Tools to enrich user experience during "Visual Analysis"

A Dissertation Presented

by

Supriya Garg

to

The Graduate School

in Partial Fulfillment of the

Requirements

For the Degree of

Doctor of Philosophy

in

Computer Science

(Visualization)

Stony Brook University

December 2010

Stony Brook University

The Graduate School

Supriya Garg

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Klaus Mueller – Dissertation Advisor,
Associate Professor, Computer Science

I.V. Ramakrishnan – Chairperson of Defense,
Professor, Computer Science

Tamara Berg,
Assistant Professor, Computer Science

Kevin T. McDonnell,
Associate Professor, Computer Science, Dowling College

This dissertation is accepted by the Graduate School

Lawrence Martin

Dean of Graduate School

Abstract of the Dissertation

Tools to enrich user experience during "Visual Analysis"

by

Supriya Garg

Doctor of Philosophy

in

Computer Science

(Visualization)

Stony Brook University

2010

Visual Analytics is the integration of interactive visualization with analysis techniques to help answer questions, and find interesting “patterns” in a given dataset. This approach is useful in cases where human knowledge and intuition is required. Further, with the increasing size of data, an interactive visual interface can give the user control over what she wants to see. Towards this end, we develop a Visual Analytics (VA) infrastructure, rooted on techniques in machine learning and logic-based deductive reasoning. This system assists people in making sense of large, complex data sets by facilitating the generation and validation of models representing relationships in the data.

The above data is often communicated among people, and by systems as graphs and other visual forms. However, currently the degree of semantics encoded in these representations is quite limited. In order to design more expressive icons, we present a framework that uses natural language text processing and global image databases to help users identify metaphors suitable to visually encode abstract semantic concepts. While using images as forms of communication, people often like to add annotations in forms of text, arrows, etc. to make them more expressive. The process of selecting a color for the annotation can often be difficult if the background has a variation in color and texture. Our tool *Magic marker* helps users by suggesting appropriate colors for annotating an image. All colors in a modified CIE-Lab color-space are assigned a preference value, and users can navigate this space using an interactive interface. This interactivity also allows them to make the final choice based on personal preferences.

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

People use visual analytics tools and techniques to synthesize information and derive insight from massive, dynamic, ambiguous, and often conflicting data; detect the expected and discover the unexpected; provide timely, defensible, and understandable assessments; and communicate assessment effectively for action.

Visual analytics is a multidisciplinary field that includes the following focus areas [92]:

1) *analytical reasoning techniques* that let users obtain deep insights that directly support assessment, planning, and decision making;
2) *visual representations and interaction techniques* that exploit the human eye's broad bandwidth pathway into the mind to let users see, explore, and understand large amounts of information simultaneously;
3) *data representations and transformations* that convert all types of conflicting and dynamic data in ways that support visualization and analysis; and
4) techniques to support *production, presentation, and dissemination* of analytical results to communicate information in the appropriate context to a variety of audiences.

The focus in current Visual Analytics (VA) systems is to tease out relationships in a given dataset. Interaction with the system helps users discover interesting trends and patterns in the data. The user is expected to do the major work in the analysis, including hypothesis formulation, testing, and validation. The system usually assists by providing interactive visual displays, with related information linked together [22], with multiple coordinated views [84], or with domain specific visual interfaces [76]. The user can often save the data views for his discoveries, including the path which led to the finding. However, there is no real formal learning going on, just an understanding in the analyst's mind for a particular data scenario. The model that evolves as a by-product is lost.

We want to make the process of visual analytics more rewarding by using the immense capacity of human mind to detect patterns and passing on the task of generalization and rule formation to the machine. Machines can do a better job of learning formal models, since they are not limited by the size of working memory, and can consider the entire dataset before forming any rules or hypothesis. In our framework the hard work is saved for later re-use and extension. However, learning is hard − both for humans and machines. A human with some knowledge of the given dataset can easily mark new examples for new or existing models. The machine on the other hand can use these examples as guidelines to generalize and learn the most efficient model.

We have the same overall approach to all our problems. First, the user explores the data space to discover interesting and useful patterns. Next, these marked examples are used in further learning – either via generalization using machine learning, or via distilling the essence of the examples to come up with a representative item for the category. After this initial stage of learning, the user is put back in the loop to refine the results by specifying further input to refine the results. Once the user is satisfied with the results he sees, these results are stored in the most general way possible, so as to help other users in the future.

In Chapter 2, we accomplish this by building an infrastructure based on machine learning and logic-based reasoning. The system assists analysts make sense of large, complex data sets by facilitating the generation and validation of models representing relationships in the data. While the user can explore the high dimensional dataset using an interactive visual interface and provide the system with interesting patterns, the system handles the analysis and rule generation, and returns the results back. The user can now refine the rule by providing further examples, hence becoming a part of the well known sense-making loop [70].

In Chapter 3, we extend the above idea to develop a system where the user helps the machine learn classification/segmentation models by providing his input via a user interface. The main idea behind this work is that tokens belonging to the same class(es) will have similar feature vectors, and cluster together well. Hence, we can present the tokens to the users in a node link diagram, instead of one document at a time. The user has control over both the clustering parameters (how fine should the clustering be), and the relative weights of the various feature-vector components. Fine tuning these parameters helps users provide a good initialization for the classification model. In the later stages, the user is required to debug this model by looking at the most *uncertainly* classified documents. This means that the burden on the users is lowered, and they only need to individually tag documents which are difficult to classify, possibly due to ambiguity, or lack of evidence (e.g. new tokens not seen by the model before). In this fashion we provide a user interface for both initializing the model, and then refining it later on the principles of *Active Learning.*

Human beings can detect patterns immensely better with visual representations than with textual ones, which in turn enables more complex reasoning processes − this makes visualization a powerful means to support analytical reasoning. However, the amount of information encoded in these visual representations is still quite limited. The majority of visual communication occurs by ways of spatial groupings, plots, graphs, data renderings, photographs and video frames. The use of icons as a form of information encoding has been explored to a much lesser extent. In Chapter 4 we describe a framework that uses a dual domain approach involving natural language text processing and global image databases to help users identify metaphors suitable to visually encode abstract semantic concepts. The dual domain approach is especially useful, since semantic tools like WordNet can help us in the case of ambiguity, either when words are polysemous (multiple meanings), or when the names of brands clash with the actual word (e.g. Apple). The visual representations resulting from these visual encodings can then be used as stand-alone expressive icons or as clip art in visualization and visual analytics applications. The main idea behind this framework is that given sufficient user studies, we can build a database of these visual metaphors, and people can use it similar to a clipart program with personalization capabilities.

While using images and icons to communicate ideas and information, people often like to add embellishments by adding text, arrows, etc. In the case of a busy background, choosing an appropriate color for such embellishments or annotations is not an easy task, and most involves several rounds of trial and error. To help users in this process, we present a system in Chapter 5 suggests appropriate colors for annotations. We capture different findings in the field of legibility and readability into our system, and let the user make the final choice since aesthetics and personal preference plays a large role on what is deemed as *legible.* We use a modified version of the perceptual color space CIE-LAB as a basis of our calculations and visualizations. Each point in this space is assigned a preference value where colors with higher values are better choices.

All the tools we present here can help make the process of visual analysis more efficient and productive. Expressive visual encodings means that the same screen space can effectively convey a lot more information. Further, humans can use their strongest strengths – intuition and world knowledge to identify patterns in any dataset, leaving the task of *learning* to the computer.

Chapter 2. Model-Driven Visual Analytics

This chapter describes a Visual Analytics (VA) infrastructure, rooted in techniques in machine learning and logic-based deductive reasoning that can assist analysts to make sense of large, complex data sets by facilitating the generation and validation of models representing relationships in the data. We use Logic Programming (LP) as the underlying computing machinery to encode the relations as rules and facts and compute with them. A unique aspect of our approach is that the LP rules are automatically learned, using Inductive Logic Programming, from examples of data that the analyst deems interesting when viewing the data in the high-dimensional visualization interface. Using this system, analysts will be able to construct models of arbitrary relationships in the data, explore the data for scenarios that fit the model, refine the model if necessary, and query the model to automatically analyze incoming (future) data exhibiting the encoded relationships. In other words it will support both model-driven data exploration, as well as data-driven model evolution. More importantly, by basing the construction of models on techniques from machine learning and logic-based deduction, the VA process will be both flexible in terms of modeling arbitrary, user-driven relationships in the data as well as readily scale across different data domains.

2.1 Introduction

Modern day enterprises, be they commerce, government, science, engineering or medical, have to cope with voluminous amounts of data. Effective decision making based on large, dynamic datasets with many parameters requires a conceptual high-level understanding of the data. Acquiring such an understanding is a difficult problem, especially in the presence of incomplete, inconsistent, and noisy data acquired from disparate real-world sources. To make progress on this problem one must draw on the complementary strengths of computing machinery and human insight.

Recognizing this promising human-computer synergy, Visual Analytics (VA), defined as the science of analytical reasoning facilitated by interactive visual interfaces

[92], has become a major development thrust. It seeks to engage the fast visual circuitry of the human brain to quickly find relations in complex data, trigger creative thoughts, and use these elements to steer the underlying computational analysis processes towards the extraction of new information for further insight. VA has widespread applications, such as homeland security, the financial industry and internet security among others. Research papers and tools related to VA are beginning to emerge [3,90,99].

However, so far the main emphasis in VA has been mostly on visualization, data management, and user interfaces (see e.g. [12] and other work mentioned in the following). As far as analytical computing goes, VA research has mainly focused on relatively low-level tasks, such as image [101] and video [54] analysis and database operations [91]. In today's VA systems, it is the human analyst who performs the actual reasoning and abstraction. Obviously this type of workflow is of limited scalability in terms of reasoning chain complexity. While there has been recent promising work on explicit knowledge management [100], this system has been mainly designed to help users shape and keep track of emerging insight, but not to derive higher-level models from the data. In this chapter, we present a framework that allows human analysts to off-load some of the reasoning to a computational analyst. Collaboration with this machine agent occurs by pointing out interesting data patterns discovered in the visual interface.

The VA infrastructure we describe is rooted in techniques in *machine learning* and *logic-based deductive reasoning*. They assist analysts in making sense of large data sets by facilitating the generation and validation of models representing relationships in the data. By basing the construction of models on techniques from machine learning and logic-based deduction, the VA process will both be flexible in terms of modeling arbitrary, user-driven relationships in data and it will readily scale across different data domains.

Locating patterns in high-dimensional space via visual inspection can be challenging. The method of parallel coordinates (PC) [40], which has seen various refinements in recent years (see e.g., [77]), maps a high-dimensional point into a piece-wise linear line that spans the vertical dimension axes. With PC, clusters can be isolated by brushing the appropriate data axes. There are, however, certain limits on the spatial complexity of the clusters. Scatter plots, on the other hand, provide an intuitive way to observe possibly arbitrarily complex relationships. However, in most applications, scatter plots are confined to projections onto two data axes at a time, and in order to view higher-order relationships spanning more than two dimensions, the associated projections must be serialized, viewed as a matrix, or be stacked [47].

We seek a visual interface that allows users to point out arbitrary high-dimensional patterns in a more direct manner. An early high-dimensional exploration paradigm providing a combined scatter plot of an arbitrary number of variables is the Grand Tour (GT) [4], which is part of the GGobi package [89]. The user is guided through hyper-space along a trajectory that is decided along the way by selecting the maximum of a given projection pursuit metric. Projection pursuit is a statistical technique which involves finding the most *interesting* projections in multidimensional data. Projections which deviate more from a Normal distribution are considered to be more interesting. As each projection is found, the data is reduced by removing the component along that projection, and the process is repeated to find new projections; this is the *pursuit* aspect that motivated the technique known as matching pursuit.

The user stops when the current view point is the local maximum of this projection-pursuit metric. While traveling, the user views projections (in form of scatter plots) of the data onto a 2D hyper-plane. In GT the motion parallax resulting from the quick but continuous transition of hyper-plane projections provides a very compelling experience for getting a sense of the high-dimensional structure of the data. We present an autonomous high-dimensional navigation interface augmented with intuitive navigation aids that allows the user to fluently control the orientation of the projection hyper-plane, utilizing the resulting motion parallax to explore high-dimensional neighborhoods in an insightful manner.

This chapter is structured as follows. In Section 2.2, we discuss existing VA work relevant to ours. Section 2.3 provides technical preliminaries and gives an overview of our approach. In Section 2.4 we describe our model learning framework, while Section 2.5 describes our high-dimensional visual data navigation and pattern painting interface. Section 2.5 demonstrates the use of our VA framework for the learning of new models from user-suggested data examples within a few representative application scenarios.

