowledge discovery explicitly, although NASA's IDM project does use expert systems and inference reasoning to answer user queries about earth science datasets [Treinish and Goettsche, 1991]. We believe the advantages of knowledge discovery algorithms are twofold. First, they can be used to reduce the amount of data that needs to be displayed. Second, they can be used to "discover" previously unknown and potentially useful information. Some examples of how this can be done include:

- datasets can be filtered by identifying the subset of elements that participate in a particular relationship being investigated; irrelevant data elements can be ignored and do not need to be displayed
- attributes that are significant to a given relationship can be identified; only displaying significant attributes reduces the dimensionality of each data element

- data elements can be grouped or classified, and elements from specific groups can then be displayed, reducing the amount of data being sent to the visualization tool
- data elements can be grouped or classified, and only the classification value (and possibly a confidence measure) can be displayed, compressing the dataset into one or two dimensions
- significant dependencies or relationships between data elements or attributes can be identified and displayed
- temporal data can be compressed or filtered along the time axis, if a temporal data attribute exists in the dataset

Because our tools will be used for exploratory data analysis, we do not require techniques that are 100% accurate. User knowledge about the datasets being visualized can help to determine the accuracy or relevance of results being returned. Each technique is expected to provide confidence weights for the classifications, dependencies, and relationships it discovers. These can be used to decide how "sure" the algorithm is about the results it suggests. This might allow results from different algorithms to be combined in various ways.

Our focus is not on the design of new knowledge discovery algorithms. Instead, we implemented four existing techniques, then used them in a visualization environment to see if they offered improved efficiency or usefulness compared to visualization without any form of data management. We modified and extended the algorithms to provide a set of unified results required by our visualization tool, in particular, classification confidence weights, the ability to compare different possible classifications, and the ability to identify attributes that are significant to a specific classification (these extensions are explained in detail in the next chapter).

S	the number of tuples in the dataset to be classified
N	the number of attributes in each tuple to be classified
$\{A_1, \ldots, A_N\}$	the N individual attributes in each tuple to be classified
n_i	the number of unique values for attribute A_i
$\{a_{i,1}, \ldots, a_{i,n_i}\}$	the n_i unique values for attribute A_i
P	the number of possible classification values
$\{c_1, \ldots, c_P\}$	the P individual classification values

Table 7.1: An explanation of the common terms used to describe the four knowledge discovery algorithms

The four knowledge discovery algorithms we used are described below. Two of the techniques, Quinlan's ID3 approach and Agrawal's interval classifier, use decision trees to represent classification rules. The other two methods, Chan's statistical tables and Ziarko's rough set algorithm, build classification rules using mathematical and statistical analysis of an initial training dataset. These four techniques offer a good overview of the types of methods being used for knowledge discovery. Table 7.1 identifies the common terms used to describe each knowledge discovery algorithm.

7.4 Decision Trees

One of the first knowledge discovery techniques applicable in a database environment was a method proposed by Quinlan called ID3 [Quinlan, 1986]. ID3 builds a decision tree from an initial training set where each tuple in the set has been previously classified into one of P possible values. The decision tree is then used to assign a classification value c_k to unclassified tuples. ID3 was specifically designed to handle noisy and erroneous data. This means it can be used for knowledge discovery in a database environment.

The algorithm begins by selecting significant attributes from the training set. Assume

that the training set has p_1 tuples with classification value c_1, p_2 tuples with classification value c_2, and so on up to p_P tuples with classification value c_P. Moreover, any attribute A_i can be used to subdivide the training set into n_i subsets $\{S_1, \ldots, S_{n_i}\}$. Each subset S_i has $p_{i,1}$ tuples with classification value c_1, $p_{i,2}$ tuples with classification value c_2, and so on. The expected value $\overline{p}_{i,j}$ of $p_{i,j}$ is proportional to the number of tuples with classification value c_j in the original training set, namely

$$\overline{p}_{i,j} = \left(\frac{p_j}{p_1 + p_2 + \cdots + p_P} \right) size_i \tag{7.1}$$

where $size_i = \sum_{j=1}^{P} p_{i,j}$ represents the number of tuples in subset S_i. The expected values can be used to perform a chi-squared independence test on attribute A_i, by computing

$$\chi^2_{n_i - 1} = \sum_{i=1}^{n_i} \sum_{j=1}^{P} \frac{(p_{i,j} - \overline{p}_{i,j})^2}{\overline{p}_{i,j}} \tag{7.2}$$

The chi-squared value can be used to determine whether attribute A_i can be rejected as independent of the values chosen for classification. Usually, only attributes with a high confidence level (*e.g.,* 99.9%) are marked as significant. These attributes are then used to build a decision tree.

The attribute that provides the largest information gain is used to partition the root of the decision tree. The tree is simply a data structure that returns a classification value c_k for an unknown tuple. This is done by traversing the tree from root to leaf and returning the classification value stored at the leaf. Attribute values in the unclassified tuple are used to direct the path at each node in the tree (this is described in a more detailed example below). The expected information needed to generate this result is:

$$I(P) = \sum_{i=1}^{P} -\frac{p_i}{total} log_2 \frac{p_i}{total} \tag{7.3}$$

where $total = \sum_{i=1}^{P} p_i$. If A_i is chosen to be used at the root of the decision tree, the training set is subdivided into n_i subsets $\{S_1, \ldots, S_{n_i}\}$. The expected information required for each subtree S_i is $I(S_i)$. The expected information required for the decision tree with A_i as root is the weighted average

$$E(A_i) = \sum_{i=1}^{n_i} \frac{size_i}{total} I(S_i) \tag{7.4}$$

Because information gain is defined as $gain(A_i) = I(P) - E(A_i)$ and $I(P)$ is constant, maximizing gain is the same as minimizing expected information. The attribute with the minimum expected information is used at the root of the decision tree.

Subtrees are processed in a recursive manner. The expected information for the remaining attributes in each S_i is computed. This is used to choose an attribute for the root of the subtree. Subdivision continues until one of two stopping conditions is reached:

1. Every tuple in the given subset has the same classification value.
2. The tuples in the given subset are made up of only one attribute value.

Each node in the decision tree contains a subset of the tuples in the original training set, and each tuple has a classification value attached to it. Within any given node the classification value with the maximum frequency is marked as the "winning" classification value. This is used to handle noise and uncertainty in the training set. Suppose that the training set was reduced to a subset of tuples with only one attribute value (stopping condition 2 above),

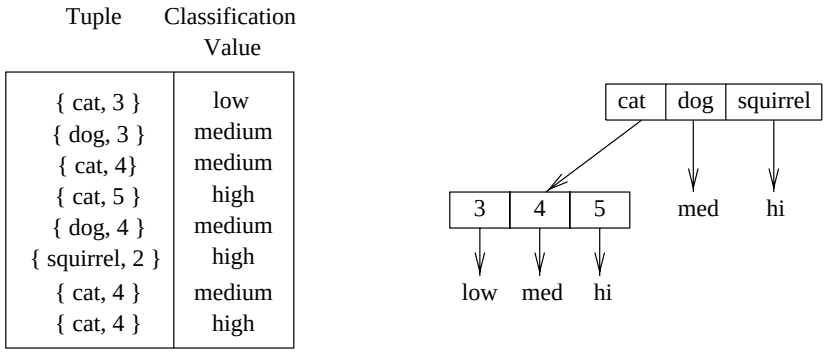


Figure 7.1: An example of a training set with eight tuples and the corresponding decision tree; in this example $N = 2$, $\{A_1, A_2\} = \{$ animal, age $\}$, $n_1 = 3$, $n_2 = 4$, $\{a_{1,1}, a_{1,2}, a_{1,3}\} = \{$ cat, dog, squirrel $\}$, $\{a_{2,1}, a_{2,2}, a_{2,3}, a_{2,4}\} = \{$ 2, 3, 4, 5 $\}$, $P = 3$, $\{c_1, c_2, c_3\} = \{$ low, medium, high $\}$

but the tuples had more than one classification value. There are only two possible situations where this could happen: either the original dataset was underdefined by its attributes (*i.e.,* two tuples with the same attribute values were assigned two different classification values), or the original dataset contained errors. Although other methods exist for choosing a winning classification value (*e.g.,* assigning a probability to each possible value, generating a random number between 0 and 1, and using this to choose a classification), Quinlan found that choosing the value with the maximum frequency minimized expected classification error.

As an example, consider the training set and corresponding decision tree shown in Figure 7.1. The root of the tree was partitioned using the attribute $A_1 =$ `animal`. This resulted in three subsets of tuples: those with an animal value of $a_{1,1} =$ `cat`, those with an animal value of $a_{1,2} =$ `dog`, and those with an animal value of $a_{1,3} =$ `squirrel`. Tuples in the `dog` and `squirrel` subsets had a single classification value, so subdivision stopped at that point. The subset containing tuples with an animal value of `cat` was further divided about the $A_2 =$ `age` attribute. The resulting subsets are made up of a single attribute value, so subdivision stops. Notice that tuples in the subset with age value $a_{2,3} = 4$ had three classification values: `medium`, `medium`, and `high` (the tuple {`cat`, 4} appears three times in the training set, twice classified as `medium`, and once classified as `high`). This is an example of noise or uncertainty.

Following Quinlan, we note that `medium` has the highest frequency, so it is chosen as the winning classification value for this node in the tree

Classification of an unknown tuple is performed by matching the tuple's attribute values against each node in the decision tree. When a leaf node is reached, that node's winning classification value is assigned to the tuple. For example, an unknown tuple (`squirrel`, 2) would pass through the interval containing `squirrel` in the root node, then be assigned a classification value of `high`. An unknown tuple (`cat`, 4) would pass through the interval containing `cat` (in the root node), then through the interval containing 4 (in the child node), and be assigned a classification value of `medium`.

7.5 Statistical Tables

Chan and Wong describe a data classification technique based on statistical theory [Chan and Wong, 1991]. The user provides an initial training set, with each tuple classified using one of P possible values $\{c_1, \ldots, c_P\}$. Each tuple consists of N attributes $A_1, \ldots, A_N$, where each $domain(A_i) = \{a_{i,j} \mid j = 1, \ldots, n_i\}$ has n_i possible values. The algorithm associates an attribute value $a_{i,j}$ to a possible classification c_k using a weight. A positive weight implies that evidence exists in the training set to support classifying a tuple with attribute value $a_{i,j}$ as c_k. A negative weight implies that evidence exists in the training set to support classifying a tuple with attribute value $a_{i,j}$ as something other than c_k. Tables are built to provide weights for all possible combinations of attribute and classification values. This allows unclassified tuples to be "tested" against all possible classifications. The classification that provides the highest positive weight is assigned to the tuple.

The explanation that follows assumes every attribute has a discrete domain. Continuous attributes are first discretized using a simple maximum entropy approach. An at-

group	original $a_{i,j}$	group range
1	$\{1, 9, 15\}$	$g = 1 \Longleftrightarrow 1 \leq a_{i,j} < 15$
2	$\{20, 28, 32, 33\}$	$g = 2 \Longleftrightarrow 15 \leq a_{i,j} < 33$
3	$\{35, 37, 38, 40\}$	$g = 3 \Longleftrightarrow 33 \leq a_{i,j} < 40$
4	$\{45, 45, 45, 47\}$	$g = 4 \Longleftrightarrow 40 \leq a_{i,j} \leq 47$

Table 7.2: An example of a continuous attribute A_i being ranged into four separate groups

tribute A_i in the training set with n_i values is sorted and split into m groups (where m is a user-chosen constant) by placing $\frac{n_i}{m}$ values in each group. For example, attribute $A_i = \{1, 9, 15, 20, 28, 32, 33, 35, 37, 38, 40, 45, 45, 45, 47\}$ would be split into four groups as shown in Table 7.2.

A_i is now treated as a discrete attribute with four unique values. Chan believes this technique minimizes information loss by more closely matching the attribute's original probability distribution function. His method has been extended to iteratively adjust range boundaries in an attempt to provide an optimal discretization [Wong and Chiu, 1987; Chiu et al., 1991].

If we compute weights for all n_i values in $domain(A_i)$ and all P possible classifications, we obtain an $n_i \times P$ table that completely describes A_i. For any value $a_{i,j} \in domain(A_i)$, we can provide positive or negative evidence for choosing classification c_k. Based on value $a_{i,j}$, we would pick the classification c_k with the highest positive weight.

Chan's classification method builds tables for each of the N attributes $A_1, \ldots, A_N$. An unknown tuple $(v_1, \ldots, v_N)$ is classified as follows.

- for a given classification c_k, for each value $v_i \in domain(A_i)$ of the tuple, obtain the weight to support c_k
- sum the N weights, giving a total weight to support classification c_k for the tuple

- perform the above steps for all P classifications; choose the classification that gives the highest positive weight, and assign it to the tuple

Weights for each value $a_{i,j}$ are computed as follows. For a given attribute A_i let o_{kj} be the number of tuples in the training set with classification c_k and attribute value $a_{i,j}$. If c_k is independent of $a_{i,j}$, the expected number of values is $e_{kj} = o_{k+}o_{+j}/N'$, where $o_{k+} = \sum_{j=1}^{n_i} o_{kj}$, $o_{+j} = \sum_{k=1}^{P} o_{kj}$, and $N' = \sum_{k,j} o_{kj}$. The measure

$$X^2 = \sum_{k=1}^{P} \sum_{j=1}^{n_i} \frac{(o_{kj} - e_{kj})^2}{e_{kj}} \tag{7.5}$$

is sometimes used to determine whether the classification is dependent on the given attribute. If X^2 is greater than the chi-squared value $\chi^2_{n_i-1,\alpha}$, then with confidence level $1 - \alpha$ a dependency exists. This does not, however, tell us whether a specific value $a_{i,j} \in domain(A_i)$ contributes information to a particular classification. This would be true if Pr(class is $c_k \mid A_i = a_{i,j}$) is significantly different from Pr(class is c_k). We know that Pr(class is $c_k \mid A_i = a_{i,j}$) $= o_{kj}/o_{+j}$ and Pr(class is c_k) $= o_{k+}/N'$. Simple reduction shows that the problem is to determine whether o_{kj} is significantly different from e_{kj}.

Using a modified chi-squared value $z_{kj} = (o_{kj} - e_{kj})/\sqrt{e_{kj}}$ and the maximum likelihood estimate of variance $\nu_{kj} = (1 - o_{k+}/N')(1 - o_{+j}/N')$, the measure of difference is defined as

$$d_{kj} = \frac{(o_{kj} - e_{kj})/\sqrt{e_{kj}}}{\sqrt{\nu_{kj}}} = \frac{z_{kj}}{\sqrt{\nu_{kj}}} \tag{7.6}$$

If $d_{kj} > 1.96$, we conclude with confidence level 95% that a tuple with value $a_{i,j}$ is more likely to belong to c_k than to other classes. If $d_{kj} < -1.96$, we conclude that a tuple with

value $a_{i,j}$ is more likely to belong to classes other than c_k. For attributes with a significant d_{kj} value, a weight W is computed using the mutual information measure

$$I(class = c_k : a_{i,j}) = log_2 \frac{Pr(c_k \mid a_{i,j})}{Pr(c_k)} \tag{7.7}$$

W is defined as $I(class = c_k : a_{i,j}) - I(class \neq c_k : a_{i,j})$, the difference in the gain in information by assigning a tuple with attribute value $a_{i,j}$ to class c_k rather than to some other class. As mentioned above, a positive weight supports classification c_k; a negative weight supports a classification other than c_k.