Section 2.6 provides details on how to generalize our approach to a wider field of applications and ends with conclusions.

2.2 Related Work

Learning models from patterns is an active research topic in various branches of computer vision. But there the pattern examples in most cases originate directly from image analysis, promoting unsupervised learning where subtle anomalies (the unexpected data) or new families of patterns which the model parameters cannot capture often go undetected or are misunderstood. Visual analytics, on the other hand, aims to be more flexible in the data constellations encountered, appealing to the complex pattern recognition apparatus of humans and their intuition, creativity, and expert knowledge to point out unusual configurations for further testing and model refinement. Papers recognizing this potential have been sparse, but are currently emerging. Janoos et al. [42] used a visual analytics approach to learn models of pedestrian motion patterns from video surveillance data, in order to distinguish typical from unusual behavior in order to flag security breaches in outdoor environments. Their semi-supervised learning approach in which users interact with video stream data improves upon the standard unsupervised learning schemes that are typically used in these scenarios (see for example [56]) allowing personnel to better train the models used for anomaly detection. The patterns encountered are fairly straightforward to visualize: motion trajectories of pedestrians visualized as flow fields. In contrast, our system attempts to learn from high-dimensional data.

Xiao et al. [99] describe a visual analytics system for network traffic analysis. In their system, users manually assemble and construct declarative knowledge rules based on example patterns they mark in the visual interface. These rules are then available for the classification of incoming new network traffic data. However, the manual assembly of these declarative rules can be laborious, especially in the context of high-dimensional data. Our system automates this process by replacing the manual rule assembly by machine learning and logic-based reasoning, based on the patterns specified by the user.

2.3 Overview

An important aspect of high-level semantic understanding of data is the identification of entities that are present in the data and relationships among them. Computational Logic offers an elegant and powerful vehicle to facilitate this kind of understanding. In particular Logic Programming languages such as Prolog offer a platform to encode and compute with such relations. A logic program consists of a data base of facts representing background domain knowledge and logical relationships (encoded as rules) that describe the relationships which hold in the data. Rather than running a program to obtain a solution, the user poses queries and the run time system searches through the data base of facts and rules to determine (by logical deduction) the answer [53].

Inference rules in Prolog, are of the form 'Head :- Body', meaning "conclude Head if Body holds". Facts are rules with empty bodies. Below is a specification of a path in a graph:

```
Rule1: path(X,Y)  :- edge(X,Y).
Rule2: path(X,Y)  :- edge(X,Z),path(Z,Y).
Rule 3: edge(a, b). edge(b, c). edge(a, d). edge(c, d).
```

Rule 1 says there is a path from X to Y if there is an edge from X to Y. Rule 2 says that there is a path from X to Y if there is an edge from X to Z and a path from Z to Y. Note that X, Y and Z are variables which are instantiated with actual nodes in the graph. The terms path(X,Y), edge(X,Y) are called *predicates*. An instance of a graph is specified by the facts. In the program above the four facts edge(a, b), edge(b, c), edge(a, d) and edge(c, d) specify a graph with edges from a to b, b to c, a to d and c to d respectively. A query '?path(X, d)' to the program will return one answer 'true' with X bound to "a" using resolution-based deduction. One can also get two more answers with X bound to "b" and "c".

In our approach the relations in the data are encoded as logical rules in Prolog. They constitute the *model* of the relations. Rather than encoding these rules manually we learn them from examples of data using Inductive Logic Programming (ILP) methods [62]. Typically ILP methods use a top-down approach of refining a very general rule,

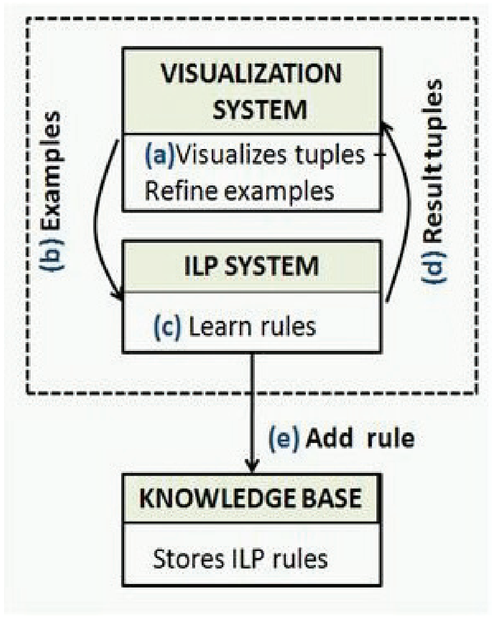


Figure 2.1: The overview of our system:
a) Visualize the dataset, b) Provide example, c) Learn rules, d) Return and display results, Steps b, c, d continue in a loop till the user is satisfied with the rule.
e) Add rule to the knowledge base.

where the refining process involves adding a body predicate one at a time with the purpose of "better fitting" the given training examples, i.e., eliminate negative examples while still maintaining the coverage of positive examples. The added predicates can be base predicates, previously learned predicates, or arithmetic constraints (such as $|X-Y| < \delta$). ILP methods have produced impressive practical results (see e.g., [62]) and are now well established as mainstream machine learning technology. The strength of ILP methods is that learning is done in the presence of prior background knowledge encoded as facts.

Figure 2.1 provides an overview of how our VA system works: the analyst starts off by visualizing the dataset in the visualization interface, identifies interesting patterns and provides them as examples to the ILP system, and waits for the results. Once the ILP system has learned the rules, the analyst can visualize the tuples returned by the system. These tuples correspond to the relation encoded by the learnt rule (e.g. path (a, b), path (a, c), path (a, d) in the path predicate above). A single loop of "visualize-supply examples-learn rules-visualize returned tuples" constitutes an iteration of the VA cycle. Based on the returned results in the next iteration the analyst can supply additional positive and/or negative examples so that the ILP system can refine the rules to correct the accuracy of the learnt relation. Finally, when the rules encompass the patterns deemed relevant by the analyst, the new rule is added to the knowledge base.

With the user actively providing the examples from which to learn, an effective visual interface is crucial. One of our demo applications is the learning of models from sensor-rich high-dimensional network traffic data. Locating patterns in this high-dimensional space via visual inspection can be challenging. In the most general case these patterns can form high-dimensional manifolds. Current techniques, for the most part, can only reveal clusters and patterns of rather regular topology, such as quadrics. As mentioned, scatter plots provide a powerful means to visualize irregular patterns, but most applications only provide scatter plots along axis dimensions. This limits abilities to discover joint relationships in high dimensions. Instead, we provide an interface in which analysts can mark interesting example patterns directly onto arbitrary projections of hyper space, navigated via an intuitive flight control interface. But high-dimensional projections are always ambiguous, and thus overall quality assessments need to be made. For this, we couple our N-D projection display with a spectral plot that is used to fine-tune the patterns before sending them to the learning engine. The same spectral plot is also utilized for the visualization of tuples retrieved from the data on queries over the learnt model, which allows further model refinement.

As mentioned, the Grand Tour allows users to explore the N-D space by projecting the N-D data into a general hyperplane. Such projections have the potential to reveal complex high-dimensional relationships that axis-wise projections may not. However, the curse of dimensionality makes self-guided tours a daunting exercise, and to provide some guidance the method of projection pursuit [25] was devised. It allows users to 'chase' pre-defined or ‘interesting’ patterns and cluster configurations. We describe here an interface that allows users to interactively navigate a given N-D subspace, using motion parallax to defuse the limitations of N-D to 2-D projections. This subspace may either be encountered during a projection pursuit-based guided exploration or during an unguided space exploration, initialized by human analyst intuition in our view-setup interface and refined using the flight controls.

2.4 Model Learning Framework

For our model learning framework, we use *Aleph* [1], a Prolog based ILP system. It can learn rules which are disjunctions of conjunctions like the path predicate shown earlier.

To learn rules in *Aleph,* we need background knowledge in the form of Prolog facts and rules, and some positive and negative examples corresponding to patterns that are in the relation and not in it respectively. Further, the program should know which background rules the new rule can depend on. Sometimes the user knows this and can restrict the rule's dependency to a subset of attributes. The user can add domain specific rules as well.

2.4.1. Rule Learning

The positive and negative examples are all provided through the visual interface. An example consists of either a single tuple, or a set of tuples. This is automatically converted into example files for *Aleph* which expects them to be in the form of Prolog facts as well. Note that when we say we pass tuples as examples, we only pass their unique indices in the implementation.

In our implementation, we modified some of the *Aleph* parameters to facilitate the handling of large data sets. These include:

- Clause length (*clauselength)*: This specifies the length of the rule, including the head (the rule name). We increased it from the default value of 5 to 10, to learn longer rules
- Evaluation function (*evalfn)*: It is an expression which determines the importance of a clause based on the number of positive and negative examples it satisfies. We set this to "pos" only in the first round of learning; this helps Aleph learn rules in the absence of negative examples.
- Number of nodes (*nodes)*: This denotes an upper bound on the number of nodes to explore while searching for the rule. We increased the default value of 5000 to 100,000.
- Layers of new variables (*i)*: This represents an upper bound on the layers of new variables. Each layer represents sets of variables occurring adjacent to the variables in

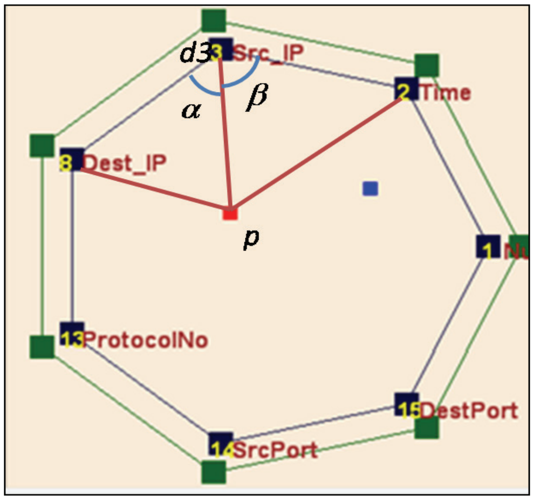


Figure 2.2: Computing weights of PPA-vectors using barycentric coordinates.

the previous layer. Layer 0 contains the variables in the head of the rule. The value of this parameter was increased from the default value of 1.

With the above changes to the default values, *Aleph* was better suited to handle large data sets.

Once the rules are generated, we need to return patterns satisfying these rules. To do so the rule are evaluated over the KB using Prolog-style evaluation to materialize the tuples of the relation and return them to the visualization system.

Observe that the learnt rules can generate patterns in new datasets in the same domain. In this manner, the models created by one analyst can be shared with other analysts.

2.5 Visual High-Dimensional Data Interaction Interface

In the following, we use the term 'Projection Plane' to denote a subspace of arbitrary dimensions d in an N-dimensional data space, where $2 \leq d \leq N$.

2.5.1. High Dimensional Visual Interface: Navigator

Figure 2.3 shows a screenshot of our interface. It is composed of (a) a scatter plot window, (b) a touch-pad like navigation interface that allows interactive control of the two orthogonal 'Projection Plane Axis vectors' (*PPA-vectors* for short), (c) a window that shows the projections of all data axes onto the current projection plane, and (d) a display that plots their N-D coordinates as a bar chart histogram. The user starts by choosing *d* dimensions of interest among the *N* dimensions available (this choice can be altered at any time). The touch pad interface will then be configured into a d-gon (an equilateral polygon with d vertices) for navigation (Figure 2.3 (b)), where each polygon vertex stands for one dimension axis. Similar to a mouse touch pad, the user may move the pointer (indicated by a small colored dot and addressed by the mouse) inside the polygon, which transforms the projection plane of the scatter plot within the *d*-dimensional sub-space. In fact, we provide two such pointers, one each to represent the *x*-axis (red) and the *y*-axis (blue) of the PPA-vectors. We found that in many cases only the dynamics of motion can reveal interesting high-dimensional patterns. This is already part of the general idea of the Grand Tour paradigm used in GGobi. However, in the Grand Tour, in conjunction with projection pursuit, users are mostly confined to watching motions until interesting projections appear. While users do have some level of interactive navigation control, they cannot change the projection plane orientations in arbitrary ways. Our interface provides significantly more freedom in that respect, but is most suited for expert users who are familiar with the space they are navigating in and can use informed intuition on the type of patterns and the location they seek to explore. Our direct navigation allows them to easily transform to these interesting projections and at the same time also gives them a sense of location and orientation – an important requirement for navigation tasks.

A smooth transition function for the projection plane is the key to intuitive observance of dynamic patterns in N-D. Further, we would like to have the ability to select a single dimension by simply moving over the polygon vertex it is associated with. Both these requirements can be achieved by interpolating the PPA-vectors via generalized barycentric coordinates [60], which calculate barycentric coordinates for irregular, convex n-sided polygons. This mechanism has found frequent use for the interpolation of high-dimensional properties (e.g. see [68]), and we extend its use here for the control of hyper-dimensional scatter plots. For a given PPA-vector, manipulated by the pointer in the touchpad d-gon, the barycentric coordinates determine the weight that

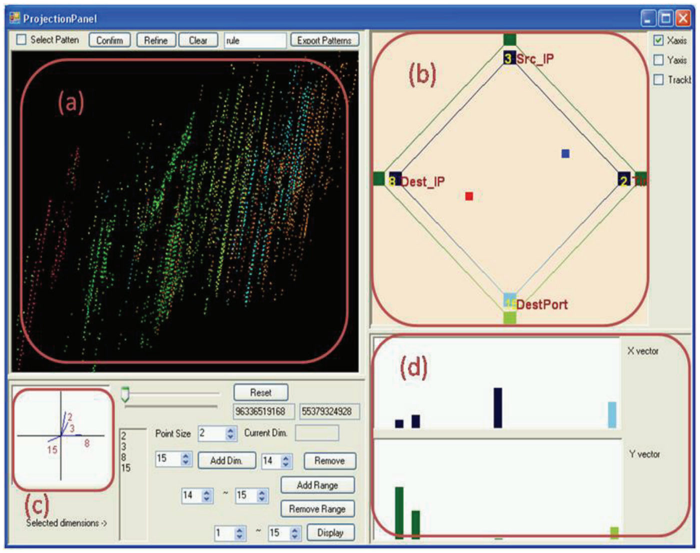


Figure 2.3: High Dimensional Navigation interface (a) Scatter plot (b) Touch Pad Navigator (c) Dimension axis display (DA-display) (d) Histogram display.

each of the dimensions has on this vector. The weight of dimension d3 (Src_IP) for the x-axis PPA-vector (the red point p) is (see Figure 2.2):

$$\overrightarrow{w_3} = \frac{\cot(\alpha) + \cot(\beta)}{\left\|\vec{p} - \overrightarrow{d_3}\right\|^2} \qquad (1)$$

After computing the weights for all d dimensions, they are first normalized and then form the components of the respective PPA-vector. In essence, by moving the point closer to a certain dimension, the dimension's weight will grow larger in the vector, and thus this 'attracts' that dimension axis to the PPA-vector. Each PPA-vector is displayed in the dimension histogram, to convey the prominent data dimensions in the current view to the user (Figure 2.3(d)). Finally, the data point positions in the scatter plot (Figure 2.3(a)) are given by the dot products between the PPA-vectors and the data points.

But even if users are permitted to freely move about space by choosing PPA-vectors via the touch pad interface, we must still enforce that the two PPA-vectors remain mutually orthogonal. To ensure this, for example, when moving the y-axis PPA-vector, we project this vector onto a vector y' that is closest to a vector orthogonal to the current x-axis PPA-vector (or vice versa):

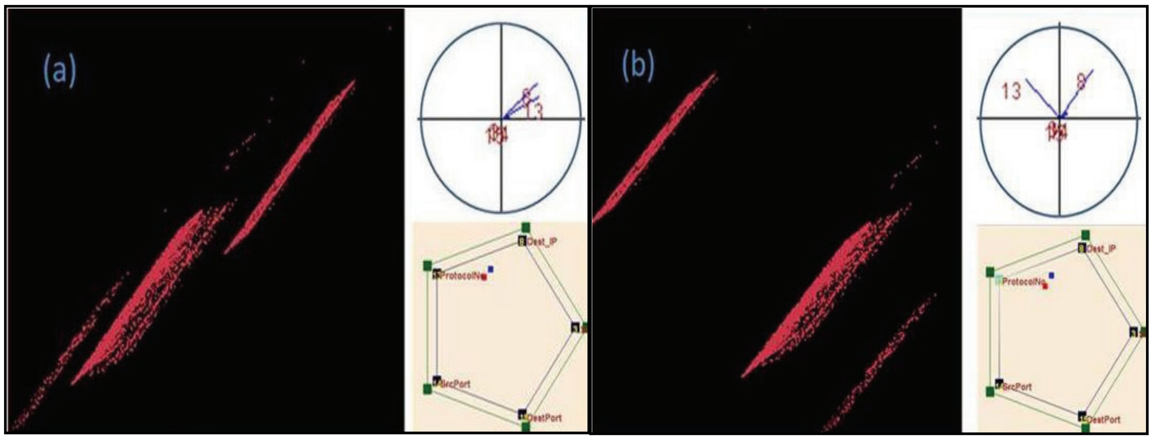


Figure 2.4: Two dimensions (8 and 13) which were close to each other (a) exhibit a broader gap in the projection plane by flipping the 13th dimension axis from positive to negative (b). Data points in the scatter plot will be spread out between the 8th and the 13th dimension. A negative flipping (negative weights) is conveyed on the touch pad navigator by coloring the axis poles with darker colors. (a) Navigating only with positive weights. (b) Flip the 13th dimension with respect to the y-axis of the projection plane.

$$y' = y - (x \bullet y).x \qquad (2)$$

Below the scatter plot, we visualize the orientation (dark shades correspond to positive orientation) and the weight that each dimension axis has in the scatterplot (see Figure 2.3(c)). This type of dimension axis display (or *DA-display*) is also part of GGobi. It intuitively shows which dimensions the scatter plot visually separates which can be judged by the axes spread in the DA-display. The user can increase the spread with the touchpad interface, by moving the mouse to reduce or increase the dimension’s weight in one of the projection plane vectors. Dimensions with a small presence in the scatter plot can be identified by their short axes in the axis plot.

We also provide a more radical means to separate the visual effect of two dimensions. Our interface allows users to flip one dimension’s weight to a selected PPA-vector. This mirrors the data along this dimension with respect to this PPA-vector only, and can yield the desired separation (see Figure 2.4). While in some cases this alone is still not sufficient (which is easily seen in the DA-display), further mouse-movements can eventually achieve the goal. The order of the vertices along the touch pad polygon

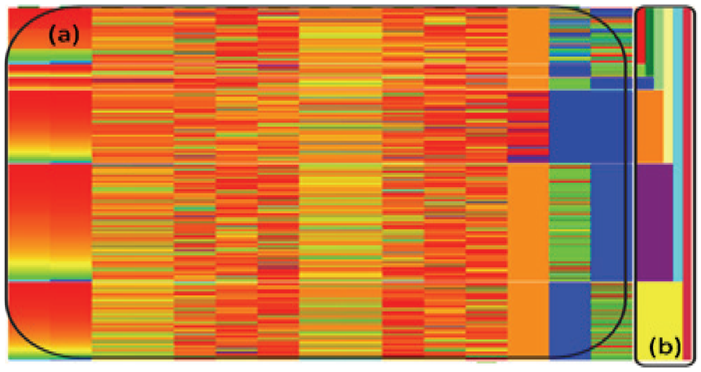


Figure 2.5: High Dimensional Visualization Interface (a) Spectrum Point Map (b) Tree-like Dendrogram.

strictly determines the sub-space that can be explored. Users can set up a new sub-space for visualizing other desired relationships among dimensions by manipulating the vectors in the DA-display. Here, orthogonal orientation of the two PPA-vectors is again enforced using a method similar to the one described above.

2.5.2. High Dimensional Visual Interface: Point Map

Even with the dynamic motion display, the scatter plot projections can still present ambiguities. Therefore, we also provide a spectral plot (called *point map*) which represents D-dimensional data completely undistorted and with no projection ambiguities (Figure 2.5(a)). The point map displays each data vector (the spectrum) as a horizontal line of colored pixels, one per axis direction. The color mapping is normalized from 0 to 1, and the color is mapped from blue to red. The point lines are grouped by the smallest cluster they reside in (the leaf nodes of the hierarchy). The dendrogram (Figure 2.5(b)) represents the current status of the cluster hierarchy. The user may double-click on any such cluster to make it the active cluster. The scatter plot will then display the information of this active cluster, and the color of the points in the scatter plot corresponds to the dendrogram cluster color. The interface also features weighted k-means clustering and sorting.

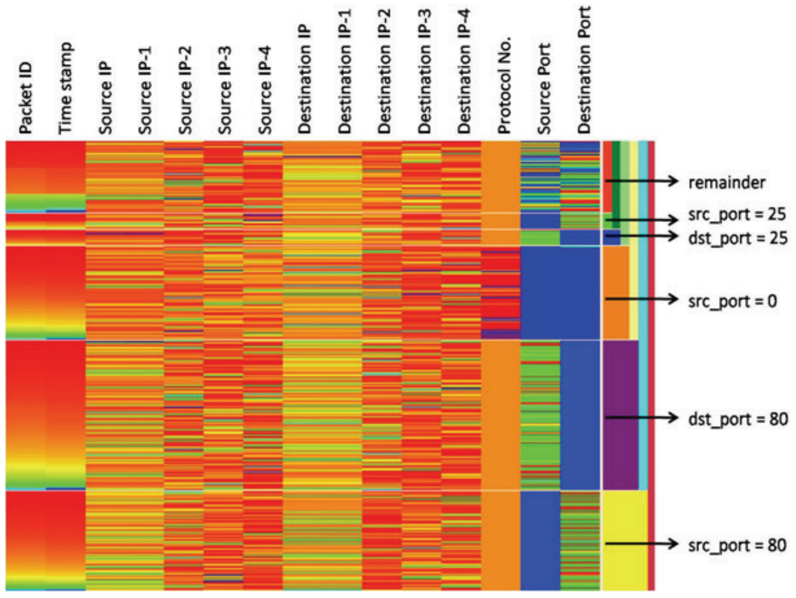


Figure 2.6: Color map showing the clusters formed after filtering based on port values.

2.5.3. Interaction with the ILP System: Pattern Painting

Upon discovering some interesting patterns in the N-D projection, the analyst may paint those patterns directly in the scatter plot, to be subsequently exported to the ILP. The selected patterns will be highlighted, after which the user moves further about in the high dimensional space to confirm that these patterns are indeed correct and not just due to projection effects. In the point map these sets can then be separated as different clusters. Since different clusters are represented by colors in both dendrogram and scatter plot, analysts can quickly gain insight into the selections and possibly refine the existing or even select additional patterns.

The selected pattern sets are exported to the ILP system. That in turn produces new sets of patterns from the data (by evaluating the rules) which can be visualized in a similar fashion. Imported pattern sets will be represented as clusters and can be viewed in different colors. Users can refine positive examples or indicate a pattern as a negative example, and then export these new sets of examples to the learning system.

2.6 Model-Driven VA in action

In this section we illustrate the operational aspects of our VA framework by building models of relations in two data sets, namely network traffic and census data.

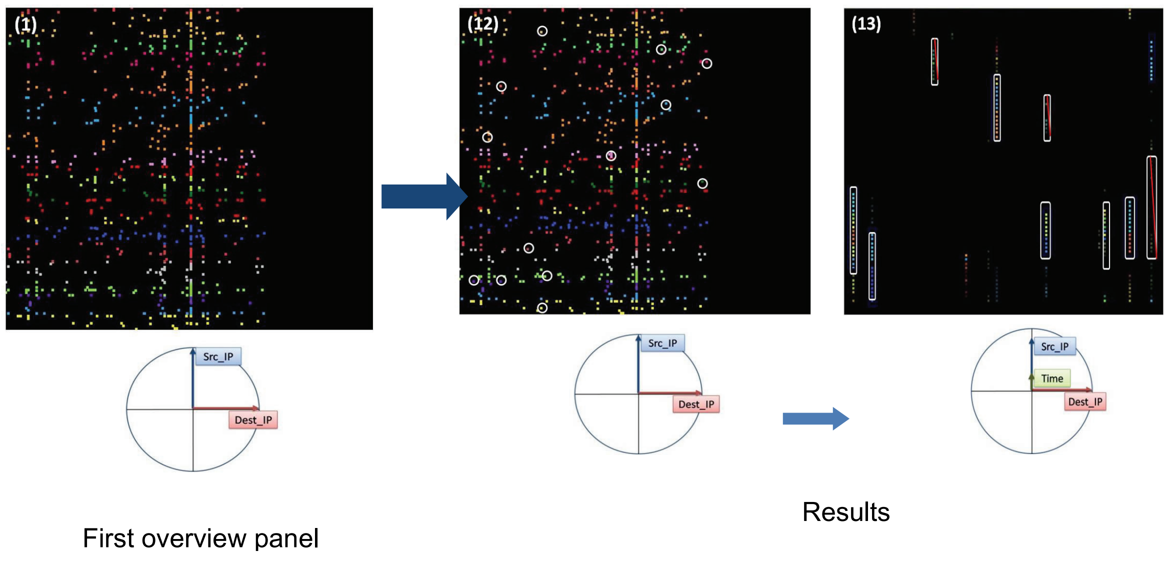


Figure 2.7: Panel 1 on the left shows the initial projection view during rule learning - source IP vs. destination IP; it shows the IPs which exchanged packets with each other. Panels 12 and 13 on the right show the final results returned by the learning program. In panel 13, we look only at the results, and select positive and negative examples.