Results from various experiments showed Chan's algorithm to be a computationally simple yet effective method for classification. Statistical tables were shown to outperform the ID3 method of classification. Classifying with weights is also useful from a data exploration perspective. A large positive weight provides strong positive evidence for class c_k. Similarly, a large negative weight provides strong negative evidence for class c_k. Chan's method handles noise, errors, and missing values in the training set because these will be "averaged into" the cumulative weight tables. Moreover, weights can offer the user a measure of confidence for the classification chosen. They could even be used to provide multiple candidate classifications.

7.6 Interval Classification

Agrawal proposes a method of tree classification based on intervals [Agrawal et al., 1992]. The interval classifier tries to improve efficiency by reducing the number of intervals in each node in the classification tree. This improves the time required both for generating the initial tree and retrieving results from it.

A_1	A_2	class
dog	6	low
cat	6	low
cat	6	low
dog	10	high
cat	10	high
whale	3	low

Table 7.3: An example training set for interval classification; tuples in the training set have two attributes A_1 = `animal` and A_2 = `age`; each tuple is classified using one of three possible values

As with Chan's statistical tables, the user provides a training set with each tuple classified using one of P possible values. The algorithm begins by computing frequency histograms for each attribute A_i. For discrete attributes, this simply requires counting the number of times each attribute value $a_{i,j} \in A_i$ occurs in the training set. Histograms are sets $\{(a_{i,1}, f_{i,1}), \ldots, (a_{i,n_i}, f_{i,n_i})\}$ of values with their corresponding frequencies. A classification list $g_{i,j}$ is also maintained for each $a_{i,j}$. Consider the training set shown in Table 7.3. The frequency histogram of A_1 = `animal` would be

$(a_{1,j}, f_{1,j})$	$g_{1,j}$
(dog, 2)	{ (low, 1), (high, 1) }
(cat, 3)	{ (low, 2), (high, 1) }
(whale, 1)	{ (low, 1) }

Continuous attributes need to be discretized in some way. A smoothed frequency histogram is returned from this discretization. Given a continuous attribute A_i, an initial frequency histogram H_{org} is built from the available attribute values. This provides a set of values $\{(a_{i,1}, f_{i,1}), \ldots, (a_{i,n_i}, f_{i,n_i})\}$ with their corresponding frequencies. The range $[a_{i,1} \ldots a_{i,n_i}]$ of the attribute is divided into equidistant intervals of width h. The smoothed frequency f for

any point v (where $a_{i,1} \leq v \leq a_{i,n_i}$) can be computed by considering the contribution of each $a_{i,j} \in H_{org}$. $W(u)$ is defined to be

$$W(u) = \begin{cases} 1 + 2cos(\pi u) & \text{if abs}(u) < 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{7.8}$$

The contribution from an individual point $(a_{i,j}, f_{i,j}) \in H_{org}$ is defined as

$$f_{i,j} \cdot W((v - a_{i,j})/h)/h \tag{7.9}$$

that is, any point $a_{i,j}$ that is within $\frac{1}{2}h$ of v contributes to the smoothed frequency f of v. The strength of the contribution falls off according to the cosine function as $a_{i,j}$ moves farther from v. This means f is simply the sum of contributions from each $a_{i,j}$, namely

$$f = \frac{1}{n_i} \sum_{i=1}^{n_i} f_{i,j} \cdot W((v - a_{i,j})/h)/h \tag{7.10}$$

Consider a continuous attribute A_i = { 30, 31, 32, 32, 36, 44, 44, 44, 44, 46, 46, 46 }. The initial frequency histogram would contain six entries, specifically H_{org} = { (30,1), (31,1), (32,2), (36,1), (44,4), (46,3) }. If we split A_i into five intervals, the range boundaries would be (30, 33.2, 36.4, 39.6, 43.8, 46) and the width of each interval would be $h = 3.2$. The frequency f for range boundary $v = 30$ can then be computed as shown in Table 7.4.

Smoothed frequencies for the remaining range boundaries can be computed in a similar manner. Group information also needs to be included at each range boundary. This is computed in a manner similar to attribute frequency. Recall that each $a_{i,j} \in H_{org}$ has a

$a_{i,j}$	$abs(\frac{v-a_{i,j}}{h})$	contribution
30	0	$1 \cdot 2cos(0)/3.2$
31	$\frac{1}{3.2}$	$1 \cdot 2cos(-\pi/3.2)/3.2$
32	$\frac{2}{3.2}$	0
36	$\frac{6}{3.2}$	0
44	$\frac{14}{3.2}$	0
46	$\frac{16}{3.2}$	0
		$f = \frac{1}{6} \cdot 1.6$

Table 7.4: An example of computing the frequency for the range boundaries of a continuous attribute during interval classification

corresponding classification list $g_{i,j}$, a set of classification values and frequencies. Assume, for example, $a_{i,j} = (31, 3)$ with a corresponding classification list $g_{i,j} = \{(\texttt{low}, 2), (\texttt{high}, 1)\}$. If $a_{i,j}$ contributes to some range boundary v with frequency $f = 0.25$, the corresponding classification value contribution would be $(0.25 \cdot 2)$ `low` and $(0.25 \cdot 1)$ `high`.

Once frequency histograms are available for every attribute, the remainder of the algorithm works recursively as follows.

- A "winning" attribute A_{win} is chosen that minimizes resubstitution error rate. The resubstitution error rate for an attribute A_i is $1 - \sum_{j=1}^{n_i} winner\_freq(a_{i,j})/total\_freq$, where $winner\_freq(a_{i,j})$ is the frequency of the winning classification value for attribute value $a_{i,j} \in A_i$, and $total\_freq$ is the total frequency of all classification values over all attribute values of A_i

- Each interval in A_{win} is classified as strong or weak. An interval is strong if the ratio of the winning classification frequency to the total frequency for the given interval is greater than a precision threshold $1 - (curr\_depth/max\_depth)^2$

- Attributes that can be ordered (*i.e.,* continuous attributes) merge adjacent intervals to reduce the width of the corresponding tree node. If two adjacent intervals have the same winning classification value and strength, they are merged into one
- For each weak interval, all the tuples from the original dataset that correspond to the interval are gathered and sent recursively to the classification algorithm

The result of the algorithm is an interval tree. Unclassified tuples can use the tree to obtain classification values. Each node in the tree corresponds to some attribute A_i. If the tuple has value $a_{i,j}$ for attribute A_i, the interval in the node that contains $a_{i,j}$ is selected. If the node is internal to the tree, the tuple is compared recursively to the interval's child. If the node is a leaf, the winning classification value for the selected interval is assigned to the tuple.

7.7 Rough Sets

Ziarko presents a method of data classification based on mathematical rough sets [Ziarko, 1991]. Rough sets were developed to allow modelling of general, imprecise, or imperfect data [Paulak, 1991]. Ziarko uses them to reduce information loss due to the use of statistical classification methods. He feels statistical models are often too specific, or require too many underlying statistical assumptions (*e.g.,* a particular probability distribution function). Since rough sets were designed for and have been used in a number of information representation models, Ziarko believes they will be useful for building classification rules in databases.

In database terms, a rough set can be thought of as a representation of objects in a universe U using one or more attributes $A = \{A_1, \ldots, A_N\}$. The question of interest is how well A can be used to characterize a subset X of a universe U. For example, suppose U

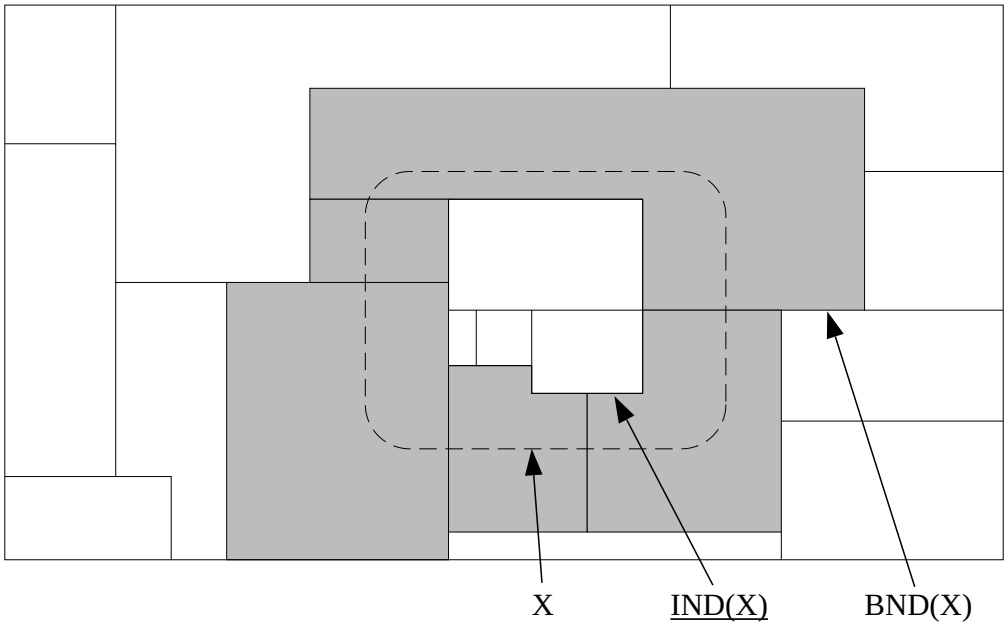


Figure 7.2: Example of a diagram of a rough set, each rectangle is an elementary set from IND. The dashed boundary represents elements of set X; embedded white region represents $\underline{IND(X)}$; grey region represents $BND(X)$

is a collection of cars, A is a single attribute representing a car's manufacturer, and X is the subset of cars that achieve high gas mileage. Is knowing the manufacturer enough to determine the type of mileage the car gets?

X can be formally expressed using an equivalence relation IND over U. IND is the set of attribute values or "elementary sets" that describe each $x \in X$. For example, if $X = \{$Accord, Camry, Tercel, Civic, Escort$\}$, then the corresponding attribute for each $x \in X$ is the collection $\{$Honda, Toyota, Toyota, Honda, Ford$\}$. Let IND' be a collection of all elementary sets from IND. The uncertainty of the representation is defined formally as

$$\begin{aligned} \overline{IND(X)} &= \bigcup \{Y \in IND' : Y \cap X \neq 0\} \\ \underline{IND(X)} &= \bigcup \{Y \in IND' : X \supseteq Y\} \end{aligned} \tag{7.11}$$

$\underline{IND(X)}$ is the lower approximation of X, the union of elementary sets Y that are completely contained in X. $\overline{IND(X)}$ is the upper approximation of X, the union of elementary sets with at least one element contained in X. If the lower approximation and upper approximation are different, X cannot be precisely specified using elementary sets, and we say X is rough. Suppose in our example the elementary sets were Honda = {Accord, Civic}, Toyota = {Tercel, Corolla, Camry}, Ford = {Escort, Taurus}. Then $\underline{IND(X)}$ = {Honda} and $\overline{IND(X)}$ = {Honda, Toyota, Ford}. This implies X is rough (Figure 7.2).

The degree of imprecision can be measured in two ways. The boundary of X $BND(X) = \overline{IND(X)} - \underline{IND(X)}$ contains elements from U whose membership status in X cannot be determined using the current set of attributes (*e.g.,* Taurus is a Ford, and Ford $\in \overline{IND(X)}$ but Ford $\notin \underline{IND(X)}$, so is Taurus $\in X$?). $m(X) = card(\underline{IND(X)})/card(\overline{IND(X)})$ is an accuracy measure that indicates how well the given attributes classify X. If $m(X) < 1$, then X is rough.

For the purpose of data classification, the tuples in the original dataset form the universe U. Recall that each tuple is assigned some c_k from a set of P known classes. Formally, an information function $f : U \times A \rightarrow V$ is defined that, for every $A_i \in A$ assigns some $a_{i,j} \in A_i$ to each $u \in U$. That is, f determines attribute values for every tuple in U. For any subset of attributes $S \subseteq A$, we can define the approximate coverage of S as $IND(S)$, specifically

$$(x, y) \in IND(S) \iff f(x, a) = f(y, a) \; \forall \; x, y \in U \text{ and } a \in S \tag{7.12}$$

$IND(S)$ splits U into groups, where elements in a group have exactly the same attribute values for every $A_i \in S$. Let $P' = IND(P)$, that is, each value in P' is a set of elements from U who all have the same classification c_k. The positive coverage of an attribute subset S is defined as

$$POS(S,P) = \bigcup \{\underline{IND(S,Y)} : Y \in P'\} \tag{7.13}$$

where $\underline{IND(S,Y)}$ is the lower approximation of the set Y in terms of elementary groups of $IND(S)$. In other words, for each group in $IND(S)$, if all the elements of the group are contained within a single group from P', the group is part of $POS(S,P)$. This means any element in $POS(S,P)$ can be uniquely classified into some $c_k \in P$ based only on the attributes in S. The degree of positive coverage is defined as

$$k(S,P) = card(POS(S,P))/card(U), \quad 0 \leq k \leq 1 \tag{7.14}$$

where k provides a measure of the dependency between P and S. If $k = 1$, then P is completely dependent on S. If $k = 0$, then P is completely independent of S. Otherwise P is partially dependent on S, as $0 < k < 1$.

k can be used to find a minimal subset of attributes that provides the same positive coverage as A. Suppose $k_{org} = k(A,P)$. Any subset of attributes $R \subseteq A$ with $k(R,P) = k_{org}$ can be used in place of A, since P is as dependent on R as it is on A. This generalizes classification of tuples by removing redundant attributes. It will also reduce the number of classification rules produced. Usually, there are many different subsets R such that $k_{org} = k(R,P)$. Ziarko uses the one with the minimum number of attributes, to maximize generality and minimize the number of classification rules.

Once a minimal subset R is found, each attribute $A_i \in R$ is assigned a weight. The weight represents the relative contribution of the attribute to the dependency between R and P. This significance factor is defined as

$$SGF(A_i, R, P) = [k(R, P) - k(R - \{A_i\}, P)]/k(R, P) \tag{7.15}$$

The original dataset U is generalized by computing $POS(R, P)$. This eliminates any $u \in U$ that have the same attribute values for every $A_i \in R$ but different classification values. Tuples from $POS(R, P)$ are further reduced by using only the attributes in R. Any duplicate tuples that may be formed are discarded. Some additional reduction can be performed in certain cases. Suppose n_i reduced tuples have the same classification and differ in only one attribute A_i. If the size of $domain(A_i)$ is n_i, these tuples can be replaced by a single tuple with a "do not care" value for attribute A_i. Any value of A_i matches the "do not care" indicator. For example, if attribute $Manufacturer$ = { Honda, Toyota, Nissan, Ford }, the tuples

A_1	A_2		A_k	P
Manufacturer	FuelSys	$\cdots$	Weight	Mileage
$\cdots$				
Honda	EFI	$\cdots$	medium	medium
Ford	EFI	$\cdots$	medium	medium
Nissan	EFI	$\cdots$	medium	medium
Toyota	EFI	$\cdots$	medium	medium
$\cdots$				

would reduce to

A_1	A_2		A_k	P
Manufacturer	FuelSys	$\cdots$	Weight	Mileage
$\cdots$				
—	EFI	$\cdots$	medium	medium
$\cdots$				

Unclassified tuples can be matched against any rule in this reduced ruleset in the following manner.