2.6.1. Network Traffic

This dataset was obtained from the MAWI Working Group Traffic Archive . It represents traffic data captured over an hour, in tcpdump format. We extracted the required fields in a comma separated file, and preprocessed it in the format needed by our VA system. The fields extracted were packet number, source IP, destination IP, source port, destination port, protocol number, and time stamp. Any unknown fields were given the value of 0.

As the dataset is huge, to see an overall structure in it, we filtered it on basis of various port values (Figure 2.6). Here, the fields Source IP-I and Destination IP-I, where I=1, 2, 3, 4, represent the 8-bit partitions of the source and destination IP respectively. The filtering was done based on the source or destination port of the packets. As seen in Figure 2 6, it is done in the following order: (i) source port 80, (ii) destination port 80, (iii) source port 0 (unknown source port), (iv) destination port 25, and (v) source port 25. In this way, we characterized the network traffic at a coarse level. In the figure, the yellow and purple clusters at the bottom contain packets involving transfer on port 80 – the http port. Since they make up more than half of our dataset, we can easily infer that most of our traffic is http traffic. This means that a great number of webpage

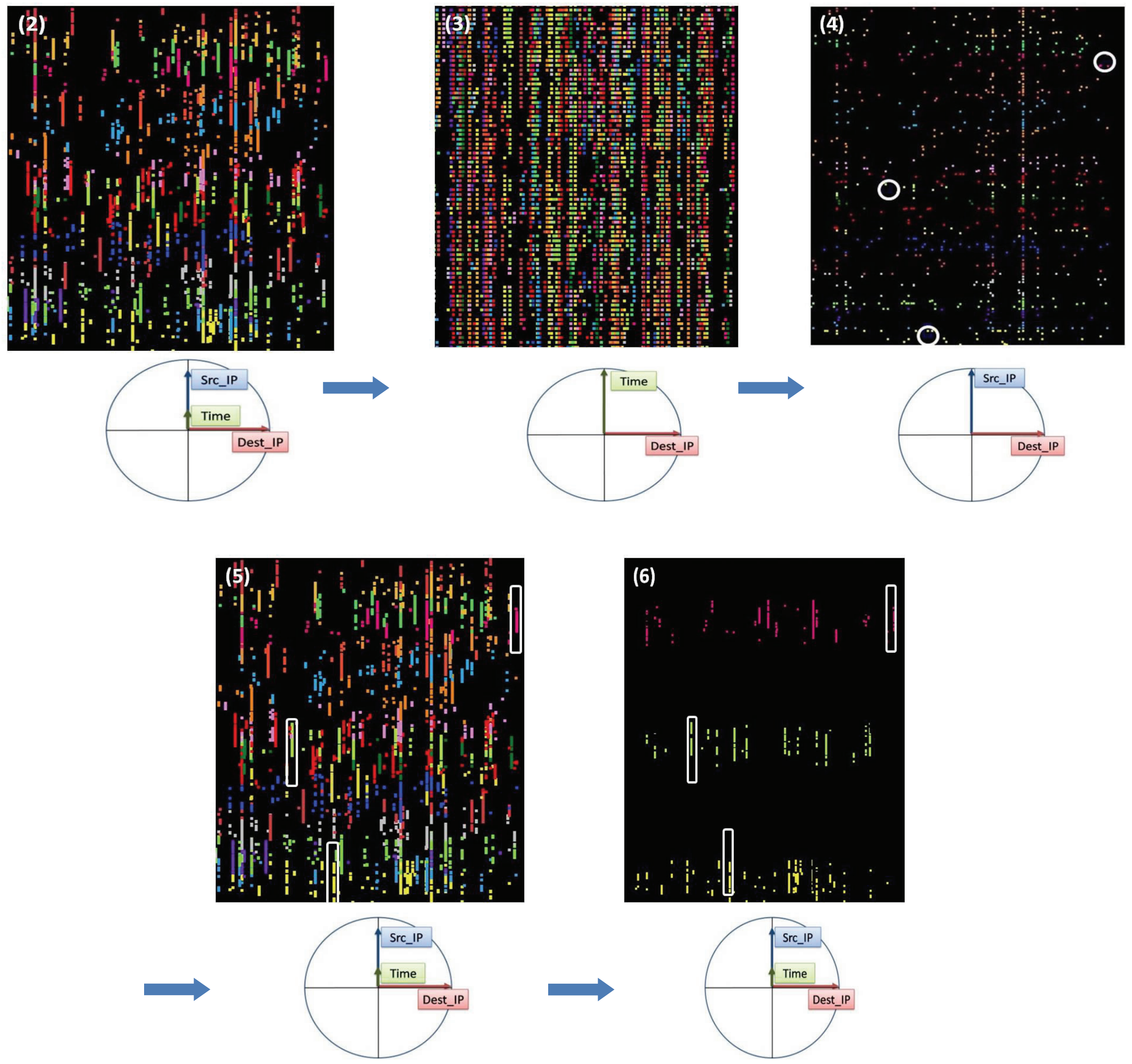
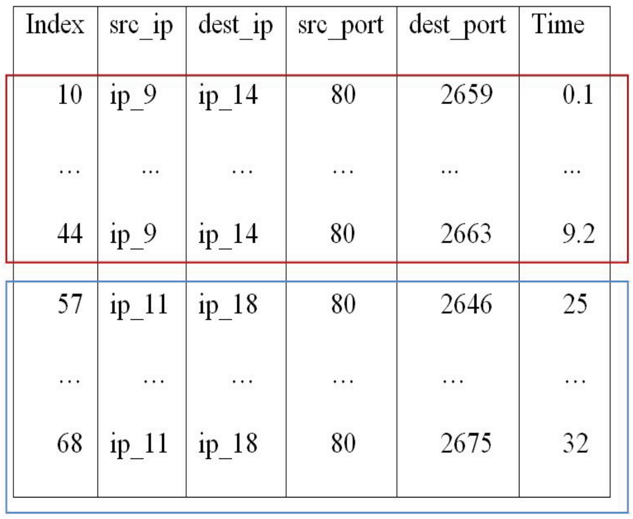


Figure 2.8: In panels 2 and 3, we move the Y-axis from source IP to time. We see the points spread in space showing that the corresponding source IP-destination IP pairs exchange multiple packets. In panels 4, 5 and 6 we select three points, and follow their path as the Y-axis is again changed. Due to the image getting cluttered, we select the clusters containing the points in panel 6.

requests/loads is happening. To find out which destinations IPs have many webpage loads, we select the cluster corresponding to source port 80 – the yellow cluster.

For further analysis, we shift our focus to the navigation on the high dimensional data space. Here, if all packets are displayed with the same color, it will be difficult to follow the patterns formed by the points sharing some attributes. For example the size of the packets exchanged between two IPs might be increasing with time. In this case, we want to follow the number of packets exchanged between two IPs in a short time range.

Table 2.1: Example of patterns selected to learn the rule webpage-load.



Index	src_ip	dest_ip	src_port	dest_port	Time
10	ip_9	ip_14	80	2659	0.1
…	...	…	…	...	...
44	ip_9	ip_14	80	2663	9.2
57	ip_11	ip_18	80	2646	25
…	…	…	…	…	…
68	ip_11	ip_18	80	2675	32

To address this, we cluster the dataset into 16 sets based on source IP. Adjacent clusters are assigned highly different colors, so that we can differentiate between points belonging to them easily.

We follow the rest of this scenario in Figures 2.7, 2.8 and 2.9, which contain 13 panels telling the story from the initial projection in panel 1, to the display and refining of results in panels 12 and 13. For an initial view, we project the packets into the domain of source IP against destination IP (Figure 2.7(1)), showing us the IPs which exchanged packets. To follow the exchange of packets between two IPs, we vary the Y-axis of our projection between source IP and time. Figure 2.8(2) and Figure 2.8(3) clearly show us a lot of points spreading, revealing one or even multiple webpage-loads during the motion.

In Figure 2.8(4) and Figure 2.8(5), we try to follow a few points, as the Y-axis is being changed. In Figure 2.8(4) and Figure 2.8(5), we try to follow a few points, as the Y-axis is being changed. However, since this scatterplot has too many points, we just select the three clusters which contain these points (Figure 2.8(6)). This step makes our task of selecting a few positive examples much easier.

In Figure 2.9(7)-(11), we look at a smaller subset of the data, hoping to follow patterns more easily. However, as we move the Y-axis between source IP and time, we notice that the yellow and green clusters clash (Figure 2.9(8) and Figure 2.9(9)), and being quite similar, they are difficult to differentiate. So we change the color of the

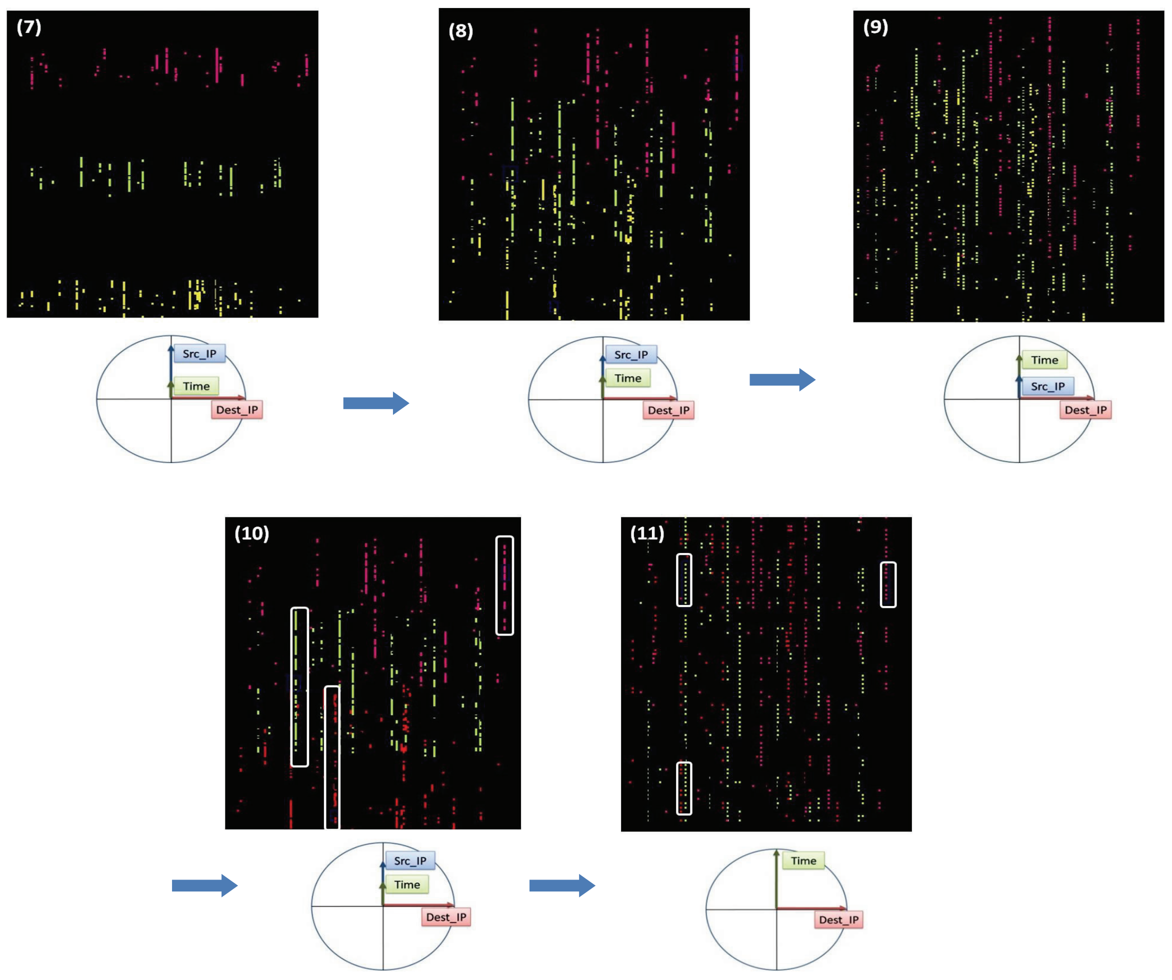


Figure 2.9: In the panels above, we are looking only at the three selected clusters. In panels 7-9, we move the Y-axis in a fashion similar to panels 2 and 3. Due to the yellow and green being very similar, it is difficult to follow the paths traced by the points. To overcome this, we recolor the yellow cluster to red in panel 10, and finally select examples in panel 11 when the Y-axis represents time.

yellow (lower) cluster to red, and continue our investigation as in Figure 2.9(10) and Figure 2.9(11).