- For each $A_i \in R$, the value $a_{i,j}$ from the unclassified tuple is matched against the corresponding attribute value in the rule
- If the values match, then $SGF(A_i, R, P)$ is added to a weight for this rule
- The rule that produces the highest weight is declared the "winner". The classification for the winning rule is applied to the unclassified tuple

Like Chan's statistical tables, Ziarko's rough set technique assigns weights to each tuple in the reduced ruleset by returning the combined SGF value. It would be easy to extend the method to return a weight for each possible classification c_k. This could be used to provide a measure of confidence for the chosen classification. SGF values could also be used to provide multiple candidate classifications. Finally, techniques based on weights could be combined. One technique could "suggest" the proper choice to another. Results from multiple techniques could be combined to return a more robust classification. Users could interactively control how different techniques communicated and interacted with one another.

Chapter 8

Knowledge Discovery in Visualization

Our review of knowledge discovery in database algorithms (KDD algorithms) shows they can, in their current form, process typical scientific datasets. This does not mean the algorithms can be integrated directly into a visualization environment. KDD algorithms build rules from a training set, then provide classification values for one or more unclassified tuples. A classification value alone may not be all that is required by the user, however. In this chapter we extend the four KDD algorithms by providing attribute significance values when the classification rules are built, and a confidence value for each classification performed.

Consider the following examples, which show the type of information the KDD algorithms need to provide during visualization. A user performs knowledge discovery on a dataset, then wants to display the attributes that were used to build the classification rules. The KDD algorithms (in their current form) do not provide this information, because the rules are hidden from the user. The rules themselves are strongly influenced by the training set. Any tuple that differs significantly from tuples in the training set (*e.g.,* a tuple with attribute values not contained in the training set) is difficult to classify, because the KDD algorithm has no previous evidence to use to predict a proper classification value. A KDD algorithm should detect these kinds of tuples. The user could then be warned that the algorithm is "not confident" about the corresponding classification value it suggests.

To work well in a visualization environment, the KDD algorithms need to be extended so that they can answer the following questions:

- How confident is the algorithm about the classification value it suggests (*i.e.,* how closely does the unclassified tuple "fit" the rules built from the original training set)?

  KDD algorithms cannot accurately classify tuples that do not fall into patterns observed in the training set. A classification weight would allow the user to determine the algorithm's "confidence" in the classification value it suggests. This weight might be used to ignore certain classifications. It could also be mapped to a visual feature, allowing the user to see both the classification value and the confidence weight during visualization.

- How "good" is the classification value suggested by the KDD algorithm, compared to other possible classification values?

  The KDD algorithms return a single "winning" classification value. It would often be useful to know how well other classification values fit an unknown tuple. The classification weights described above could be used for this purpose. The winning classification value will have the highest classification weight. If a KDD algorithm can return confidence weights for an unknown tuple and user-chosen classification values, users would be able to see how the algorithm ranked different classifications. This would also provide some indication of how much better the winning classification value was, vis-a-vis other possible values.

- Which attributes are significant to the classification being performed and which are not?

  Knowing this would allow the user to reduce the number of attributes to display, by focusing only on the dependent (or the independent) attributes. If users have some

advance knowledge about dependencies in the dataset, they can check to ensure that a KDD algorithm does not misinterpret the training set and ignore attributes that should be used to build the classification rules.

8.1 Classification Weights

Each of the four KDD algorithms was extended to include a classification weight with each classification value it suggests. Given an unknown tuple $(v_1, \ldots, v_N)$, any of the algorithms can provide a recommended classification value c_k. The classification weight *strength* is a measure of how "confident" the algorithm is about its suggested value.

We used the following guidelines when we extended the algorithms to provide confidence weights:

- *strength* was computed using the same mathematical and statistical algorithms and values that are used to choose the classification value c_k
- The algorithms to compute *strength* were kept computationally and conceptually as simple as possible

The result is a normalized confidence weight $0 \leq strength \leq 1$ provided for each classification. The larger the value of *strength*, the more confident an algorithm is about the classification value it returns.

Decision Trees

Classifying an unknown tuple $(v_1, \ldots, v_N)$ generates a path through a decision tree from root to leaf. Each node in the tree corresponds to some attribute A_i. The interval in the node that contains v_i is used to direct the path as it passes through A_i. The strength of the classification is defined to be the winning value's frequency in the leaf node interval of the classification path. In other words, given a leaf node representing attribute A_i:

$$strength = winner\_freq(v_i)/interval\_freq \tag{8.1}$$

where $winner\_freq(v_i)$ is the frequency of the winning classification value and $interval\_freq$ is the total frequency of all classification values in the interval containing v_i. For example, consider a leaf node representing the discrete attribute animal:

attribute value	*classification value(s)*
dog	{A, B}
cat	{A, A, B}
whale	{A}

Table 8.1: An example of attribute values and their corresponding group value(s) for a leaf node in a decision tree; the strength of a classification value is based on the frequency of the winning classification value in the given leaf node interval

A tuple with v_i = `cat` passes through the interval containing `cat` during classification. This interval has a winning classification value A, with $winner\_freq(cat) = 2$ and $interval\_freq = 3$, resulting in $strength = \frac{2}{3}$. Classification strength measures how strongly tuples from the training set agreed on a common classification value when the given interval was built.

Users can also ask for the strength associated with a specific classification value c_p. In this case, $winner\_freq(v_i)$ is replaced with $group\_freq(c_p)$, where $group\_freq(c_p)$ is the frequency of classification value c_p in the given interval. For example, the same tuple shown above with $v_i =$ `cat` would have a strength of $\frac{1}{3}$ for classification value B.

Statistical Tables

Chan and Wong's statistical tables use a built-in weighting scheme to perform classification. Consider an unknown tuple $(v_1, \ldots, v_N)$ and the statistical table used to weight attribute values $v_i \in domain(A_i)$. The table contains $n_i \times P$ entries for all possible combinations of attribute values $a_{i,j}$ and classifications c_k. We can select from the table the minimum value min_i, the maximum value max_i, and the weight range $r_i = max_i - min_i$. The winning classification value c_k is the one that provides the largest total positive weight $\forall\, v_i$ across all N tables. Given an individual table value of W_i for classifying a tuple with attribute v_i as c_k, the strength of the classification is defined as:

$$strength = \sum_{i=1}^{N}(W_i - min_i) \,/\, \sum_{i=1}^{N} r_i \tag{8.2}$$

Dividing by the sum of the weight ranges normalizes *strength* over the range $(0 \ldots 1)$. As with decision trees, the strength for a particular classification value c_p can be provided by finding table weights w_i for classifying v_i as c_p, then computing $\sum_{i=1}^{N}(w_i - min_i) \,/\, \sum_{i=1}^{N} r_i$.

Interval Classification

An interval classification tree is, in essence, a decision tree that has been modified for efficiency in the following ways:

- Continuous attributes are divided into ranges; the group frequencies for each range are computed using a cosine filter; the number of ranges is meant to be much smaller than the number of unique values in the original attribute
- Nodes in an interval tree are marked as strong or weak based on the frequency of the winning classification value in the node; the frequency required to mark a node as strong decreases depending on the depth of the node in tree
- Adjacent intervals in a node are merged if they have the same winning classification value and the same strength (*i.e.,* if they are both strong or both weak)

These improvements do not affect the basic workings of the classification weighting algorithm used for decision trees. Because of this, interval classification uses exactly the same formula as decision trees:

$$strength = winner\_freq(v_i)/interval\_freq$$

where $winner\_freq(v_i)$ is the frequency of the winning classification value and $interval\_freq$ is the total frequency of all classification values in the leaf node interval containing v_i. The algorithm for computing the strength of a particular classification value c_p is also the same, $strength(c_p) = group\_freq(c_p) \,/\, interval\_freq$.

Rough Sets

The rough set algorithm assigns a degree of coverage SGF to each attribute A_i in the minimal attribute set R (Equation 7.15). An unknown tuple $\{v_1, \ldots, v_N\}$ is matched against a rule r_j by comparing each v_i against the corresponding attribute value in the rule. If they match, then $SGF(A_i, R, P)$ is added to the weight of evidence W_j for choosing r_j. The winning rule

is the rule that produces the highest total weight W_{win}. The maximum possible weight is $W_{max} = \sum_{i=1}^{N} SGF(A_i, R, P),\ \forall\ A_i \in R$. The classification weight is therefore defined to be:

$$strength = W_{win} \,/\, W_{max} \tag{8.3}$$

As in the previous three algorithms, dividing by W_{max} guarantees $0 \leq strength \leq 1$. The strength for a particular classification value c_p can be found by examining the rules that correspond to c_p. Assuming that rule r_p produces the highest total weight w_{win} for an unknown tuple, then $strength(c_p) = w_{win} \,/\, W_{max}$.

8.2 Results

We built an experimental training set with the following structure to test classification weights for each of the four extended KDD algorithms:

- each tuple consists of three attributes: age (`age`), a continuous integer attribute on the range $[20, \ldots, 80]$; education level (`elvl`), a discrete attribute with five possible values $\{0, 1, 2, 3, 4\}$; and zipcode (`zip`), a discrete attribute with nine possible values
- one of two possible classification values, A or B, is assigned to each tuple using the following rules:

$$\begin{array}{lll} \text{Group A:} & \texttt{age} < 40 & \& \ \texttt{elvl} \in \{0, 1\} \\ & 40 \leq \texttt{age} < 60 & \& \ \texttt{elvl} \in \{0, 1, 2, 3\} \\ & 60 \leq \texttt{age} & \& \ \texttt{elvl} = 0 \\ \text{Group B:} & \text{otherwise} & \end{array}$$

The classification was designed specifically to be independent of `zip`. This set of rules is exactly the same as the one used by Agrawal et al. [1992] to test their interval classification technique. Our training sets contained 61 different `age` values by 5 different `elvl` values for a total of 305 tuples. One of the nine zipcodes was randomly assigned to each tuple.

Classification errors were included in the training set in certain circumstances to test two separate properties of each extended KDD algorithm. First, rules built from a training set are skewed by the errors it contains. This in turn causes mistakes when we try to classify unknown tuples. Classification weights for these tuples should be lower than for tuples that are assigned a correct classification value. Training sets that contain errors also allow us to test an extended KDD algorithm's sensitivity. That is, does a small increase in errors in the training set result in a small or a large increase in the number of classification errors?

Each extended KDD algorithm was run four times using four different training sets. In the first training set none of the tuples had an incorrect classification value. In the second training set 15 tuples were randomly selected and assigned an incorrect classification value (*i.e.,* a 5% error rate). The third training set had a 10% error rate, and the fourth training set had a 25% error rate. After training, an algorithm was asked to classify 305 unknown tuples. The classification values provided by the algorithm were compared against the correct classification values (determined using the classification rules shown above). Classification weights and error rates could then be examined.

Decision Trees

The ID3 algorithm did an excellent job of classifying unknown tuples, even when the training set had a 10% error rate (the classification error rate was 0%, 0%, and 3.3% for the 0%, 5%, and 10% error training sets, respectively). Tuples that were incorrectly classified had correspondingly low classification weights (Figure 8.1).

Decision Tree Classification Error Results

Training Set	Minimum Class. Weight	Misclassified Tuples
0% error	0.80	0.0%
5% error	0.60	0.0%
10% error	0.60	3.0%
25% error	0.64	24.6%

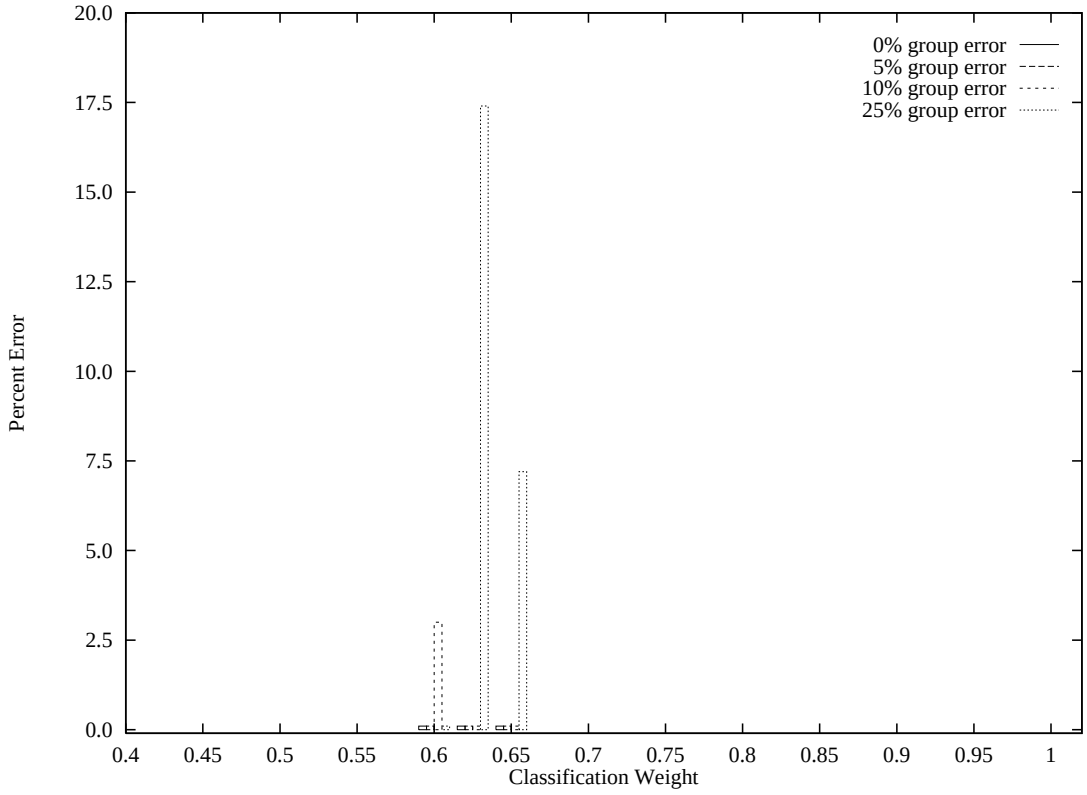


Figure 8.1: The location (in terms of classification weight) and number of classification errors for each of the four training sets; the table above the graph lists the minimum classification weight and the percentage of misclassified tuples for each of the four training sets; the location of the bar in the graph represents the classification weight for the incorrectly classified tuple, while the height of the bar represents the number of incorrect tuples with the given weight

Classification errors increased dramatically for the 25% error training set. The errors in the training set caused ID3 to identify `age` as independent of the classification being performed. The algorithm then tried to classify unknown tuples based only on `elvl`, resulting in a large number of incorrect classifications. Despite the algorithm's poor performance, however, classification weights were appropriately low for the erroneous tuples ($0.64 \leq strength \leq 0.66$).

Our results suggest that the ID3 algorithm provides effective classification up to the point where errors in the training set begin to mask attributes that should be included in the decision tree. The loss of attributes on which the classification depends results in a significant increase in classification errors.

Statistical Tables

Chan and Wong's statistical table algorithm performed very well for all four training sets (classification error rates ranged from 0% to 5%). An increase in the number of errors in the training set did not result in a large increase in the number of incorrectly classified tuples. Moreover, classification weights for those few tuples that were assigned the wrong classification value were appropriately low (Figure 8.2).

The main problem with the statistical table algorithm was the number of tuples that could not be classified at all. The algorithm uses each attribute value $v_i \in V$ as an index into a $n_i \times P$ table, to find positive or negative evidence for choosing a particular classification. If attribute value v_i was deemed insignificant for every possible classification value c_k when the tables were built, no evidence, either positive or negative, can be derived from v_i. When this is true for every $v_i \in V$, the algorithm cannot suggest any particular c_k for V. This is exactly what happened when the error rate in the training sets rose. Significant trends in the dataset were broken, and an increasingly large number of table entries were marked as

insignificant. The number of unclassified tuples ranged from 0% for the 0% error training set to 22.6% for the 25% error training set.