Now, the patterns are much clearer, and as the Y-axis is aligned with time, we can easily pick some examples. At this point we see that the red point we had selected did not give rise to any webpage load as the packets have been exchanged over long periods of time (see Figure 2.9(11)). However, the pink point and the green point provide us with one and two examples, respectively. Thus, one source-destination IP pair had in fact multiple webpage loads.

We can look for further examples in a similar fashion. After selecting the examples, we assign a name to the relation that will be learnt based on these examples send it over to the *Aleph* system for learning the rules. Table 2.1 is a fragment of the examples used. Here, two sets of packets have been selected as examples. As expected, the source and destination IPs are the same for all packets in an example. In *Aleph*, these examples are loaded along with the background facts. In the presence of only positive examples, *Aleph* learns the following rule, with X representing a set of packets:

webpage_load(X) :- same_src_ips(X),same_dest_ips(X), same-src_port(X, 80).

This rule states that a webpage contains packets which share the same source IP, destination IP and source port 80. Now we generate the tuples which satisfy the above result. Being a general rule, generating all the results will both take a long time and lead to clutter on the screen. To avoid this, we generate some results, which add up to a certain threshold. This is done by generating random subsets of the packet space, and checking if they satisfy the given rule. Further, we restrict the length of these subsets roughly to the range of the length of the positive examples. If the examples have a length between 6 and 10, we allow subsets of length 4 to 12. Note that these estimations are not required if the rule itself provides a clause specifying the length.

The results are now presented to the user, and we show them along the initial source IP vs. destination IP axes in Figure 2.7(12). This gives us an overall idea of which packets are involved in a webpage load (and which are not). At this point, we just choose to see the results, and each result is assigned a distinct color. This helps us to see multiple webpage loads between the same pair of IPs. It provides us with a chance to further guide the learning system by refining these results. This can be done by marking some of the results as negative if they do not seem correct, or merging/splitting multiple results to form a correct example. In Figure 2.7(13) we see the results where the Y-axis is a mix of time and source IP, and the X-axis is destination IP. Here, we merge 6 sets of examples, to form a correct example. Two examples are marked as negative because they are too short, and a third one is marked as negative because its packets have been exchanged over a long time span.

After three such iterations of rule refining, Aleph learns the following rule:

```
webpage_load(X) :- same_src_ips(X),same_dest_ips(X),
                   same-src_port(X, 80),length(X, L),
                   greaterthan(L,8),timeframe_upper(X,10).
```

This rule contains extra constraints when compared to the previous one. Apart from the packets sharing source and destination IPs, and the source port 80, they contain greater than 8 packets (length(X,L), greaterthan(L,8)), and occur within a time frame of 10 seconds (timeframe_upper(X, 10)). Since in **Figure 2.7**(13) we crossed out a few results based on their length, and time span, we can clearly see that Aleph was able to make use of these negative examples in learning the extra parameters in the rule.

2.6.2. Census Dataset

The Census dataset [5], extracted from the 1994 Census data contains population characteristics including age, education, religion, marriage, profession, annual income etc.

Looking at this dataset, we can identify traits of people, who have “high” earning potential or work for a certain kind of organizations, etc. We use our system to learn models for these kinds of characteristics. First, we sorted the dataset based on the age. Then we clustered it into ten sets by the age field. Since each cluster is assigned a unique color, this helps us identify age groups in the scatter plots.

Now we switch to the navigation interface and see a scatterplot of *education* vs. *working class.* As we make the Y-axis a mixture of *working class* and *age*, we see that for most education-working class combinations, people of all ages are present (Figure 2.10). We now learn a model for older men with high capital gains. We select some examples satisfying these criteria. In these examples we know that age and high savings are attributes already common to them. To learn other attributes common to them – e.g. education, hours of work etc, we mark the attributes on which the rule should depend. By default, the new rule depends on all the predicates existing in the database, and if the user does not select the attributes on which the rule should depend, the rule will simply state that *saves_more(A)* holds for old men with high capital gains, learning nothing new. Hence, using the user’s pointers, the background file used by Aleph is modified, and it learns a rule which is the disjunction of the following four rules:

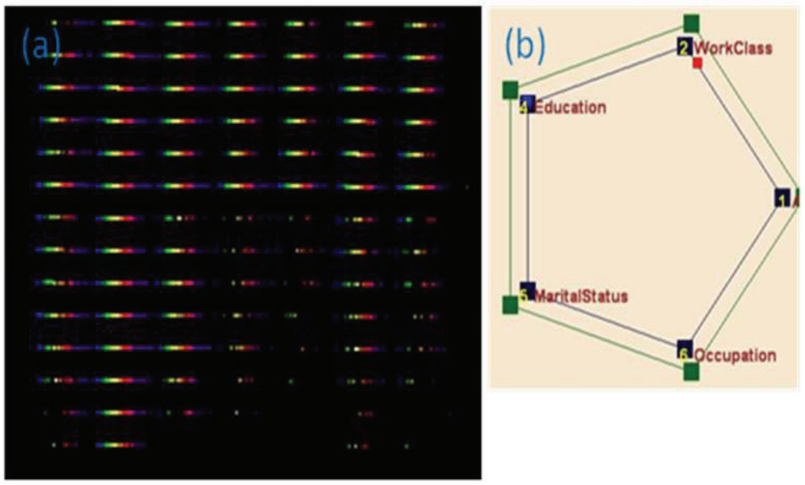


Figure 2.10: Scatterplot showing the distribution of age for education versus working class. The Y-axis is Education, and the X-axis is a mixture of WorkClass and Age.

1. **saves_more(A)** :- education_level(A, B), gteq(B,13), capital_loss(A, 0), native_country(A, 'United_States').

2. **saves_more(A)** :- race(A, 'White'),education(A, 'HS_grad'), marital_status(A,'Married_civ_spouse').

3. **saves_more(A)** :- race(A, 'White'),workclass(A, 'Private'),education_level(A, B), greater_than_equal_to(B,13).

4. **saves_more(A)** :- marital_status(A, 'Married_civ_spouse'),occupation(A, 'Exec_managerial').

The quoted strings in the above rule come from the original dataset. The first rule holds true for people with a minimum education level of 13 (Bachelors in this example), no capital loss, and US nativity. The second rule holds for white HS grads that have the marital status 'Married_civ_spouse'. The third rule holds for whites working in the private sector, who have the minimum education level of Bachelors (13). The last rule accepts people whose marital status is 'Married_civ_spouse' and work as an executive manager.

In contrast to the network dataset, here our rule is a disjunction of conjunctions. Also compared to [99], our rules are neither hand constructed, nor restricted to pure conjuncts.

2.7 Conclusion

This chapter describes a VA infrastructure for analyst-guided discovery of relations present in datasets. The analyst just needs to supply patterns from the visual interface and the system automatically learns the relations based on these patterns. Another important aspect of the infrastructure is that it is readily scalable over different datasets - all that is needed is encoding the background knowledge about the new data set; all the other aspects of model building remains the same. The feasibility of our approach is demonstrated by implementing a prototype VA system on which we build models of relations in two different data sets.

The rule learning process in our system is fast enough to allow the system to remain interactive. Further, we can allow the validation of the learnt model by presenting the rules to the user in a simplified format, i.e. in an everyday language format.

In the next chapter, we present another VA system which helps users classify or segment data by providing their input through a visual interface. This project differs in the aspect that we are not trying to *find* patterns in a large amount of data, but rather want to quickly classify large datasets with minimal user effort.

Chapter 3. A Visual Analytics Approach to Machine Learning

The process of learning models from raw data typically requires a substantial amount of user input during the model initialization phase. In this chapter, we present an assistive visualization system which greatly reduces the load on the users and makes the process of model initialization and refinement more efficient, problem-driven, and engaging. Utilizing a sequence segmentation task with a Hidden Markov Model as an example, we assign each token in the sequence a feature vector based on its various properties within the sequence. These vectors are then clustered according to similarity, generating a layout of the individual tokens in form of a node link diagram where the length of the links is determined by the feature vector similarity. Users may then tune the weights of the feature vector components to improve the segmentation, which is visualized as a better separation of the clusters. Also, as individual clusters represent different classes, the user can now work at the cluster level to define token classes, instead of labelling one entry at time. Inconsistent entries visually identify themselves by locating at the periphery of clusters, and the user then helps refine the model by resolving these inconsistencies. Our system therefore makes efficient use of the knowledge of its users, only requesting user assistance for non-trivial data items. It so allows users to visually analyze data at a higher, more abstract level, improving scalability.

3.1 Introduction

With the tremendous growth in physical and online data collection technology, we are now experiencing an explosion of digital information. Since a large amount of these data are unstructured, various machine learning techniques have been developed to assign structure to these data to make them machine readable. This process can allow the machine to reason with and draw insight from data almost automatically. However, all such tasks depend heavily on large amounts of user-tagged data as the starting point, and use various semi-supervised learning methods [105]. Due to the high user input required, such tagged data is difficult to construct. On the other hand, learning is hard for machines too, since without human guidance, they cannot efficiently generalize from a few examples. Efficient teachers are needed that point out relevant things to learn which gave rise to active learning. In active learning, the user still tags the query examples in isolation, with no idea about the bigger picture. We provide an overall view of the data which allows users to both initialize the model, and provides them a context during the labelling stage. Hence our system has the theme of visual analytics meets active leaning.

One crucial idea behind our system is that given good feature vectors to represent each data point, points that are similar will be close-by in the feature vector space. Here, we mean data-points which though rich in semantics, do not have an explicit high-dimensional feature vector automatically attached to them. In such cases we need to *design* feature vectors to represent the semantics and structure of the data-points. We aim to achieve this in our system by designing feature vectors which encompass a data point's structure, context, and location in the dataset. If some sort of semantic information is available, that can be added to the feature vectors as well.

Based on the above feature vectors, we design a visual interface where data is displayed in 2D space based on their feature vector similarity. This gives the users an overview of the dataset, and they can easily observe sets of data points which form spatial clusters. At this stage, we can apply clustering algorithms which arrange similar points closer together.

Clustering as we know is a general approach which can help users explore and analyze large data sets since it lets users work with groups of objects, rather than individual objects which can be large in number. Clustering associates objects in groups such that the objects in each group share some properties (similar feature vectors) that hold to a lesser degree for the other objects. Spatial clustering builds clusters from objects being spatially close or having similar spatial properties. However, clustering methods when run automatically can give non-intuitive results. Therefore, we allow the user to tweak the parameters of both the clustering algorithm and the data's feature vector to improve results.

We can use any well-known clustering algorithm to subdivide the points into clusters. This can help provide the initial biasing for any classification algorithm.

However, the model learnt initially is a very rough estimation of the actual model, and needs fine-tuning to improve performance. For this purpose, we keep the user in the loop and utilize human insight to resolve inconsistencies. Overall, our visualization system allows the analyst to: (1) identify the classes in the data and communicate these to the model learning system; (2) assess the fitness of the generated model by the structure it imposes on the data; and (3) refine the model by communicating misrepresented patterns back to the model learning system.

The model learning system uses the visualization to *explain* the model to the analyst. While most communication of the analyst is gesture-driven, editing facilities are available to directly specify or refine the model. We call this iterative model building process data-driven, user-assisted model *sculpting* or model *debugging*.

After we have learnt an initial model, we apply it to the dataset to segment it. The user can then examine the visualization for possible misclassifications, and reclassify the data points. To guide the refinement of the model, we develop an interface similar to a step debugger in a common software development environment. Here, the model's structural graph represents the program script while the visualization is the program output. Users can step through the model's graph, examine the visual model explanations, and refine the corresponding rules if required.

The main contribution of this work is that it lightens the huge burden of individually hand tagging data, and allows users to tag data at a higher level, with greater interactivity. Our approach can provide a solution to a large range of segmentation and classification problems in the presence of complex and ambiguous data. This includes the audio/video domain where we want to segment the input by speakers, scenes, moods, etc, and the image domain where we classify images by content type, or segment an image into objects.

On the other end of the spectrum are methods which try to learn important feature vectors for data classification automatically [73]. However, they are highly time intensive. By using our approach, we can let the user into the loop, and allow him to guide the system making the process much more efficient.

This chapter is structured as follows. Section 3.2 presents related work, Section 3.3 describes relevant theoretical aspects, Section 3.4 provides details on implementation, Section 3.5 presents results, and Section 3.6 presents some results for categorizing images. Section 3.7 ends the chapter with conclusions and pointers for future work.

3.2 Related work

This work follows the general Visual Analytics idea of combining human domain knowledge with automatic data analysis techniques by providing users with interactive visual interfaces, whereby 'interactive' means that users can actively participate in the analytical process as it evolves. Though our main focus is on allowing users to facilitate the process of model learning, visually guided data clustering is an integral part of it. There has been some work related to this field, including some in Visual Analytics. The approach described by Schreck et al. [78] allows users to leverage existing domain knowledge and user preferences, arriving at improved cluster maps. Zhang et al. [102] present a paradigm for visual exploration of clusters. Further, to make sense of the cluster results, visual representations are necessary. Projection-based approaches as presented in [36,20] are common. We use a Multidimensional Scaling (MDS) approach in this work.

Learning models from patterns is an active research topic in various branches of computer vision. But there the pattern examples in most cases originate directly from image analysis, promoting unsupervised learning where subtle anomalies (the unexpected data) or new families of patterns which the model parameters cannot capture often go undetected. Visual analytics, on the other hand, aims to be more flexible in the data constellations encountered, appealing to the complex pattern recognition apparatus of humans and their intuition, creativity, and expert knowledge to point out unusual configurations for further testing and model refinement. Papers recognizing this are currently emerging. Janoos et al. [42] used a visual analytics approach to learn models of pedestrian motion patterns from video surveillance data, in order to distinguish typical from unusual behaviour in order to flag security breaches in outdoor environments. Their semi-supervised learning approach in which users interact with video stream data improves upon the standard unsupervised learning schemes that are typically used in these scenarios.

There has been little work in the field of visually assisted machine learning. The last few years at VAST has seen some papers in this domain. Our own work on Model-

Driven Visual Analytics [26] as described in the previous chapter describes a visual analytics system for high-dimensional data analysis. In this system, users visually explore the data in a high-dimensional space, and mark example patterns to iteratively learn rules using Logic Programming. The paper on LSAView [17] provides a visual analytic framework to analyze the model parameters in Latent Semantic Analysis, hence promoting model learning and debugging. Andrienko et al. [2] present an approach to extracting meaningful clusters from large databases by combining clustering and classification, which are driven by a human analyst through a visual interface.

3.3 Theory

We shall use a text segmentation application as an example to illustrate our work. Important here are the concepts of *document* and *token*. In text segmentation we then have a collection of documents that contain the text and the tokens are the words appearing in these text items (each token is composed of a collection of letters). But we may just as well perceive a set of images (or even videos) as a collection of documents and coherent image regions as tokens (which are then further composed into individual pixels). So we see that these concepts are quite general.

The data segmentation task involves working on multiple documents, which are further divided into tokens to do any processing and calculations. A document is the unit which is subject to either classification or segmentation. For text, it can be a single string (an address), a story (news entry), or an entire web page. A token is usually the smallest semantically meaningful unit in the document, and can vary depending on your approach. For text, it is usually a word; for images, it can be a pixel, or a contiguous iso-value region. For video, the natural unit is a frame. We observe that if there is a well-known boundary, then it is easy to extract tokens. In cases where such boundaries are missing, a multi-scale approach can work well.

Given the tokenized dataset, we often need to start with a coarse segmentation. This segmentation gets refined as we apply the learning algorithm, and possibly involve the user in the loop. This *windowing* approach has been used in the segmentation of audio broadcast news into stories [94]. In the absence of a numeric way to define data tokens, these windows provide a simple way to define a feature vector.

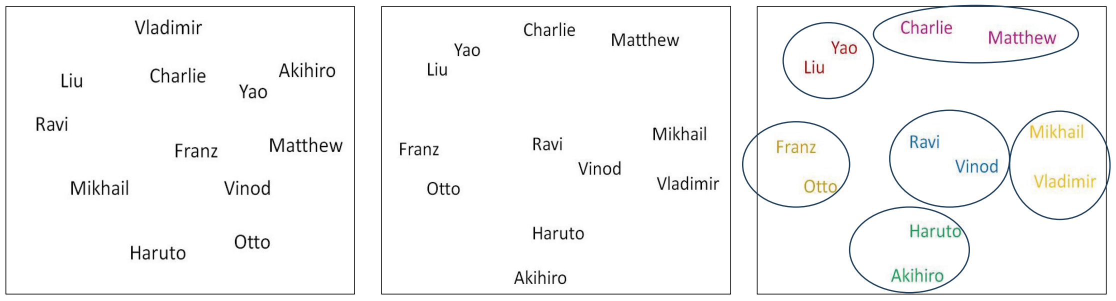
Segmentation is the process of converting the data in a raw stream of information into structured records. Given a schema consisting of *n* attributes and an input string, the problem of segmenting the input string involves partitioning the string into contiguous sub-strings and assigning each sub-string a unique attribute from the *n* attributes. For example, given the address schema consisting of the five attributes <COMPANY, STREET, CITY, STATE, PHONE> and the input string "Adieu Travel 117 Franklin St Dansville NY (716) 335-2222", the task of segmentation is to convert the string into the address record: <Adieu Travel, 117 Franklin St, Dansville, NY, (716) 335-2222>. Further difficulties may emerge when the records appear in different order in different strings. The running example in this chapter uses the BigBook business address dataset [3] which contains approximately 3000 business addresses from New York State.

Information on the web (such as product listings, audio/video collections, or newscast) exists in an unstructured format. Segmentation into structured records is necessary to facilitate efficient query processing and analysis. The same goes for images and video for content-based image retrieval.

Segmentation techniques either use rules for identifying attributes or employ statistical models. Rule-based approaches require domain experts to create and maintain a set of rules for each application domain. It is difficult to anticipate all possible variations in the documents to be segmented and design rules accordingly. Further, noise in the data can compound the difficulty. Therefore rule-based approaches are neither scalable nor robust. In contrast, statistical approaches automatically learn a statistical model for each application domain. The variability and noise in the input text data are elegantly dealt with by the statistical characteristics inherent in such approaches.

3.3.1. Hidden Markov Models

Without loss of generality, we use a Hidden Markov Model (HMM) [71] to learn the segmentation model. The HMM is a widely used statistical model used for data segmentation. It is a generative model since it captures the probability distribution of observations (e.g. the input strings in case of text segmentation). HMMs are commonly used to represent a wide range of phenomena in text, speech, and even images (using 2D HMM [43,51].). An HMM consists of a set of states *S*, a set of observations (in our case words or tokens) *W*, a transition model specifying $P(s_t|s_{t-1})$, the probability of



(a) Random layout (b) Window-based layout (c) Clustered layout

Figure 3.1: This image shows the layout of points representing people's names (a) randomly, and (b) based on the window based approach. The user interacts with the graph in (b), and marks people who belong together (here based on nationality), giving us (c).

transitioning from state s_{t-1} to state s_t, and an emission model specifying $P(w|s)$ the probability of emitting word w while in state s.

To compute hidden state expectations efficiently, we use the Baum Welch algorithm. Emission and transmission models are initialized using the approximate classification done using the visual interface. Finally, we use the Viterbi algorithm with the learned parameters to label the test data.

3.3.2. Active Learning

Active learning is a form of supervised machine learning in which the learning algorithm is able to interactively query the user (or some other information source) to obtain the labels at new data points. In statistics it is sometimes also called optimal experimental design. A good survey of various active learning approaches is covered in [80].

For a large number of situations unlabeled data is abundant but labeling data is expensive. In such cases the learning algorithm can *actively* query the user for labels. This type of iterative supervised learning is called *active learning*. Since the learning algorithm optimally chooses the examples, the number of examples to learn a concept can often be much lower than the number required in normal supervised learning. However, with this approach there is a risk that the algorithm might focus on unimportant or even invalid examples.

Let T be the total set of all data under consideration. For example, in a disease prediction problem, T would include all the people who have been diagnosed with a

Table 3.1: Feature vector table. Each matrix entry represents the presence or absence of a token in a document. Each row represents the feature vector for the corresponding token.

	ABBAB	ABBCB	CDADC	DCCDC	EDFEF	FEEFF
A	1	1	1	0	0	0
B	1	1	0	0	0	0
C	0	1	1	1	0	0
D	0	0	1	1	1	0
E	0	0	0	0	1	1
F	0	0	0	0	1	1

disease and all additional people that want to test for that disease.

At every iteration, T is broken up into three subsets: a) data points where the label is known, b) data points where the label is unknown and c) A subset of the unlabelled data points that is chosen to be labeled.

All active learning scenarios involve evaluating the informativeness of unlabeled instances, which can either be generated afresh or sampled from a given distribution. The simplest and most commonly used query framework is uncertainty sampling [49]. In this framework, an active learner queries the instances about which it is least certain how to label. This approach is often straightforward for probabilistic learning models, and can be easily modified to work for other approaches as well.

To select the next data point to be sampled, we use the pool-based sampling method since we start our analysis with a large collection of unlabelled data.

3.3.3. Overall Concept

The main idea behind this project is that humans with specific domain knowledge can easily spot the pattern in data, something which is very difficult for a machine to do. However, to make use of this expert knowledge, the data should be displayed such that the task for the user becomes easier. A good approach is to identify terms that exhibit similar characteristics, and display them together. This essentially means that we need to find appropriate feature vectors for each term and then cluster them together. In data segmentation and classification tasks, the location and neighborhood of a word has a great bearing on its classification. We observe two distinct cases:

I. Data belonging to the same class appears together *within* documents (e.g. when they belong to the same news topic), or

II. Data belonging to the same class appears in similar locations *across* documents (e.g. city names appearing in addresses).

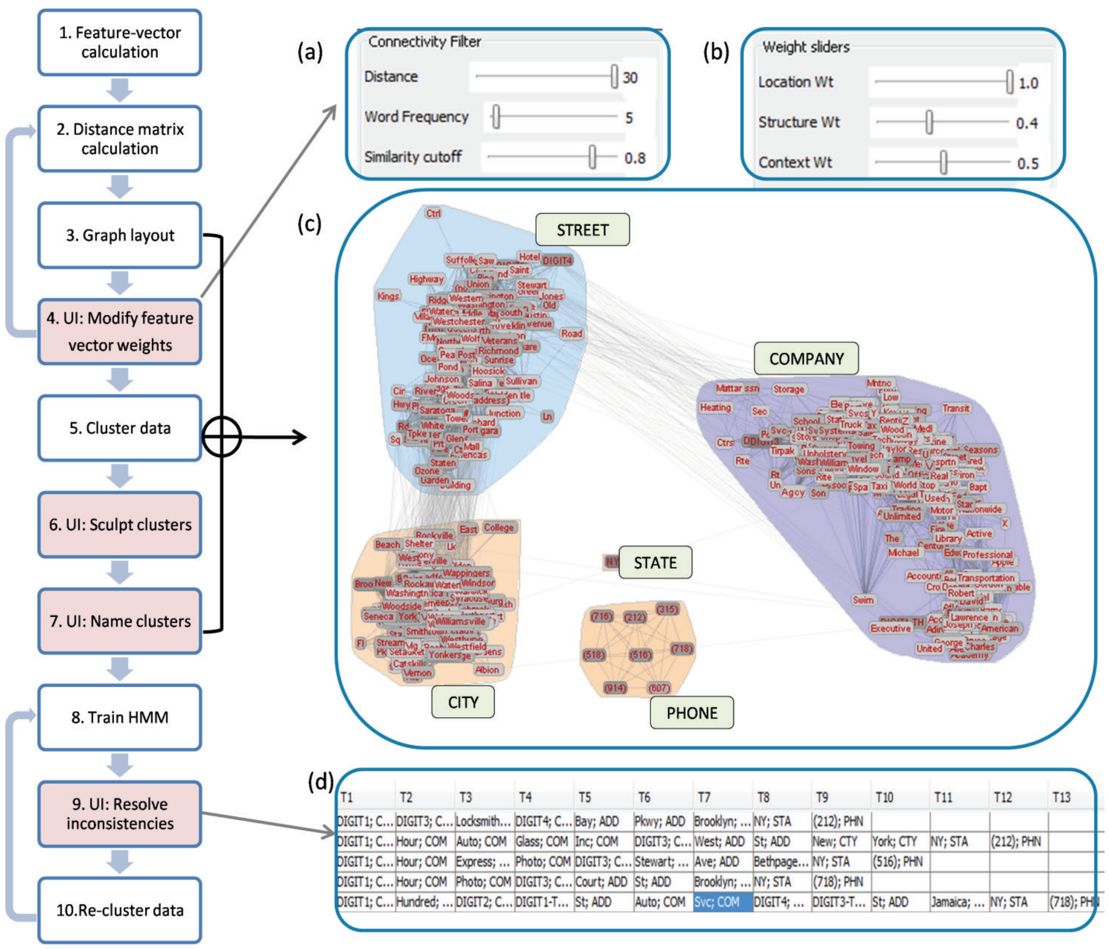
We propose to treat each document as a sequence of tokens. For the case where each document contains one topic, the feature vector for the tokens is simply a binary vector with a 1 representing presence, and a 0 absence. An example is shown in Table 3.1 where each column is a document, and each row is a token. Here our vocabulary contains the letters (A, B, C, D, E, F), and we can clearly see the three clusters (A, B), (C, D) and (E, F). A better way to see patterns in a large dataset is to calculate the similarity between the feature vector of the tokens (using the dot product, or some other well-known method), and laying them out in 2D using Multidimensional Scaling (MDS) [45]. When tokens belong to exactly one topic, the classification task is quite simple. However the fact that most tokens belong to multiple topics makes the task harder, and often impossible to solve for the machine.