Our results show that the statistical table algorithm can be extremely sensitive to errors in the training set. This does not result in classification errors; rather, the algorithm is unable to classify certain tuples due to lack of significant trends in the training set.

Interval Classification

The interval classification algorithm did a good job of classifying unknown tuples using any of the four training sets (the classification error rate ranged from 0.7% to 3.3%). Classification weights for the tuples with an incorrect classification value were low relative to the weights for correct classifications.

Unlike the ID3 algorithm, interval classification did not "lose" important attributes as training set errors rose (see Error Results in Table 8.3). Apparently, this was because interval classification chose significant attributes at each node in the tree by minimizing resubstitution error. This is less sensitive to errors in the training set, compared to the modified chi-squared technique used by ID3.

Rough Sets

Like interval classification, the rough set algorithm did very well classifying unknown tuples, regardless of the training set that was used. Classification error rates ranged from 1% to 7.9% (Figure 8.4).

Unfortunately, the weights assigned to incorrectly classified tuples were all 1.0. Classification weights provided no indication of when classification values might be wrong. This is a

Statistical Table Classification Error Results

Training Set	Minimum Class. Weight	Misclassified Tuples	Unclassified Tuples
0% error	0.51	3.3%	0.0%
5% error	0.50	0.0%	8.2%
10% error	0.51	5.2%	8.5%
25% error	0.51	3.3%	22.6%

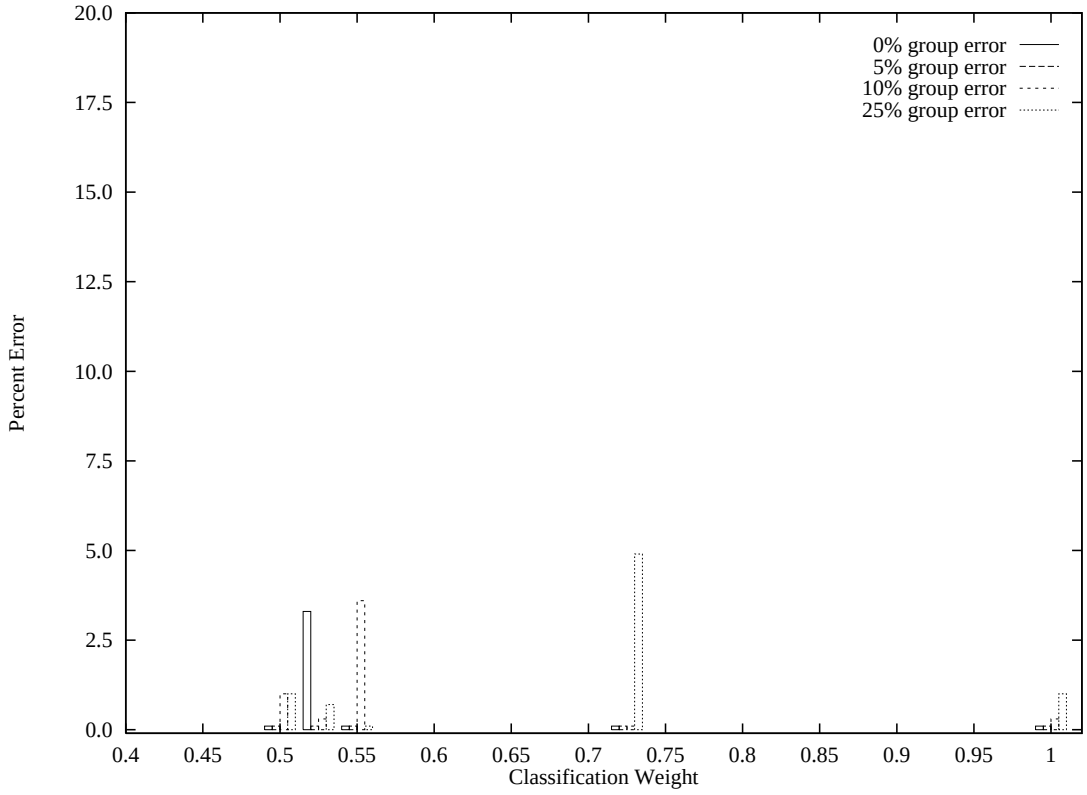


Figure 8.2: The location (in terms of classification weight) and number of classification errors for each of the four training sets; the table above the graph lists the minimum classification weight, the percentage of misclassified tuples, and the percentage of unclassified tuples for each of the four training sets; the location of the bar in the graph represents the classification weight for the incorrectly classified tuple, while the height of the bar represents the number of incorrect tuples with the given weight

Interval Classification Error Results

Training Set	Minimum Class. Weight	Misclassified Tuples
0% error	0.57	0.7%
5% error	0.57	0.7%
10% error	0.50	2.0%
25% error	0.64	3.3%

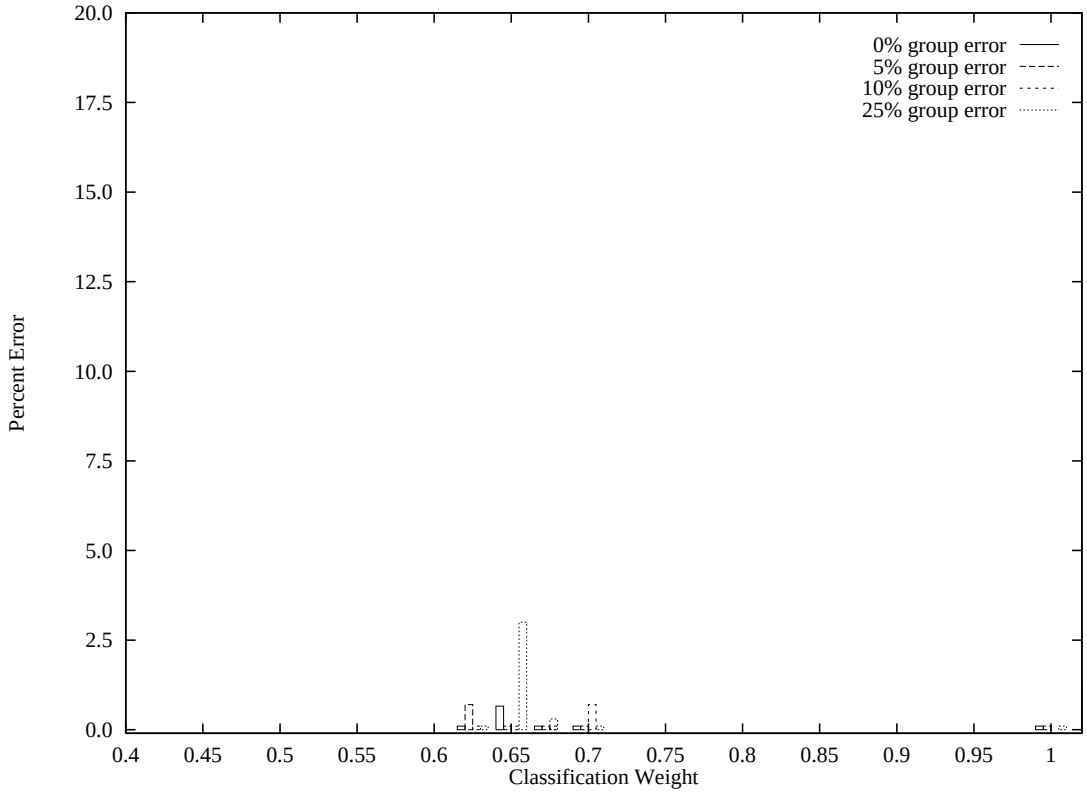


Figure 8.3: The location (in terms of classification weight) and number of classification errors for each of the four training sets; the table above the graph lists the minimum classification weight and the percentage of misclassified tuples for each of the four training sets; the location of the bar in the graph represents the classification weight for the incorrectly classified tuple, while the height of the bar represents the number of incorrect tuples with the given weight

direct consequence of errors in the training set. The rough set algorithm has difficulty handling these kinds of errors. For example, consider a training set that included the following tuples:

(35, 1, V6R 3C4)	A
(35, 1, V6R 5X4)	B
(35, 1, V2L 3X9)	A
(35, 1, N2L 3X1)	A
(35, 1, M4W 3Z2)	A

Tuple (35, 1, V6R 5X9) has been incorrectly classified as B. This forces the rough set algorithm to use all three attributes to build its rules (since no two attributes provide a positive coverage $k(R, P)$ equal to the original coverage k_{org}). The resulting rules are something like:

Group = A	Group = B
(35, 1, V6R 3C4)	(35, 1, V6R 5X4)
(35, 1, V2L 3X9)	
(35, 1, N2L 3X1)	
(35, 1, M4W 3Z2)	

Whenever an unknown tuple V = (35, 1, V6R 5X9) is classified, it will be assigned classification value B, since a rule for that classification matches exactly the attribute values $v_i \in V$. Moreover, since every v_i matched the winning rule, $W_{win} = \sum_{i=1}^{3} SGF(A_i, R, P) = W_{max}$, and $strength = W_{win} \,/\, W_{max} = 1.0$.

The rough set algorithm tries to detect these kinds of errors in the training set by ignoring any tuples that have a common set of attributes but different classification values. This works as planned if two or more instances of the tuple appear in the training set (and if not all of them are incorrectly classified in the same way). Tuples that appear only once in the

Rough Set Classification Error Results

Training Set	Minimum Class. Weight	Misclassified Tuples
0% error	0.95	1.0%
5% error	0.88	1.6%
10% error	0.86	3.3%
25% error	0.77	7.9%

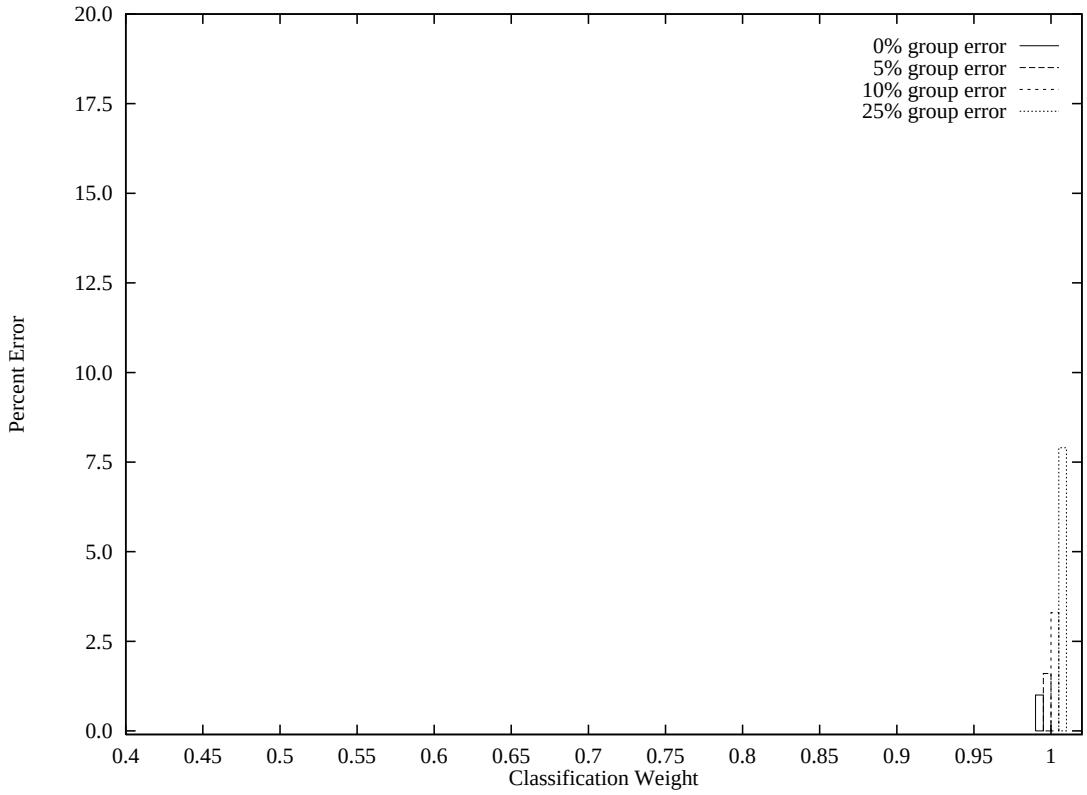


Figure 8.4: The location (in terms of classification weight) and number of classification errors for each of the four training sets; the table above the graph lists the minimum classification weight and the percentage of misclassified tuples for each of the four training sets; the location of the bar in the graph represents the classification weight for the incorrectly classified tuple, while the height of the bar represents the number of incorrect tuples with the given weight

training set will pass through this test, and cause the problem described above. It is not possible to "fix" the classification weights, because the rough set algorithm itself provides no information that can be used to identify the potential classification error.

In spite of this, the classification weights are not completely useless. The algorithm is often asked to classify an unknown tuple with attribute values v_i that did not appear in the original training set. Classification weights for these tuples will be less than 1.0, because attribute values v_i will not match any rule and hence $SGF(A_i, R, P)$ will not be added to W_{win}. This means $W_{win} < W_{max}$, and therefore $W_{win}/W_{max} < 1.0$. A weight of less than 1.0 signals to the user that V did not completely match any rule, and hence there is some measure of uncertainty about which classification value to assign to V. An attribute that is important for classification will have a relatively large SGF value, so not matching on that attribute will result in a relatively low *strength* value.

8.3 Attribute Significance Weights

Each of the four extended KDD algorithms makes use of "significant attribute" values during classification. Attributes that are seen as independent of the classification being performed are usually ignored. The algorithm can then determine the importance of each significant attribute. For example, the decision tree algorithm picks the significant attribute with the largest information gain to partition the root of the tree. The rough set algorithm uses the coverage value SGF to measure attribute significance.

We used the information built-in to each of the four algorithms to provide an attribute significance weight *significance*. Independent attributes have very low significance weights. Significant attributes have significance weights that increase based on their importance to the classification being performed. As before, we used the formulas and values provided by

the individual KDD algorithms to compute attribute significance weights.

Decision Trees

The significance weighting algorithm for attributes in a decision tree was based on the following observations:

- Attributes that are not part of the decision tree are considered independent of the classification being performed, and therefore have a significance weight of zero
- Information gain is maximized at each level of the tree to determine which attribute to use to partition a given node; therefore, an attribute's significance should be based in part on the level of the node (or nodes) it partitions
- Multiple nodes can appear at every level in the tree except the root; therefore, an attribute's significance should be based on the number of nodes at a given level it partitions

These observations were used to build an algorithm for computing an attribute's significance. For each level in the decision tree, an attribute A_i that is used to partition nodes has significance

$$level\_sig(A_i) = \frac{coverage(A_i)}{level} \tag{8.4}$$

where $level$ is the height of the given level in the decision tree, and $coverage(A_i)$ is the number of nodes partitioned by A_i divided by the total number of nodes in the level. The total significance weight for A_i is the sum of $level\_sig(A_i)$ at every level in the tree. As an

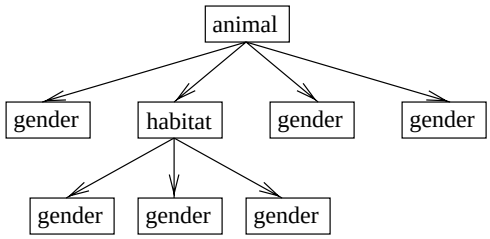


Figure 8.5: An example of a decision tree with three levels and three different attributes: animal, gender, and habitat.

example, consider the decision tree shown in Figure 8.5. The significance weights for each of the three attributes shown in the tree would be:

$$
\begin{aligned}
significance(animal) &= \tfrac{1.0}{1} = 1.0 \\
significance(gender) &= \tfrac{0.75}{2} + \tfrac{1.0}{3} = 0.708 \\
significance(habitat) &= \tfrac{0.25}{2} = 0.125
\end{aligned}
$$

Statistical Tables

Statistical tables differ somewhat from the other algorithms, because independent attributes are not removed before building the classification rules. In spite of this, the individual table for an attribute A_i can be used to determine how relevant A_i is to the classification being performed.

The statistical table for attribute A_i contains $n_i \times P$ entries for all possible combinations of attribute values $a_{i,j}$ and classifications c_k. An individual value $a_{i,j} \in A_i$ is considered relevant with respect to a specific classification value c_k when $abs(d_{pk}) \geq 1.96$. If the classification is strongly dependent on A_i, most of the (attribute value, classification value) pairs $(a_{i,j}, c_k)$ will have a corresponding $abs(d_{pk}) \geq 1.96$. Attributes that are independent of the classification will have a large number of table entries with $-1.96 < d_{pk} < 1.96$. The

significance of the attribute is therefore defined to be:

$$significance(A_i) = sig\_entry(A_i,\, P) \,/\, tot\_entry(A_i,\, P) \tag{8.5}$$

where $sig\_entry(A_i,\, P)$ is the number of d_{pk} values in A_i's statistical table with a value greater than or equal to 1.96, and $tot\_entry(A_i,\, P)$ is the total size of the table, $n_i \times P$.

Interval Classification

As described in the previous section on classification weights, an interval classification tree is a modified decision tree. The modifications improve efficiency, but they do not change the correctness of the significance algorithm. Therefore, attribute significance is computed in exactly the same way as for decision trees, namely:

$$level\_sig(A_i) = \frac{coverage(A_i)}{level}$$

where $level$ is the height of the given level in the interval tree, and $coverage(A_i)$ is the number of nodes partitioned by A_i divided by the total number of nodes in the level. The total significance weight for A_i is the sum of $level\_sig(A_i)$ at every level in the tree.

Rough Sets

The rough set algorithm provides built-in weights SGF representing the importance of each attribute A_i in the minimal attribute set R. A simple examination of Equation 7.15 shows that $SGF(A_i, R, P)$ is guaranteed to lie on the range $(0 \ldots 1)$. Therefore, the significance of attribute $A_i \in R$ is simply

$$significance(A_i) = SGF(A_i, R, P) \tag{8.6}$$

Any attribute $A_i \notin R$ is considered independent of the classification being performed, and is assigned a significance weight of zero.

8.4 Results

We used the same training sets described in the Classification Weights section to test our attribute significance weights. The results are shown in Table 8.2.

KDD Algorithm	0% error	5% error	10% error	25% error
ID3	`elvl`: 1.0	`elvl`: 1.0	`elvl`: 1.0	`elvl`: 1.0
	`age`: 0.5	`age`: 0.5	`age`: 0.5	`age`: 0.0
	`zip`: 0.0	`zip`: 0.0	`zip`: 0.0	`zip`: 0.0
Stat Tables	`elvl`: 1.0	`elvl`: 0.8	`elvl` 0.8	`elvl`: 0.8
	`age`: 0.67	`age`: 0.67	`age`: 0.5	`age`: 0.5
	`zip`: 0.0	`zip`: 0.0	`zip`: 0.0	`zip`: 0.11
IC	`elvl`: 1.0	`elvl`: 1.0	`elvl`: 1.0	`elvl`: 1.0
	`age`: 0.67	`age`: 0.67	`zip`: 0.43	`age`: 0.5
	`zip`: 0.33	`zip`: 0.33	`age`: 0.4	`zip`: 0.34
Rough Sets	`elvl`: 0.65	`age`: 0.65	`age`: 0.65	`age`: 0.75
	`age`: 0.55	`elvl`: 0.64	`elvl`: 0.64	`elvl`: 0.64
	`zip`: 0.05	`zip`: 0.2	`zip`: 0.25	`zip`: 0.48

Table 8.2: Significance weights for each of the four extended KDD algorithms and each of the four training sets; in each case attributes are listed in decreasing order of significance weight

As expected, significance weights are highest for the "most significant" attributes. For the 0% classification error training set, all four algorithms agreed that `elvl` and `age` (in that

order) were significant. `zip` was ignored by the ID3 and statistical table algorithms, and it was given a low significance weight by the interval classification and rough set algorithms.

Errors in the training set caused a corresponding shift in significance weights in each algorithm. For example, the ID3 algorithm assigned `age` a significance of zero for the 25% error training set. As was previously explained, errors in the training set drove the modified chi-squared value for `age` below the required minimum. ID3 therefore assumed `age` was not related to the classification being performed. This resulted in a significance weight of zero, and in a large number of classification errors. Significance weights for `elvl` and `age` dropped sharply in the statistical table algorithm for the 10% and 25% error training sets. Errors in the training set caused an increasing number of table entries to be marked as insignificant. The result was a drop in the attributes' significance weights, and a rise in the number of unclassified tuples. The rough set algorithm identified `age` as more important than `elvl` for the 5%, 10%, and 25% error training sets. However, `elvl` was given a significance nearly equal to `age`, and the weights for `zip` continued to remain relatively low. This resulted in only a small number of classification errors.

Chapter 9

Future Work and Applications

Results from our research have uncovered a number of additional areas that warrant further study. Two of these are described below. First, we plan to complete a study on the use of texture in an exploratory visualization environment. Second, we hope to investigate the use of emergent features for identifying specific combinations of attribute values in a high-dimensional dataset.

After describing these two areas for future work, we then conclude this thesis with a description of our visualization techniques being applied to three practical applications: tracking salmon migration patterns, displaying computerized tomography (CT) medical image slices, and calculating and visualizing changes in sea-surface temperatures. These examples demonstrate how each of our techniques (estimation, colour selection, and attribute compression through knowledge discovery) can be used to produce an effective visualization of the underlying dataset.

9.1 Texture

A large part of this thesis describes an investigation of colour and its use in scientific visualization. Another commonly-used visual feature is texture. Research on the use of texture in

computer vision and computer graphics has tried to identify the fundamental components of texture patterns. Ware and Knight believe orientation, size, and contrast (or density) are the orderable dimensions of a visible texture [Ware and Knight, 1992; Ware and Knight, 1995]. Liu and Picard believe the three most important perceptual dimensions of a texture pattern are periodicity (or repetitiveness), directionality, and granularity (or randomness) [Liu and Picard, 1994]. Liu and Picard's directionality and granularity correspond closely to Ware and Knight's use of orientation and contrast.

Ware and Knight used Gabor filters to create their textures, while Liu and Picard used Wold features. It is also possible to create a "texture icon" that varies across one or more of the basic texture dimensions. Figure 9.1 shows an example of texture icons that are made up of nine bars. A scale factor and an orientation are applied to each texture icon. We chose to vary size and orientation across three separate values, in part because Wolfe's experiments showed that orientation is categorized into three separate ranges (steep, tilted, and shallow) in the low-level visual system [Wolfe et al., 1992]. The texture icons in Figure 9.1 can be used to represent data elements with two attributes, where each attribute is no more than three-valued. If density can be uncoupled from size, we could visualize elements with up to three separate three-valued attributes.

There are a number of important issues that need to be investigated regarding the use of texture in visualization. First, we want to know how the perceptual dimensions of a texture pattern interact with one another. Ware and Knight describe size–location and size–orientation tradeoffs that were discovered during their visualization experiments. Orientation, size, and contrast may also form a visual feature hierarchy, similar to the hue–form and intensity–hue hierarchies described by Callaghan [1984, 1989]. Understanding and controlling these effects is key to building guidelines for the effective use of texture in a real-time visualization environment.

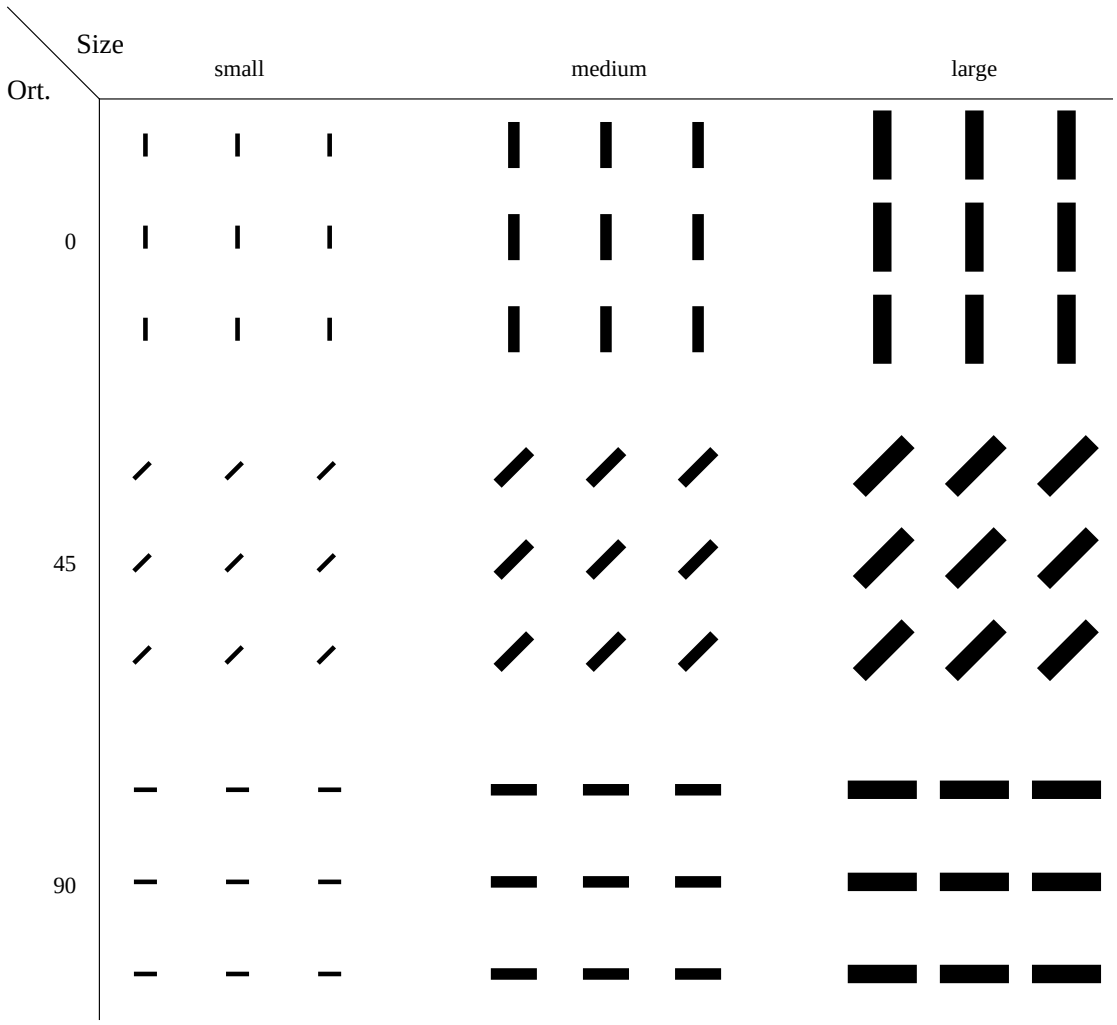


Figure 9.1: Examples of texture icons created by varying the orientation and size of the bars that make up each icon; notice that size and density (or contrast) are not independent of one another; decreasing the size of the bars also decreases the density of the texture icon they produce, and vise-versa

Another area of research is the use of both colour and texture in a single display. The ability to do this effectively would dramatically increase the expressive power of our visualization techniques. A study of the fundamental dimensions of texture patterns could be combined with results on effective colour selection. Again, we would have to investigate potential interactions between hue and intensity and orientation, size, and contrast. If these features are combined properly, the end result would be a visualization technique capable of representing up to five independent attributes through the use of texture and colour.

9.2 Emergent Features

One of our original goals was to investigate methods for visualizing high-dimensional data elements. Two of our results address this problem. First, we showed that hue and orientation do not interfere with one another during preattentive estimation. This means both hue and orientation can be used to encode independent attributes; a user is able to perform rapid and accurate estimation using either feature. Moreover, results from Wolfe et al. [1992] and from our own colour selection experiments [Healey, 1996] suggest we can display up to three orientations or seven hues in a single display, while still allowing for rapid and accurate target identification (it would be useful to determine if these results extend to the estimation task).

We also showed how knowledge discovery algorithms can be used to reduce the number of attributes in a multidimensional dataset. Classification combines multiple attributes into a single classification value. The knowledge discovery algorithms identify significant attributes during classification. Users can choose to pursue or ignore these attributes when they visualize the resulting dataset.

Tools that support the visualization of multiple data dimensions must deal with a poten-

tial interaction between some or all of the features being used to represent the dimensions. Rather than trying to avoid this, we can sometimes control the interaction and use it to our advantage. For example, Pickett and Grinstein use texture to represent high-dimensional data; each dimension controls one aspect of the texture element displayed to the user. Another promising avenue of investigation involves emergent features. An emergent feature can be created by grouping several simpler shapes together [Pomerantz and Pristach, 1989]. The emergent feature cannot be predicted by examining the simpler shapes in isolation. For example, in Figure 2.11a, the target element cannot be detected preattentively. In Figure 2.11b we rotated one of the component elements to create a new target with an emergent feature, non-closure, that is easily detected.

As an example of the use of emergent features, consider the salmon migration simulations described in Section 3.1 (Salmon Migration Simulations). A careful choice of simple features will allow a target element or a region of similar data elements to be detected preattentively, thereby signalling a correlation of variables in the data. Figure 9.2 shows one possible example of this technique. In the salmon tracking simulations scientists search for salmon entering hot ocean regions. This correlation of hot and present combines to form an emergent feature, closure. Because background elements (hot-absent, cold-present, and cold-absent) do not have closure, the target salmon can be easily detected.