In Figure 3.1, we show an example to illustrate the value of this approach. In Figure 3.1(a), the tokens from our dataset are laid out randomly. Briefly looking at it makes it clear that these nodes represent male first names. However, in Figure 3.1(b) which shows the points laid out using the windowed approach, a user can see the further pattern that these names belong to different nationalities, and in fact this property (same nationality) causes nodes to be laid out close by. Figure 3.1(c) shows the words colored and circled according to their classification by the user. This task could not be done automatically since even though tokens in the same class occur close by, tokens belonging to different classes are also at a similar distance in multiple instances. Only the domain knowledge of a user can help resolve such ambiguities.

For case II, i.e. when tokens in the same class appear in similar locations across documents, we need to capture the locations they appear within all documents. To do so, we divide the tokens in each document into equal numbers of windows (N_w), i.e. each token is assigned a window according to its relative position within the document. Given document d_i , token tok_j, number of tokens in the document $num(d_i)$ and the location of the token $loc(tok_j)$, its window number is:

$$win(i, tok_j) = N_w \times \left(\frac{loc(tok_j)}{num(d_i)}\right) \qquad (3.1)$$

$$\frac{tok_1\ tok_2 |\ tok_3\ tok_4 |\ tok_5\ tok_6}{win_1 \quad | \quad win_2 \quad | \quad win_3} \qquad \text{(e.g.)}$$



Connectivity Filter

Distance	30
Word Frequency	5
Similarity cutoff	0.8

Weight sliders

Location Wt	1.0
Structure Wt	0.4
Context Wt	0.5

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
DIGIT1; C...	DIGIT3; C...	Locksmith...	DIGIT4; C...	Bay; ADD	Pkwy; ADD	Brooklyn; ...	NY; STA	(212); PHN				
DIGIT1; C...	Hour; COM	Auto; COM	Glass; COM	Inc; COM	DIGIT3; C...	West; ADD	St; ADD	New; CTY	York; CTY	NY; STA	(212); PHN	
DIGIT1; C...	Hour; COM	Express; ...	Photo; COM	DIGIT3; C...	Stewart; ...	Ave; ADD	Bethpage...	NY; STA	(516); PHN			
DIGIT1; C...	Hour; COM	Photo; COM	DIGIT3; C...	Court; ADD	St; ADD	Brooklyn; ...	NY; STA	(718); PHN				
DIGIT1; C...	Hundred; ...	DIGIT2; C...	DIGIT1-T...	St; ADD	Auto; COM	Svc; COM	DIGIT4; ...	DIGIT3-T...	St; ADD	Jamaica; ...	NY; STA	(718); PHN

Figure 3.2: Overall system. (Left): Flowchart outlining our approach. The red boxes represent user-interaction steps. (Right): An overview of the system. (a) and (b) show sliders used by the user in step 4. (a) Sliders to filter the graph, (b) Sliders to adjust the weights of the feature vector components, (c) The entities are laid out in a node-link diagram, along with the clusters; (d) The interface showing segmentation results that fulfill certain inconsistency criteria – the user then interacts with the system (step 9) to resolve these inconsistencies.

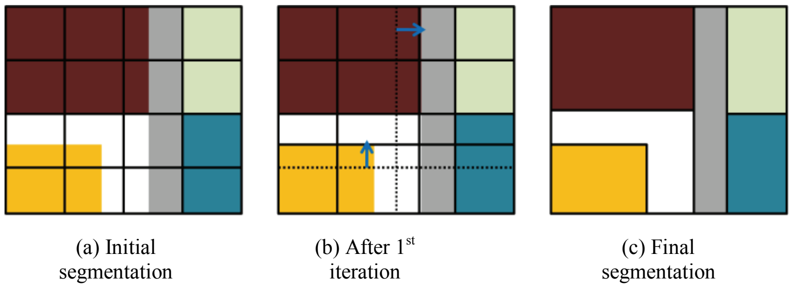


Figure 3.3: An example of using a coarse segmentation (windowing) to initialize the image segmentation process.

The example in Table 3.1 shows how a document with 6 tokens is divided into 3 windows. We construct the feature vector of each token as the histogram summarizing the frequency with which a token has appeared in different windows. An ideal window size is data and task dependent, and is best chosen while interacting with the visual layout of the data. For visualization, we can again calculate token similarity by taking a dot product between feature vectors, and display them in 2D using MDS.

An example of using windows for the initial segmentation of data is shown in Figure 3.3. Here our task is to segment the colored image into its constituent blocks (Figure 3.3(c)). Initially we divide the image into 4×4 sub-images. This starts off the learning algorithm, and after a few iterations gives the required result. Note that the window sizes are at a similar scale to the final segments. If we start off with windows too large (larger than the largest segment), then the results might not be even close to expected. On the other hand, windows that are too small increase the problem size, making it longer (and harder) to solve.

This windowing approach is similar to the one used in [57] which use bigrams and trigrams as basic units for document visualization.

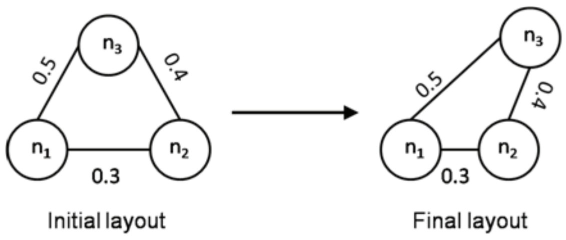


Figure 3.4: Graph evolution based on node dissimilarities (as shown along the edges)

3.4 Implementation

The basic approach of our system is as follows: we take tokenized data documents, calculate the relevant feature vectors, and allow the user to help initialize and refine models to cluster the data. As mentioned, we use HMMs as an example of a learning approach. An overview of our system is shown in the flowchart in Figure 3.2.

Initially the data is preprocessed, and tokenized. So each document will contain a sequence of tokens which need to be labeled. We consider all identical tokens to be instances of the same *entity*, and our feature vector calculations are based on these entities.

Observing the datasets led us to the realization that there are three important properties which can help us classify a token –

(a) Its structure. For text this includes – is it a word, is it a number, its length etc.
(b) Its context, e.g. the tokens that appear before and after it,
(c) Its location in the document.

Structure and context can also include information beyond the information contained in the tokens themselves. This includes meta-data which comes attached. Webpage data though not necessarily machine readable already exhibits some structure via html meta-data. The non-visual web browser presented in [55] uses the meta-data to segment web-pages into semantically related regions.

We numerically capture the above properties in each entity's feature vector. This means that a feature vector contains multiple high-dimensional vectors. When we calculate the similarity between any two entities, we need to calculate similarities based on each property separately, and then combine them into a final value.

A survey on cluster data mining techniques [10] concludes that data specific attribute selection has yet to be invented. Recent work on unsupervised feature learning [73], confirms that user interaction can greatly reduce the learning time. To help alleviate this problem to some extent, we give the power of assigning the weights of the feature vector elements to the user. The similarities due to different elements are all normalized to ranges between [0, 1]. Given two entities x and y, and similarity values $sim_i(x, y)$ and weights w_i due to the three properties, the final similarity between them is:

$$sim(x, y) = \frac{\sum_i w_i * sim_i(x, y)}{\sum_i w_i} \qquad (3.2)$$

3.4.1. Initialization Stage

The entities are displayed as a node-link diagram using a force-directed layout algorithm [35]. This algorithm considers a spring-like force for every pair of nodes *(x, y)* where the ideal length δ_{xy} of each spring is proportional to the graph-theoretic distance between nodes x and y. Minimizing the difference between Euclidean and ideal distances between nodes is equivalent to a metric multidimensional scaling problem. Here we use the *dissimilarity* between nodes as the ideal distance between two nodes. The dissimilarity is simply the inverse of similarity:

$$dissimilarity(x, y) = 1 - sim(x, y) \qquad (3.3)$$

Figure 3.4 shows the graph evolution, and finally patterns emerge showing the overall structure of the dataset. At this stage, the user can modify the weights of the feature vectors. This updates the node similarities as well as the graph layout.

Note that any pair of nodes will have a similarity measure, but keeping all the edges will give us an extremely dense graph with no discernible structure. Removing

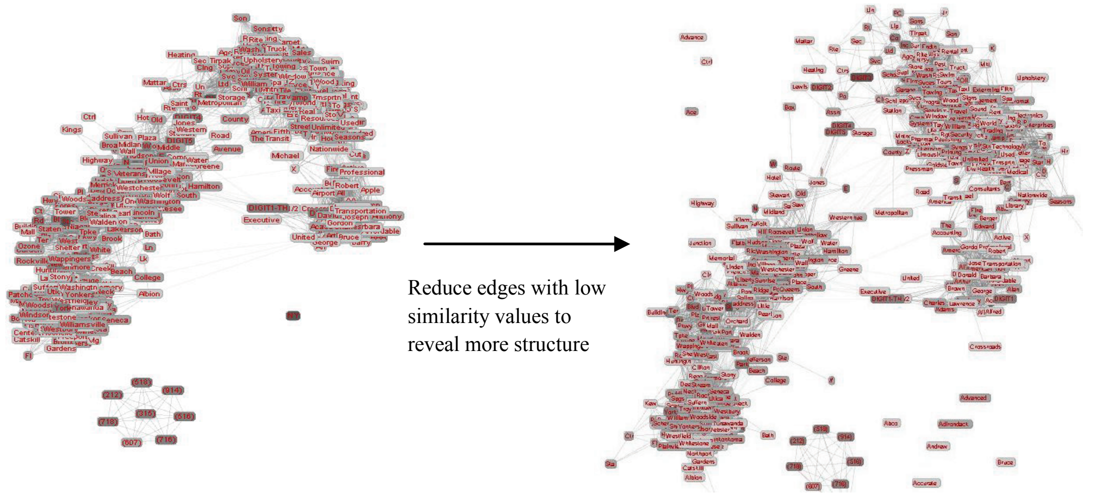

Reduce edges with low similarity values to reveal more structure

Figure 3.5: Graph structure: As we change the cutoff values for node-pair similarity from 0.5 to 0.85, the graph goes from being almost a single mass of nodes on the left, to one displaying more clusters on the right

edges between nodes with low similarity (< 0.5) helps this structure to emerge better. The optimal cutoff value depends on the data density and the feature vector selection (i.e. weights). We allow the user to select this cutoff value to find the appropriate optimal value. Figure 3.5 shows an illustration of how the graph layout changes with the removal of edges representing low similarity. The one disadvantage is that some nodes become isolated, i.e. are not connected to any other node. We can handle this by either allowing the user to assign them to a cluster *after* we call the clustering algorithm. Else, the HMM can classify them during the learning stage, and they get displayed in the corresponding cluster during the refinement stage.

Users can affect the graph layout, and the clustering using the following inputs:

(a) Modifying the weights of the feature vector elements

(b) Modifying the similarity cutoff – this removes the edges representing low similarity, and also reveals a clearer structure of the dataset.

(c) Modifying the frequency cutoff of the data points. This leaves only the most prominent nodes behind, which act as representative nodes, and give the users a good idea of the classes present in the dataset.

(d) Modifying the "fineness" level of clustering – this controls how many clusters the data gets split into. Getting the most appropriate number of clusters will reduce the load on the user during the *sculpting* stage. (Step 6 in Figure 3.2).