9.3 Practical Applications

We tested our visualization techniques in a practical application environment. This allowed us to observe their effectiveness at visualizing real-world data and performing common analysis tasks. We chose three representative visualization examples: estimating salmon migration results, colouring medical slices from computerized tomography scans, and deriving and

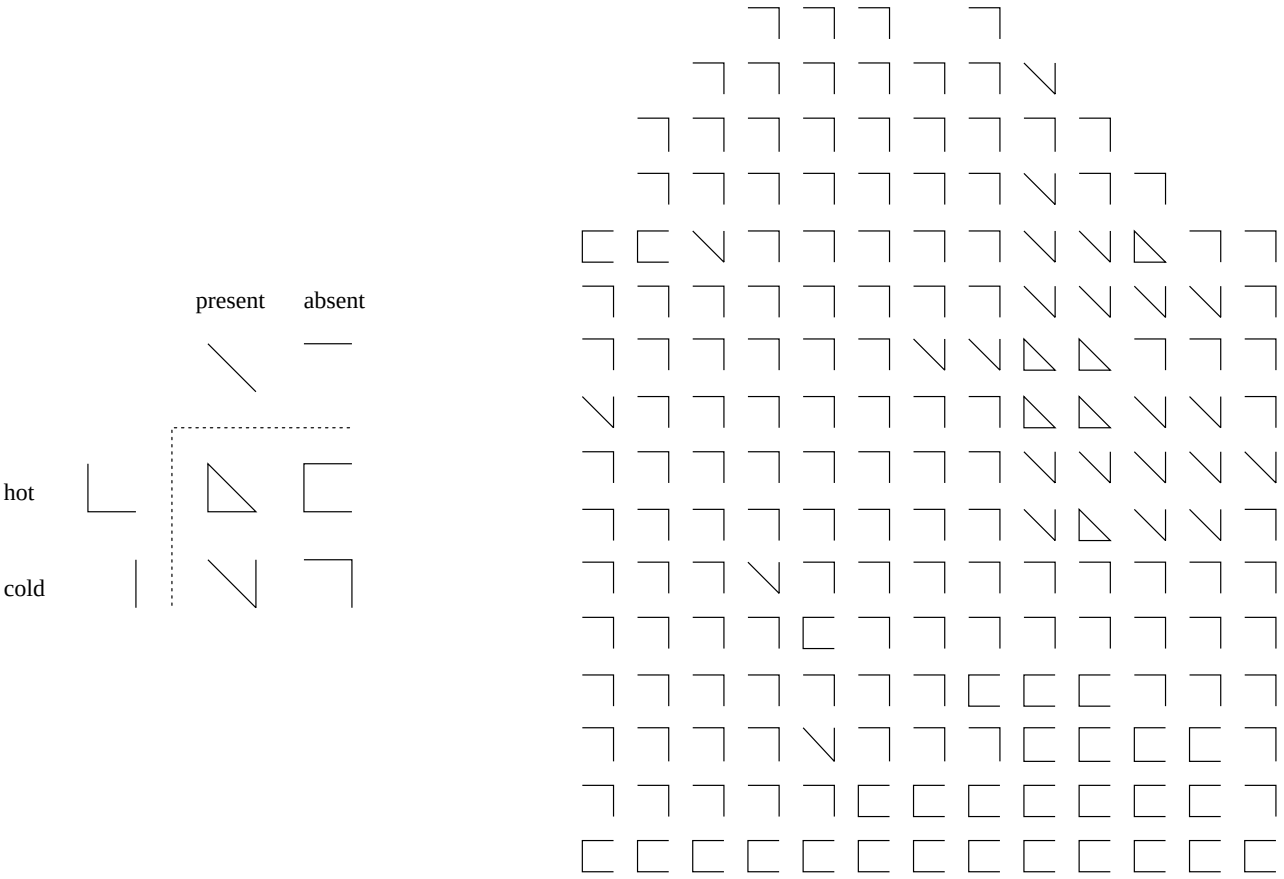


Figure 9.2: Example of output from salmon tracking simulations: salmon in hot ocean regions are displayed with an emergent feature closure; these can be detected because the three background objects do not use closure

tracking temporal ocean surface temperatures.

We designed and implemented a simple software tool called PV to perform our visualizations. PV does not to try to match the overall power and flexibility of commercial visualization systems like Iris Explorer or the Wavefront Data Visualizer. Rather, PV was built to implement results from our research that are unavailable in these visualization tools. For example, PV can impose constraints to guide the user during the data-feature mapping. Rules based on known limitations of visual features like hue, intensity, and orientation are built-in to PV. PV also provides direct access to the four knowledge discovery algorithms;

this allows users to investigate dependencies, filter their datasets, and add additional attributes through the use of training and classification. All of these features were used when we visualized our representative datasets.

Oceanography

The experimental displays we tested during our preattentive estimation experiments were based on data generated from salmon migration simulations being run in the Department of Oceanography at the University of British Columbia. The simulations were designed to test various hypotheses about how salmon migrate from the open ocean back to the B.C. Coast to spawn. A complete description of the simulations is provided in Section 3.3 (Salmon Migration Simulations) of this thesis. Results from the simulations were stored in a salmon migration dataset for later analysis.

The salmon migration dataset consisted of 8,004 elements distributed across 46 individual frames, representing simulation results for the years 1945 to 1990. Each element contained data for one simulated salmon. This included two independent attributes (the latitude and longitude where the simulated salmon started its migration run) and two dependent attributes (the stream function, a scalar value representing ocean current at the salmon's starting position, and the point of landfall, representing the location where the salmon arrived on the B.C. coast). Stream function had two possible values {`low, high`}. Point of landfall also had two possible values {`north, south`}. The dataset format and the data-feature mapping used to visualize the dataset are shown in Tables 9.1 and 9.2. Figure 3.3 shows an example of four different display frames.

Oceanographers scanned rapidly through the individual frames, looking for frames (each frame corresponds to migration data for a single year) where more than a fixed percentage of the of salmon had a landfall value of `north`. This corresponds to a preattentive estimation

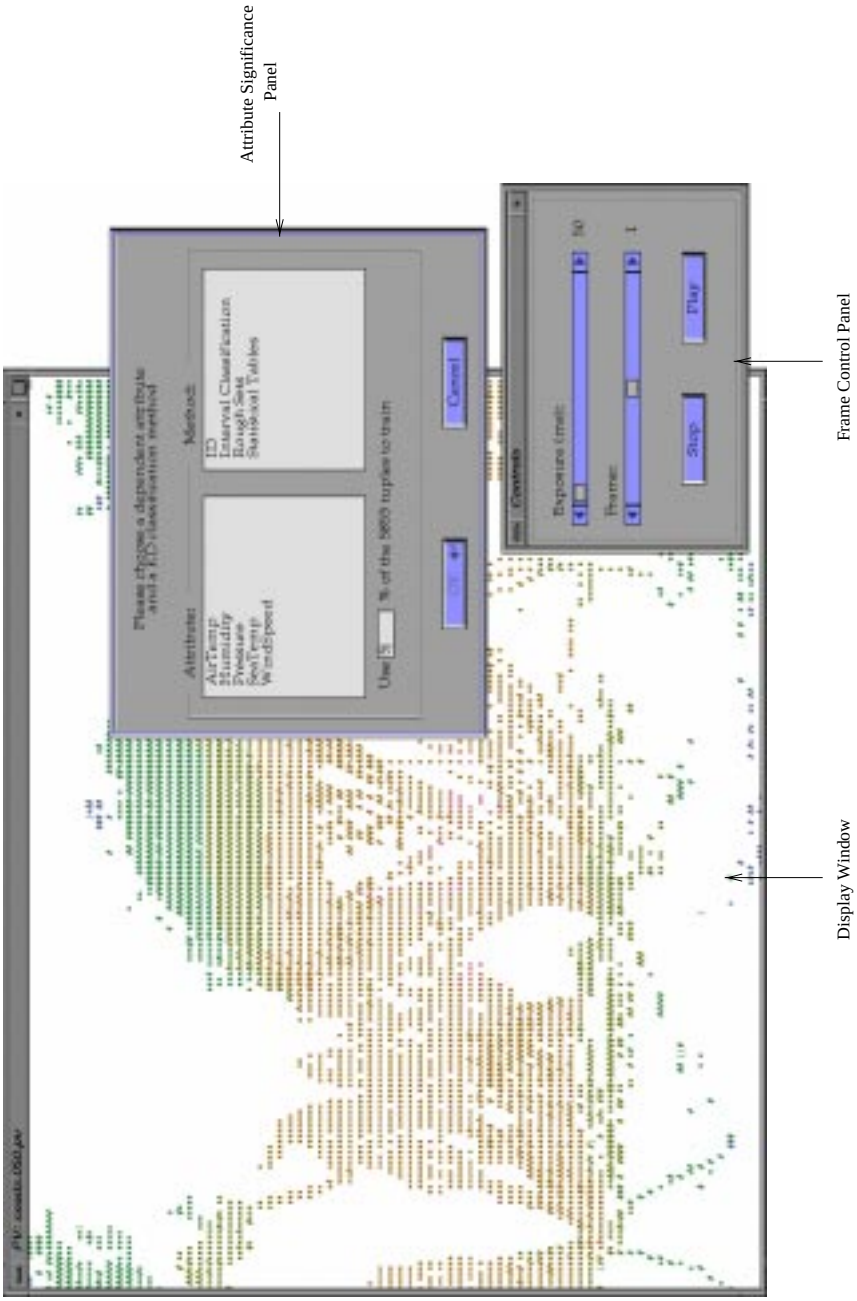


Figure 9.3: An example of the PV program being used to display sea surface temperatures for ocean positions on a map of the world; the figure shows the main display window, the dialog used to choose and animate frames, and the dialog used to compute significance weights for individual attributes

Size	8,004 elements, each element represents one simulated salmon
Frames	46 frames, each frame represents simulation results for one year from the range 1945 to 1990
Independent Attributes	latitude (continuous), $[\ 136^\circ E \ldots 158^\circ E\ ] \in \mathbb{Z}$ longitude (continuous), $[\ 45^\circ N \ldots 60^\circ N\ ] \in \mathbb{Z}$
Dependent Attributes	stream function (discrete), {`low, high`} latitude of landfall (discrete), {`north, south`}

Table 9.1: Format of Oceanography's salmon migration dataset, showing the size of the dataset in elements, the number of logical frames in the dataset, the independent attributes, and the dependent attributes

longitude	x-position of icon
latitude	y-position of icon
latitude of landfall	hue of icon, {`north` = red, `south` = blue}
stream function	orientation, {`low` = 0°, `high` = 60°}

Table 9.2: Data attribute to visual feature mapping for Oceanography's salmon migration dataset; longitude and latitude controlled the (x, y) position of a rectangular icon representing the simulated salmon, latitude of landfall controlled its colour, stream function controlled its orientation

task based on hue. Although this part of the analysis could be automated, the oceanographers often vary their cutoff percentage while viewing the displays. Years of interest are compared for the percentage and spatial location of `low` and `high` stream function values. This often involves estimation on various user-chosen subregions of the display. These subregions could not be pre-chosen for comparison without first identifying and examining the years of interest. Next, the oceanographers go back to the original data and search for displays with a stream function pattern similar to those found in the years of interest. Displays with a similar makeup of stream function values but a different percentage of point of landfall values must be explained in the context of the migration hypotheses. Finally, point of landfall values are compared statistically to historical salmon distributions provided by the Department of Fisheries and Oceans. This provides a computational measure of how well

the simulation (and the hypothesis on which it is based) "fits" actual salmon migration patterns. It would not be possible to perform this type of exploratory data analysis by simply pre-programming the computer to search for displays with a user-chosen range of point of landfall and stream function values.

Computerized Tomography Slices

An important part of our research focused on how to select effective colours for data visualization. Our goal was a set of colours that could be rapidly and accurately differentiated from one another. Results showed how we can measure and control colour distance, linear separation, and colour categories to pick up to seven isoluminant colours that satisfy our requirements. In order to test our technique, we used colours from our experiments to visualize slice data from a computerized tomography scan. The scan was used to search for the location of an aneurysm in a patient's brain. This resulted in 129 individual slices, each of which had a 512×512 pixel resolution.

Size	26,501,889 pixels, each pixel represents one CT sample point
Frames	129 frames, each frame represents one 512×512 CT slice (in each frame 205,441 of the 262,144 pixels represent actual data)
Independent Attributes	x-position (continuous), $[\ 0 \ldots 511\ ] \in \mathbb{Z}$ y-position (continuous), $[\ 0 \ldots 511\ ] \in \mathbb{Z}$
Dependent Attribute	material density (discrete), {`Level 1`, `Level 2`, `Level 3`, `Level 4`, `Level 5`, `Level 6`, `Level 7`}

Table 9.3: Format of CT dataset, showing the size of the dataset in pixels, the number of logical frames in the dataset, the independent attributes, and the dependent attribute

Each pixel in a CT slice contained two independent attributes (the x and y-position

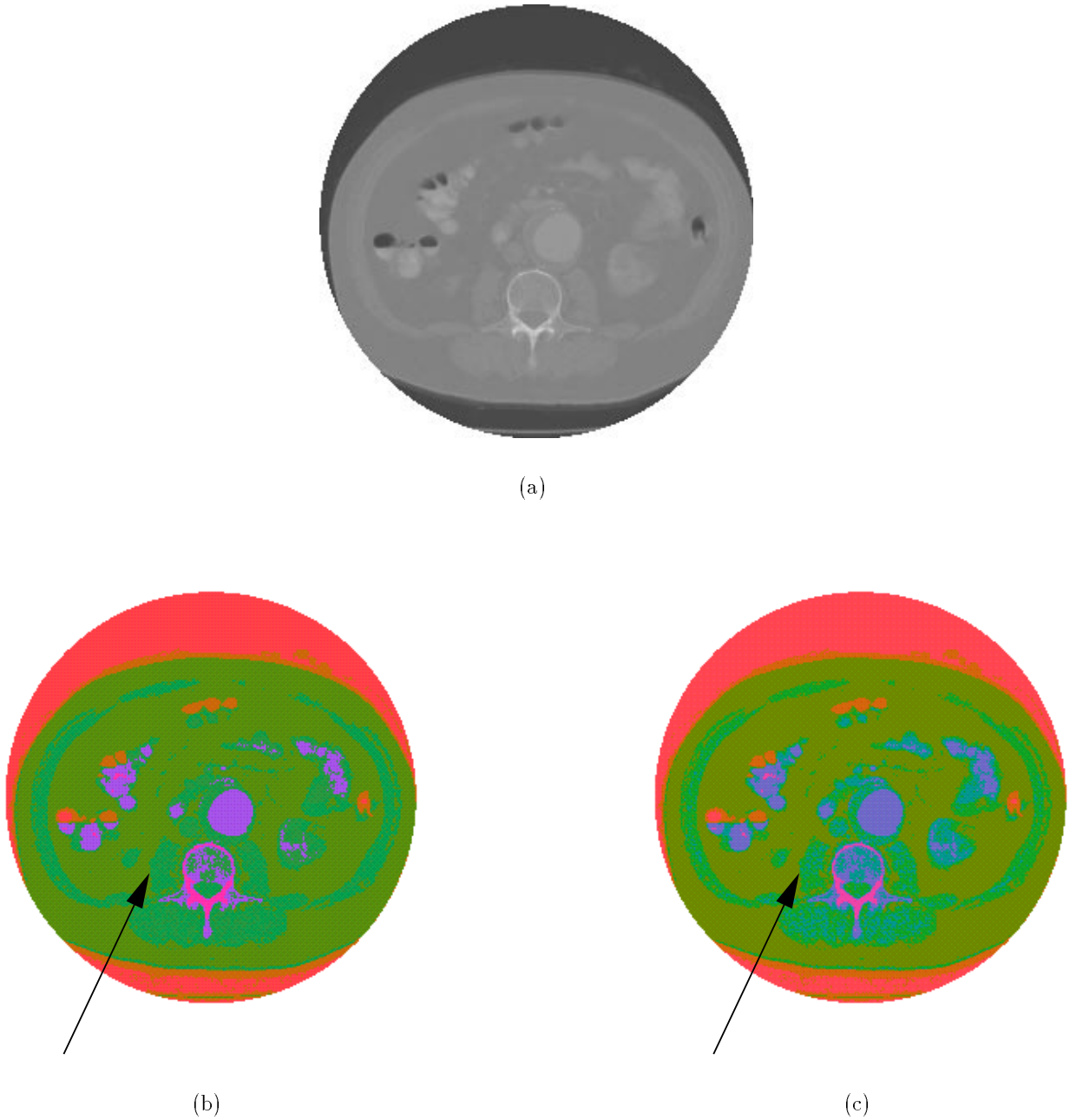
`Level 1`	$0 \leq density < 75$
`Level 2`	$75 \leq density < 121$
`Level 3`	$121 \leq density < 134$
`Level 4`	$134 \leq density < 139$
`Level 5`	$139 \leq density < 145$
`Level 6`	$145 \leq density < 161$
`Level 7`	$161 \leq density < 256$

Table 9.4: Intensity ranges used to segment the greyscale images into seven distinct regions; each region was coloured with a unique isoluminant colour

of the CT sample point), and one dependent attribute (material density). Material density initially ranged from 0 (lowest) to 255 (highest). This scale was chosen specifically to display the slices as 8-bit greyscale images (Figure 9.4a). Greyscale images are a standard method for viewing medical slice data [Ware, 1988; Bergman et al., 1995], because they allow a user to easily identify the location and shape of individual regions of similar intensity (*i.e.,* similarly density material). Through consultation we identified seven intensity ranges that corresponded to seven individual regions of interest (Table 9.4). Slices were then shown using seven different colours (*i.e.,* one for each intensity range). Figure 9.4b shows the slice in Figure 9.4a displayed using colours from our original seven-colour experiment. Figure 9.4c shows the slice displayed using colours from our final colour category experiment.

There are three important points to note. First, as with the greyscale representation, it is easy to locate individual regions in both colour slices. The densities (and therefore the intensity values) of different regions of interest are consistent across slices. Therefore, our colour displays work well for any individual slice chosen by the user.

Second, as predicted by our experiment results, the colours used in Figure 9.4b do not always provide good differentiation between regions compared to the colours used in Figure 9.4c. For example, consider the region referenced by the arrow. In Figure 9.4b, it appears



(a)

(b)

(c)

Figure 9.4: Examples of a single CT slice: (a) the slice displayed using a greyscale intensity ramp; (b) colours from the original seven-colour experiment, although the location and borders of each region are clearly visible, the makeup of certain regions is difficult to determine, compared to; (c) colours from the final seven-colour experiment, that show clearly that the region referenced by the arrow is made up of two different groups of elements

x-position	x-position of pixel
y-position	y-position of pixel
density	hue of pixel, {`Level 1` = Red, `Level 2` = Yellow, `Level 3` = Green-Yellow, `Level 4` = Green, `Level 5` = Blue-Green, `Level 6` = Purple, `Level 7` = Red-Purple}

(a)

x-position	x-position of pixel
y-position	y-position of pixel
density	hue of pixel, {`Level 1` = Red, `Level 2` = Yellow-Red, `Level 3` = Yellow, `Level 4` = Green, `Level 5` = Blue-Green, `Level 6` = Purple-Blue, `Level 7` = Red-Purple}

(b)

Table 9.5: Data attribute to visual feature mapping for the Computerized Tomography dataset; (x, y) location controlled the (x, y) position of the pixel representing the CT sample point; (a) density controlled the pixel's colour, colours were chosen from the original seven-colour study; (b) colours were chosen from the final colour category study (see Table 5.5 for a list of the RGB triples used to display each colour)

to made up of a single type of element. In fact, it is made up of two types of elements; this is clear when examining the same region in Figure 9.4c. Notice there is no corresponding tradeoff between Figure 9.4c and Figure 9.4b, that is, there are no pairs of elements that are difficult to differentiate in Figure 9.4c but easy to differentiate in Figure 9.4b. A large G-GY overlap suggested there would be a low perceived difference between the two colours used to represent the region of interest in Figure 9.4b. This overlap was explicitly removed by choosing a single colour from the G-GY region during the final colour category experiment. The result is a set of seven colours that clearly mark the locations and boundaries of all seven regions of interest.

Finally, colour images can provide visual cues that might not be immediately apparent in a greyscale display. For example, the region referenced by the arrow in Figure 9.4c is

obviously made up of two different types of elements. This is not as clear in the greyscale image, because the intensity ranges for the two regions are narrow (values [134 ... 138] and [139 ... 145], respectively) and adjacent to one another. Moreover, colour images often highlight spatial locations where large differences in value occur. For example, consider the small spots of purple at the top of the region in the center of the image. These spots represent areas of high relative density, and are quickly identified as different from the surrounding material. The same information is present in the greyscale image, but the visual system is not immediately drawn to it, because the contrast between the region and its neighbours is not as pronounced. One method of visualizing slice data is to "fly" through the slice stack, rapidly displaying individual slices one after another in a movie-like fashion. In this context, we expect users could more accurately detect areas of large difference using colour rather than greyscale.

Sea Surface Temperatures

NASA has recently made available a dataset that contains environmental conditions for sea and ocean locations throughout the world (see `http://podaac-www.jpl.nasa.gov/mcsst/` for more details). The dataset has been named the Comprehensive Ocean-Atmosphere Data Set (COADS). Each element in the dataset represents a $2° \times 2°$ sea or ocean region. An element contains (up to) eight observed and eleven derived attributes. Attribute means and standard deviations have been collected for each element for each month from January 1980 to December 1993.

We concentrated on five of the available attributes when we examined COADS: air temperature, sea level pressure, scalar wind, relative humidity, and sea surface temperature (Table 9.6). This meant each element contained two independent attributes (the latitude and longitude of the center of the ocean position represented by the element) and five inde-

pendent attributes.

Size	964,064 elements, each element represents a $2° \times 2°$ sea surface region
Frames	168 frames, each frame represents mean data for one month from the range January 1980 through December 1993
Independent Attributes	longitude (continuous), [$1° E \ldots 359° E$] $\in \mathbb{Z}$ latitude (continuous), [$89° S \ldots 89° N$] $\in \mathbb{Z}$
Dependent Attributes	air temperature (discrete) , {`Air 1`, `Air 2`, `Air 3`, `Air 4`, `Air 5`} sea level pressure (discrete) , {`SLP 1`, `SLP 2`, `SLP 3`, `SLP 4`, `SLP 5`} scalar wind (discrete) , {`WSp 1`, `WSp 2`, `WSp 3`, `WSp 4`, `WSp 5`} relative humidity (discrete) , {`Hum 1`, `Hum 2`, `Hum 3`, `Hum 4`, `Hum 5`}

Table 9.6: Format of COADS dataset, showing the size of the dataset in elements, the number of logical frames in the dataset, the independent attributes, and the dependent attributes

Sea surface temperatures (SSTs) were available in a separate file, but on a much coarser grid of $10° \times 10°$ (in fact, an SST dataset sampled on a $2° \times 2°$ grid was also available; we used the coarser resolution to test our knowledge discovery algorithms). Normally, SSTs are interpolated to provide values at 2° steps. One problem with this technique is that fine details that occur between sample points may be "smoothed out" and lost during interpolation. We decided to use our knowledge discovery algorithms to add fine-grained SST values to our original dataset. Continuous SST values were discretized into one of five possible values (Table 9.9). Air temperature, sea level pressure, wind speed, and relative humidity values from each frame in the original dataset were obtained at 10° intervals. The corresponding SST value was read and attached directly to the sample points. The result was used as a training set to build rules for mapping (*air, slp, wind, hum*) tuples to an SST value.

SST 1	-6° C	$\leq$	*sst*	$<$	0° C
SST 2	0° C	$\leq$	*sst*	$<$	10° C
SST 3	10° C	$\leq$	*sst*	$<$	20° C
SST 4	20° C	$\leq$	*sst*	$<$	30° C
SST 5	30° C	$\leq$	*sst*	$<$	36° C

WSp 1	0m/s	$\leq$	*wsp*	$<$	8m/s
WSp 2	8m/s	$\leq$	*wsp*	$<$	16m/s
WSp 3	16m/s	$\leq$	*wsp*	$<$	23m/s
WSp 4	23m/s	$\leq$	*wsp*	$<$	31m/s
WSp 5	31m/s	$\leq$	*wsp*	$<$	33m/s

SLP 1	954mb	$\leq$	*slp*	$<$	973mb
SLP 2	973mb	$\leq$	*slp*	$<$	993mb
SLP 3	993mb	$\leq$	*slp*	$<$	1013mb
SLP 4	1013mb	$\leq$	*slp*	$<$	1033mb
SLP 5	1033mb	$\leq$	*slp*	$<$	1049mb

Table 9.9: Sea surface temperature, sea level pressure, and scalar wind value ranges used to discretize the attributes into five individual values

All of the KDD algorithms reported air temperature and humidity, in that order, as the most significant attributes. Because we had available the exact $2° \times 2°$ SST values, we were able to compute classification error rates from interval classification, ID3, statistical tables, and a standard bilinear interpolation.

Table 9.10 shows that interval classification and ID3 gave better results than bilinear interpolation (both in terms of mean SST error $\overline{x}$, and in terms of the percentage of misclassified tuples). We wanted to see whether classification weights could be used to identify and ignore potentially erroneous tuples. To do this, we sorted tuples according to classification weight and ignored the lowest 5%, assuming they had been given an incorrect classification value.

Table 9.11 shows that mean SST error $\overline{x}$, standard deviation of SST error $\sigma^2(x)$, and

Method	$\overline{x}$	$\sigma^2(x)$	% error
Interp.	0.122	0.107	12.3%
ID3	0.107	0.310	10.7%
IC	0.095	0.293	9.5%
Stat.	0.236	0.607	17.1%

Table 9.10: Mean SST error $\overline{x}$, standard deviation of SST error $\sigma^2(x)$, and the percentage of incorrect classification values for bilinear interpolation, ID3, interval classification, and statistical tables

the percentage of misclassified tuples were all reduced when we filtered our results based on classification weights. The final two columns in the table show a breakdown of the number of correctly classified tuples (Good Tuples) and the number of incorrectly classified tuples (Bad Tuples) which were ignored. A low classification weight does not necessarily mean the classification value is incorrect. It simply means strong evidence did not exist in the training set to choose one particular classification value for the given tuple. Obviously, we want to minimize the number of correctly classified tuples which are mistakenly ignored during the filtering operation. This percentage was highest for Interval Classification (at 68% of the total number of tuples ignored) and lowest for ID3 (at 14% of the total number of tuples ignored).

Method	$\overline{x}$	$\sigma^2(x)$	% error	Good Tuples Ignored	Bad Tuples Ignored
ID3	0.082	0.274	8.2%	32	190
IC	0.082	0.274	8.2%	222	105
Stat.	0.226	0.598	16.3%	197	98

Table 9.11: Results when tuples with classification weights from the bottom 5% were ignored: mean SST error $\overline{x}$, standard deviation of SST error $\sigma^2(x)$, and the percentage of incorrect classification values were all lower than when no filtering was performed; the final two columns denote the number of correctly and incorrectly classified tuples which were ignored

We chose the ID3 algorithm to generate SST values, in part because it gave consistently

good performance when we tested our KDD algorithms, and in part because it ignored the fewest correctly classified tuples (tuples which are ignored correspond to elements with no SST value; these elements are incomplete, and are removed from the dataset to be visualized). Because the KDD algorithms reported that air temperature and humidity were closely matched to sea temperature, we did not consider these attributes when we visualized the COADS data. Sea level pressure and scalar wind were stored as continuous variables in the original dataset; we discretized each range to produce a five-valued attribute for use during visualization (Table 9.9). Figure 9.5 shows one frame from the classified dataset, with SSTs mapped to hue and wind speed mapped to orientation. Figure 9.6 shows a close-up of the Pacific and Atlantic Ocean around the coast of North America. SSTs were represented using five colours that we named red, yellow, green-yellow, blue, and purple (exact RGB values are listed in the 5-colour section of Table 5.5). Wind speeds were split into orientations of 0° (low wind speed), 30°, 45°, and 60° (medium wind speeds), and 90° (high wind speed).

The five colours used to represent SSTs were chosen using our colour selection technique. This guaranteed that each colour could be rapidly and accurately detected, even in the presence of all the others. Orientations were divided into shallow (low wind speed), tilted (medium wind speeds), and steep (high wind speed) based on Wolfe's orientation category results. User could search rapidly for outlier elements, yet at the same time determine individual values using focused attention.

Our visualization tool PV was used to scan forwards and backwards through stacks of individual frames, where each frame represented data from COADS for a particular month. Choosing colours that were easy to differentiate allowed a user to track the boundaries of different sea surface temperature regions. Target elements (*i.e.,* hot or cold sea surface locations) could also be rapidly identified. Moving through the data month-by-month allowed users to follow seasonal environmental changes. The data could also be organized to show a specific month for each of the 14 available years. This allowed users to observe year-by-year

Figure 9.5: An example display frame from COADS; sea temperature is mapped to hue and wind speed is mapped to orientation

Figure 9.6: A close-up of the Pacific and Atlantic Ocean regions around the coast of North America; sea temperature is mapped to hue and wind speed is mapped to orientation

differences across a much longer time period.

To summarize, our three visualization environments show practical examples of all of the visualization techniques described in this thesis. The salmon migration simulations used preattentive estimation (Chapter 3). Visualizing the CT medical image slices used both colour selection and real-time visualization (Chapters 5 and 6). Finally, the COADS visualization used colour selection, real-time visualization, and knowledge discovery (Chapters 5, 6, and 8). This demonstrates that, even in their current form, our techniques can be applied to real-world data and tasks.

Bibliography

Agrawal, R., Ghosh, S., Imielinski, T., Iyer, B., and Swami, A. (1992). An interval classifier for database mining applications. In *Proceedings 18th Very Large Database (VLDB) Conference*, 560–573.

Bauer, B., Jolicoeur, P., and Cowan, W. B. (1996). Visual search for colour targets that are or are not linearly-separable from distractors. *Vision Research* (in press).

Beck, J., Prazdny, K., and Rosenfeld, A. (1983). A theory of textural segmentation. In *Human and Machine Vision*, Beck, J., Prazdny, K., and Rosenfeld, A., Eds., 1–39. Academic Press, New York, New York.

Bell, P. C. and O'Keefe, R. M. (1987). Visual interactive simulation—history, recent developments, and major issues. *Simulation 49*, 3, 109–116.

Bell, P. C. and O'Keefe, R. M. (1994). Visual interactive simulation: A methodological perspective. *Annals of Operations Research 53*, 321–342.

Bergman, L. D., Rogowitz, B. E., and Treinish, L. A. (1995). A rule-based tool for assisting colormap selection. In *Proceedings Visualization '95*, 118–125, Atlanta, Georgia.

Berk, T., Brownston, L., and Kaufman, A. (1982). A new colour-naming system for graphics languages. *IEEE Computer Graphics & Applications 2*, 3, 37–44.

Birren, F. (1969). *Munsell: A Grammar of Color*. Van Nostrand Reinhold Company, New York, New York.

Braddick, O. J. and Holliday, I. E. (1987). Serial search for targets defined by divergence or deformation of optic flow. *Perception 20*, 345–354.

Brown, J. M., Weisstein, N., and May, J. G. (1992). Visual search for simple volumetric shapes. *Perception & Psychophysics 51*, 1, 40–48.

Brown, M. D., Greenberg, D., Keeler, M., Smith, A. R., and Yaeger, L. (1988). The visualization roundtable. *Computers in Physics 2*, 3, 16–26.

Bruckner, L. A. (1978). On Chernoff faces. In *Graphical Representation of Multivariate Data*, Wang, P. C. C., Ed., 93–121. Academic Press, New York, New York.

Callaghan, T. C. (1984). Dimensional interaction of hue and brightness in preattentive field segregation. *Perception & Psychophysics 36*, 1, 25–34.

Callaghan, T. C. (1989). Interference and domination in texture segregation: Hue, geometric form, and line orientation. *Perception & Psychophysics 46*, 4, 299–311.

Callaghan, T. C. (1990). Interference and dominance in texture segregation. In *Visual Search*, Brogan, D., Ed., 81–87. Taylor & Francis, New York, New York.

Cambell, W. J., Short, Jr., N. M., and Treinish, L. A. (1989). Adding intelligence to scientific data management. *Computers in Physics 3*, 3, 26–32.

Carter, R. C. (1982). Visual search with color. *Journal of Experimental Psychology: Human Perception & Performance 8*, 1, 127–136.

Carter, R. C. and Carter, E. C. (1981). Colour and conspicuousness. *Journal of the Optical Society of Aperica 71*, 723–729.

Chan, K. C. C. and Wong, A. K. C. (1991). A statistical technique for extracting classificatory knowledge from databases. In *Knowledge Discovery in Databases*, Piatetsky-Shapiro, G. and Frawley, W. J., Eds., 107–123. AAAI Press/MIT Press, Menlo Park, California.

Chernoff, H. (1973). The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association 68*, 342, 361–367.

Chiu, D. K. Y., Wong, A. K. C., and Cheung, B. (1991). Information discovery through hierarchical maximum entropy discritization and synthesis. In *Knowledge Discovery in Databases*, Piatetsky-Shapiro, G. and Frawley, W. J., Eds., 125–140. AAAI Press/MIT Press, Menlo Park, California.

CIE (1976). *CIE Publication No. 15, Supplement Number 2 (E-1.3.1): Official Recommendations on Uniform Color Spaces, Color-Difference Equations, and Metric Color Terms.* Commission Internationale de L'Èclairge.

Collins, B. M. (1993). Data visualization—has it all been seen before? In *Animation and Scientific Visualization*, Earnshaw, R. and Watson, D., Eds., 3–28. Academic Press, New York, New York.

Driver, J., McLeod, P., and Dienes, Z. (1992). Motion coherence and conjunction search: Implications for guided search theory. *Perception & Psychophysics 51*, 1, 79–85.

Duncan, J. (1989). Boundary conditions on parallel search in human vision. *Perception 18*, 457–469.

Duncan, J. and Humphreys, G. W. (1989). Visual search and stimulus similarity. *Psychological Review 96*, 3, 433–458.

D'Zmura, M. (1991). Color in visual search. *Vision Research 31*, 6, 951–966.

Enns, J. T. (1990a). The promise of finding effective geometric codes. In *Proceedings Visualization '90*, 389–390, San Francisco, California.

Enns, J. T. (1990b). Three-dimensional features that pop out in visual search. In *Visual Search*, Brogan, D., Ed., 37–45. Taylor & Francis, New York, New York.

Enns, J. T. and Rensink, R. A. (1990a). Influence of scene-based properties on visual search. *Science 247*, 721–723.

Enns, J. T. and Rensink, R. A. (1990b). Sensitivity to three-dimensional orientation in visual search. *Psychology Science 1*, 5, 323–326.

Enns, J. T. and Rensink, R. A. (1991). VSearch Colour: Full-colour visual search experiments on the Macintosh II. *Behavior Research Methods, Instruments, & Computers 23*, 2, 265–272.

Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. (1991). Knowledge discovery in database: An overview. In *Knowledge Discovery in Databases*, Piatetsky-Shapiro, G. and Frawley, W. J., Eds., 1–27. AAAI Press/MIT Press, Menlo Park, California.

Grinstein, G., Pickett, R., and Williams, M. (1989). EXVIS: An exploratory data visualization environment. In *Proceedings Graphics Interface '89*, 254–261, London, Canada.

Healey, C. G. (1996). Choosing effective colours for data visualization. In *Visualization '96*, (accepted), San Francisco, California.

Healey, C. G., Booth, K. S., and Enns, J. T. (1993). Harnessing preattentive processes for multivariate data visualization. In *Proceedings Graphics Interface '93*, 107–117, Toronto, Canada.

Healey, C. G., Booth, K. S., and Enns, J. T. (1996). High-speed visual estimation using preattentive processing. *ACM Transactions on Computer-Human Interaction* (in press, scheduled for July 1996 issue).

Hibbard, B. and Santek, D. (1990). The VIS-5D system for easy interactive visualization. In *Proceedings Visualization '90*, 28–35, San Francisco, California.

Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology 4*, 11–26.

Hurrion, R. D. (1980). An interactive visual simulation system for industrial management. *European Journal of Operations Research 5*, 86–93.

Ingraham, W. J. and Miyahara, R. K. (1988). Ocean surface current simulations in the North Pacific Ocean and Bearing Sea (OSCURS numerical model). Technical Report NMFS F/NWC-130, National Oceanic and Atmospheric Association.

Ingraham, W. J. and Miyahara, R. K. (1989). OSCURS numerical model to ocean surface current measurements in the Gulf of Alaska. Technical Report NMFS F/NWC-168, National Oceanic and Atmospheric Association.

Julész, B. (1971). *Foundations of Cyclopean Perception.* University of Chicago Press, Chicago, Illinois.

Julész, B. (1981). Textons, the elements of texture perception, and their interactions. *Nature 290*, 91–97.

Julész, B. (1984). A brief outline of the texton theory of human vision. *Trends in Neuroscience 7*, 2, 41–45.

Julész, B. and Bergen, J. R. (1983). Textons, the fundamental elements in preattentive vision and perception of textures. *The Bell System Technical Journal 62*, 6, 1619–1645.

Kaufman, A. (1986). Computer artist's colour naming system. *The Visual Computer 2*, 4, 255–260.

Kawai, M., Uchikawa, K., and Ujike, H. (1995). Influence of color category on visual search. In *Annual Meeting of the Association for Research in Vision and Ophthalmology*, #2991, Fort Lauderdale, Florida.

Kochevar, P., Ahmed, A., Shade, J., and Sharp, C. (1993). Bridging the gap between visualization and data management: A simple visualization management system. In *Proceedings Visualization '93*, 94–101, San Jose, California.

Levkowitz, H. and Herman, G. T. (1992). Color scales for image data. *IEEE Computer Graphics & Applications 12*, 1, 72–80.

Li, R. and Robertson, P. K. (1995). Towards perceptual control of Markov random field textures. In *Perceptual Issues in Visualization*, Grinstein, G. and Levkowitz, H., Eds., 83–94. Springer-Verlag, New York, New York.

Liu, F. and Picard, R. W. (1994). Periodicity, directionality, and randomness: Wold features for perceptual pattern recognition. In *Proceedings 12th International Conference on Pattern Recognition*, 1–5, Jerusalem, Israel.

McCormick, B. H., DeFanti, T. A., and Brown, M. D. (1987). Visualization in scientific computing—a synopsis. *IEEE Computer Graphics & Applications 7*, 7, 61–70.

Miyahara, M. and Yoshida, Y. (1988). Mathematical transform of (R,G,B) colour data to Munsell (H,V,C) colour data. In *Visual Communications and Image Processing '88*, 650–657. SPIE.

Müller, H. J., Humphreys, G. W., Quinlan, P. T., and Riddoch, M. J. (1990). Combined-feature coding in the form domain. In *Visual Search*, Brogan, D., Ed., 47–55. Taylor & Francis, New York, New York.

Nagy, A. L. and Sanchez, R. R. (1990). Critical color differences determined with a visual search task. *Journal of the Optical Society of America A 7*, 7, 1209–1217.

Nagy, A. L. and Sanchez, R. R. (1992). Chromaticity and luminance as coding dimensions in visual search. *Human Factors 34*, 5, 601–614.

Nakayama, K. and Silverman, G. H. (1986). Serial and parallel processing of visual feature conjunctions. *Nature 320*, 264–265.

NBS (1976). *Color: Universal Language and Dictionary of Names.* National Bureau of Standards, special publication 440.

Nothdurft, H.-C. (1985a). Orientation sensitivity and texture segmentation in patterns with different line orientations. *Vision Research 25*, 551–560.

Nothdurft, H.-C. (1985b). Sensitivity for structure gradient in texture discrimination tasks. *Vision Research 25*, 1957–1968.

Nothdurft, H.-C. (1991). Texture segmentation and pop-out from orientation contrast. *Vision Research 31*, 6, 1073–1078.

NOTHDURFT, H.-C. (1993). The role of features in preattentive vision: Comparison of orientation, motion, and color cues. *Vision Research 33*, 14, 1937–1958.

OKABE, A., BOOTS, B., AND SUGIHARA, K. (1992). *Spatial Tesselalations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., New York, New York.

PAULAK, Z. (1991). *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, Boston, Massachusetts.

PICKETT, R. AND GRINSTEIN, G. (1988). Iconographic displays for visualizing multidimensional data. In *Proceedings of the 1988 IEEE Conference on Systems, Man, and Cybernetics*, 514–519, Beijing and Shenyang, China.

POMERANTZ, J. AND PRISTACH, E. A. (1989). Emergent features, attention, and perceptual glue in visual form perception. *Journal of Experimental Psychology: Human Perception & Performance 15*, 4, 635–649.

QUINLAN, J. R. (1986). Induction of decision trees. *Machine Learning 1*, 1, 81–106.

QUINLAN, P. T. AND HUMPHREYS, G. W. (1987). Visual search for targets defined by combinations of color, shape, and size: An examination of task constraints on feature and conjunction searches. *Perception & Psychophysics 41*, 5, 455–472.

RHEINGANS, P. AND TEBBS, B. (1990). A tool for dynamic explorations of color mappings. *Computer Graphics 24*, 2, 145–146.

ROBERTSON, P. K. (1988). Visualizing color gamuts: A user interface for the effective use of perceptual color spaces in data displays. *IEEE Computer Graphics & Applications 8*, 5, 50–64.

ROGOWITZ, B. E. AND TREINISH, L. A. (1993). An architecture for rule-based visualization. In *Proceedings Visualization '93*, 236–243, San Jose, California.

ROSENBLUM, L. J. (1994). Research issues in scientific visualization. *IEEE Computer Graphics & Applications 14*, 2, 61–85.

SILBERSHATZ, A., STONEBRAKER, M., AND ULLMAN, J. D. (1990). The "Lagunita" report of the NSF invitational workshop on the future of database systems research. Technical Report TR-90-22, Department of Computer Science, University of Austin at Texas.

STONEBRAKER, M., CHEN, J., NATHAN, N., PAXSON, C., SU, A., AND WU, J. (1993). Tioga: A database-oriented visualization tool. In *Proceedings Visualization '93*, 86–93, San Jose, California.

Thomson, K. A., Ingraham, W. J., Healey, M. C., LeBlond, P. H., Groot, C., and Healey, C. G. (1992). The influence of ocean currents on the latitude of landfall and migration speed of sockeye salmon returning to the Fraser River. *Fisheries Oceanography 1*, 2, 163–179.

Thomson, K. A., Ingraham, W. J., Healey, M. C., LeBlond, P. H., Groot, C., and Healey, C. G. (1994). Computer simulations of the influence of ocean currents on Fraser River sockeye salmon (*oncorhynchus nerka*) return times. *Canadian Journal of Fisheries and Aquatic Sciences 51*, 2, 441–449.

Tominaga, S. (1985). A colour-naming method for computer color vision. In *Proceedings of the 1985 IEEE International Conference on Cybernetics and Society*, 573–577, Tucson, Arizona.

Treinish, L. A. (1993). Unifying principles of data management for scientific visualization. In *Animation and Scientific Visualization*, Earnshaw, R. and Watson, D., Eds., 141–170. Academic Press, New York, New York.

Treinish, L. A., Foley, J. D., Campbell, W. J., Haber, R. B., and Gurwitz, R. F. (1989). Effective software systems for scientific data visualization. *Computer Graphics 23*, 5, 111–136.

Treinish, L. A. and Goettsche, T. (1991). Correlative visualization techniques for multidimensional data. *IBM Journal of Research and Development 35*, 1/2, 184–204.

Trick, L. and Pylyshyn, Z. (1994). Why are small and large numbers enumerated differently? A limited capacity preattentive stage in vision. *Psychology Review 101*, 80–102.

Triesman, A. (1985). Preattentive processing in vision. *Computer Vision, Graphics and Image Processing 31*, 156–177.

Triesman, A. (1991). Search, similarity, and integration of features between and within dimensions. *Journal of Experimental Psychology: Human Perception & Performance 17*, 3, 652–676.

Triesman, A. and Gormican, S. (1988). Feature analysis in early vision: Evidence from search asymmetries. *Psychological Review 95*, 1, 15–48.

Triesman, A. and Souther, J. (1986). Illusory words: The roles of attention and top-down constraints in conjoining letters to form words. *Journal of Experimental Psychology: Human Perception & Performance 14*, 107–141.

TUFTE, E. R. (1983). *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Connecticut.

TUFTE, E. R. (1990). *Envisioning Information.* Graphics Press, Cheshire, Connecticut.

UPSON, C. (1989). The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics & Applications 9*, 4, 30–42.

VANDE WETTERING, M. (1990). apE 2.0. *Pixel 1*, 4, 30–35.

VAREY, C. A., MELLERS, B. A., AND BIRNBAUM, M. H. (1990). Judgments of proportions. *Journal of Experimental Psychology: Human Perception & Performance 16*, 3, 613–625.

WARE, C. (1988). Color sequences for univariate maps: Theory, experiments, and principles. *IEEE Computer Graphics & Applications 8*, 5, 41–49.

WARE, C. AND BEATTY, J. C. (1988). Using colour dimensions to display data dimensions. *Human Factors 30*, 2, 127–142.

WARE, C. AND KNIGHT, W. (1992). Orderable dimensions of visual texture for data display: Orientation, size, and contrast. In *Proceedings SIGCHI '92*, 203–209, Monterey, California.

WARE, C. AND KNIGHT, W. (1995). Using visual texture for information display. *ACM Transactions on Graphics 14*, 1, 3–20.

WOLFE, J. M. (1994). Guided Search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review 1*, 2, 202–238.

WOLFE, J. M. AND CAVE, K. R. (1989). Deploying visual attention: The Guided Search model. In *AI and the Eye*, Troscianko, T. and Blake, A., Eds., 79–103. John Wiley & Sons, Inc., Chichester, United Kingdom.

WOLFE, J. M., CAVE, K. R., AND FRANZEL, S. L. (1989). Guided Search: An alternative to the feature integration model for visual search. *Journal of Experimental Psychology: Human Perception & Performance 15*, 3, 419–433.

WOLFE, J. M. AND FRANZEL, S. L. (1988). Binocularity and visual search. *Perception & Psychophysics 44*, 81–93.

WOLFE, J. M., FRIEDMAN-HILL, S. R., STEWART, M. I., AND O'CONNELL, K. M. (1992). The role of categorization in visual search for orientation. *Journal of Experimental Psychology: Human Perception & Performance 18*, 1, 34–49.

Wolfe, J. M., Yu, K. P., Stewart, M. I., Shorter, A. D., Friedman-Hill, S. R., and Cave, K. R. (1990). Limitations on the parallel guidance of visual search: Color $\times$ color and orientation $\times$ orientation conjunctions. *Journal of Experimental Psychology: Human Perception & Performance 16*, 4, 879–892.

Wong, A. K. C. and Chiu, D. K. Y. (1987). Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 6, 796–805.

Wyszecki, G. and Stiles, W. S. (1982). *Color Science: Concepts and Methods, Quantitative Data and Formulae, 2nd Edition.* John Wiley & Sons, Inc., New York, New York.

Ziarko, W. (1991). The discovery, analysis, and representation of data dependencies in databases. In *Knowledge Discovery in Databases*, Piatetsky-Shapiro, G. and Frawley, W. J., Eds., 195–209. AAAI Press/MIT Press, Menlo Park, California.