Further, we notice that the visualization reveals more structure when we display entities from the entire dataset, rather than from a smaller subset. This is because in the smaller dataset most low frequency entities do not cluster well. As we increase the number of documents, some of these frequencies improve, and the entities become better defined.

When the various settings work together to produce a semantically meaningful structure, the user can call the clustering algorithm. A good choice of weights at this early stage will minimize the work required from the user at later stages – especially at the cluster sculpting stage, and relabeling the tokens at the refinement stage. In most cases, a range of values gives decent results, hence carefully updating the weights and observing the visualization can take care of this problem.

We use the Markov Cluster Algorithm (MCL), an unsupervised cluster algorithm for graphs based on the simulation of stochastic flow in graphs [21]. Only those similarity values that are above the cutoff specified by the user are passed to the clustering algorithm. This has a similar effect as it has on the visualization – it helps segment the data better, i.e. divides them into more classes. Once the clusters are calculated, they are displayed by forming a convex-hull boundary around the data points they contain. Further, we relax the strength of edges across clusters to reduce their spatial overlap. (See Figure 3.6)

The physical boundaries between clusters, might lead the user to discover some semantic discrepancies – this usually occurs due to entities which are ambiguous and can occur in multiple classes, or in cases where the feature vector is not able to classify an entity correctly. In this case, a user can visually *sculpt* the clusters – this involves splitting and merging clusters, dragging nodes from one cluster to another, or duplicating nodes into different clusters (for example, the token 'York' may appear in 'Street', 'State', 'City', and even 'Company' depending on context). The visualization assists the

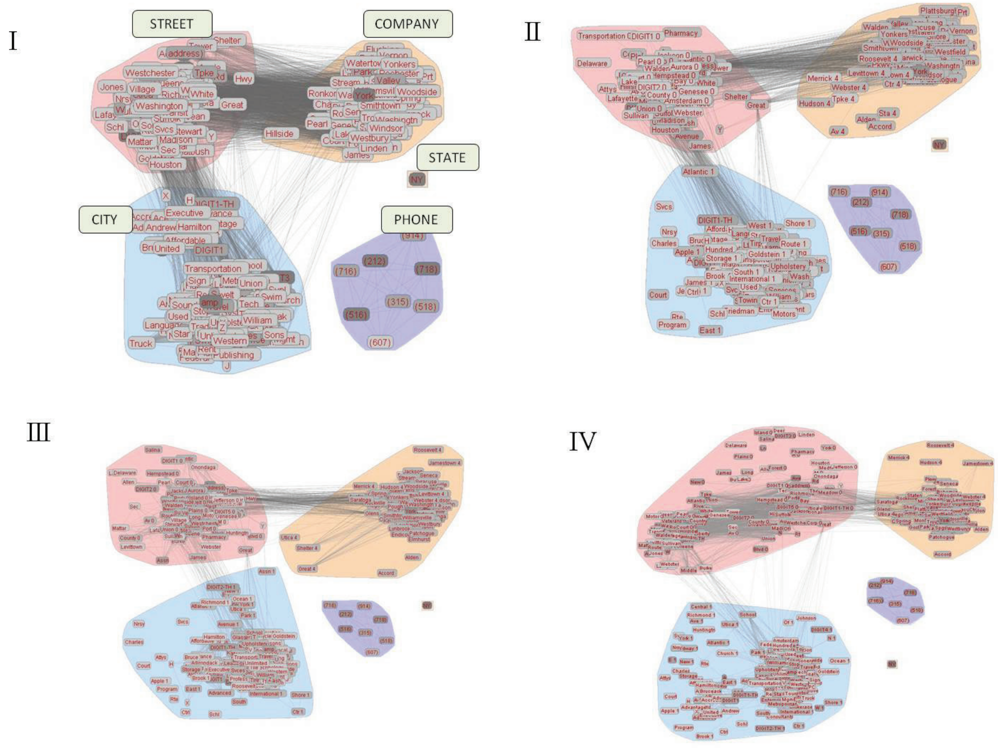


Figure 3.6: Result. The four panels show the evolution of the clusters as the user reclassifies the segmentation of some address entries, and the HMM relearns the model taking this into account

user at this stage by highlighting the nodes that have more than 99% of edges to nodes within the same cluster. At the end of this stage, the number of states in the HMM is assumed to be the number of clusters in our visualization. Further, the emission probabilities are calculated based on these clusters – entities with a higher frequency in the original dataset have a higher emission probability. To allow the model learning algorithm to assign a word to a different class from that in the above cluster, we assign a small emission probability value to all the words in all the states other than the one they appear in.

The initial transition model consists of a graph containing the states identified by the user. The transition probabilities are calculated based on the clustering as well – we increase the count of transition between two states s_i and s_j whenever a word in s_i is followed by a word in s_j. Finally these counts are normalized to calculate the transition probabilities. Since the Baum-Welch algorithm is used to *re-estimate* the parameters of an HMM, it always takes transition and emission models as inputs, refines them, and outputs these refined models. Later, to make use of the user inputs, we take the output from one step, modify it based on these user inputs, and pass it on to the next step for re-learning. After learning an HMM, we use the Viterbi algorithm to estimate the hidden states, and segment the strings.

3.4.2. Refinement Stage

At this stage the user can help resolve inconsistencies in the segmentation, and help debug the model. To make this more intuitive, we first need the user to give the clusters semantic names. This is done by highlighting the nodes in all clusters which has a very low similarity to nodes in other clusters. In case the presence of some entity makes the identity of the class ambiguous, the user can choose to see some strings where the entity appears. For example if Washington appears in a cluster with what mostly appears to be street names, seeing it in context will help the user be sure that the given cluster indeed contains street names, not cities or states. Two examples are given below:

1. D Sacilotto 399 *Washington* St New York NY (212) 966-7274
2. Burke & Casserly PC 255 *Washington* Avenue Ext Albany NY (518) 452-1961

At this stage, we have named clusters and a trained HMM. Now, we can involve multiple users in the refinement stage. Just as interaction with users can quickly help define models, misclassifications at any stage can slow the process down. Hence involving multiple users at the refinement stage can help remove mistakes made by a user due to either lack of knowledge, or uncertainty.

We pick documents that are classified with a very low probability value. This usually happens when the document contains tokens that cannot be classified easily since they belong to multiple classes. The user resolves these inconsistencies, and if such a token is indeed classified into more than one class by the user, we duplicate it to represent various identities of a single token. This also reduces the edges across clusters, making the model better defined. We ask the user to reclassify a few documents at a time, and then retrain the HMM to reflect these changes.

During this debugging stage, the user can highlight the consecutive tokens in the graph layout. When a specific token is selected, the corresponding node in the graph is highlighted. Also, the cluster to which it has been classified is also highlighted. This helps the user see the possible classes the token could belong to. The user can traverse forwards and backwards along a given document to establish if there is any misclassification.

The visualization where the clusters are marked often excludes entities with low frequency – either because they were filtered out by the user, or they were not assigned a cluster due to low similarity to any cluster. However, after the HMM is trained, all the entities in the dataset will be assigned emission probability values. Further, if we want to display duplicates for entities belonging to multiple clusters, the visualization will become too dense. To overcome this problem, after the initialization stage, we only display the entities which have very few high similarity edges to nodes in other clusters.

In order to show duplicates for an entity, a node is created for each class where its emission probability is above a minimum threshold $thresh_{emit}$ – this threshold is a function of the number of tokens in the dataset. The nodes are named such that the one with the highest emission probability retains the original name, and the others are renamed by appending the cluster number. Further, when an entity E in cluster i is split to form a node E_j in cluster j, all the edges from E to nodes in cluster j are rerouted to contain E_j as one of the incident nodes.

During refinement, as a user reclassifies a token within a document, we keep track of the number of times each entity was added to or removed from a cluster. While learning the next HMM, we update the emit values to reflect these counts. At any iteration, if the emission probability of a node falls below the threshold threshemit, it is removed from the graph as long as it is not the only representative for its entity.

Further, the first round of transition models are completely random, and allow transitions between any two states. Removing the non-existent transitions by just updating the emission probabilities would be a very slow process. To avoid this problem, at any stage, we take a weighted sum of the existing transition probabilities, and the transition probabilities calculated from the documents labeled in the current run.

This process of re-labeling the documents and re-learning the model continues till the user is satisfied by results of the most uncertain segmentations. The model can now be stored, and used for further segmentation. Later, if we have data which has the same type, but a different vocabulary, we can again repeat the learning and refinement process.

3.5 Results

The system we described can help in the classification of varied types of datasets. As a starting step, we worked with text-based datasets since they require minimal preprocessing, and can show the utility of our method. In this section we will see a running example of learning a model (HMM) for the business address dataset. The BigBook business address dataset contains approximately 3000 business addresses from New York State. This dataset contains a large number of distinct integers, and each separate integer does not have a special meaning, but usually just represents a street or building number or is a part of a business name. To make sure that the HMM is able to learn well, and apply the model to addresses with new tokens, we replace each integer with the term DIGIT-<*i*> where *"i"* is the length of the integer.

After the initial preprocessing, we calculate feature vectors based on the three properties of structure, context and location:

- The feature vector on *structure* contains the following information: does it contain a letter, does it contain a digit, does it contain a non-alphanumeric symbol, does it begin with a capital letter, is it all caps, and its length. Except for the last element, the others are binary.
- The feature vector on *context* contains a summary (i.e. histogram) of the structure based feature vectors of the tokens that appear immediately before and after the given entity in all the documents.
- Finally, for the feature vector on *location*, we use the windowing approach as presented in Section 3.3.2. Initially, if we know the number of classes (N_c) in the dataset, we use its multiples as the number of windows. In the absence of this information, the window size is decided by the document lengths (i.e. number of tokens in the document).

Now the data is displayed on the screen, and as the user plays with the similarity cutoff slider, the graph goes from being one main cluster with all nodes interconnected, to one which shows the inbuilt structure of the data. At this point, the user clusters the dataset. Since there is a lot of overlap between the terms in ADDRESS and CITY (see Figure 3.6), there is an extra cluster which contains these common tokens. The user

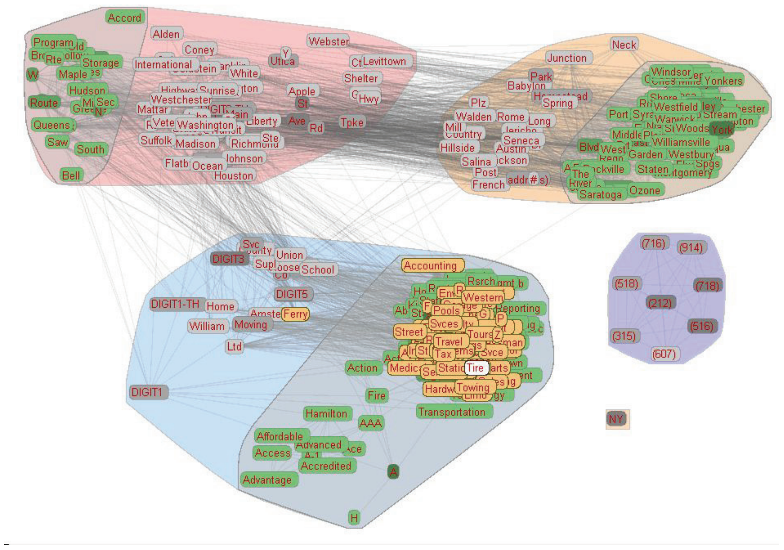


Figure 3.7: Graph showing the inner entities/nodes in green. These nodes help the user both during the cluster sculpting stage, and during the cluster naming stage. Here, we highlight some “inner” nodes in the lower left cluster - examining them shows that this cluster contains words belonging to Business Names.

organizes and splits this cluster to the best of his knowledge, and merges the corresponding halves with the two clusters.

Next we highlight the inner nodes in the dataset, which shows the user the representative terms for each cluster, and he is able to name them. Figure 3.7 shows an example of these inner nodes. Now we initialize the emission probabilities for all states based on the clusters. If an entity belongs to a cluster, its emission probability is the ratio between frequency of the entity and the total frequency of the entities in the cluster. As mentioned in Section 3.4.1, we assign a small probability value to all the entities *not* belonging to the cluster. At this stage the Baum-Welch algorithm is called to learn an initial model.

Now the user is shown strings with one token that has high similarity to nodes in multiple clusters. Given that the token has high similarity to entities in classes c_1 and c_2, we:

(a) Assign a positive value if the classification was correct

(b) If the classification in class c_i is deemed incorrect, and reclassified to class c_j, then we assign a heavy penalty to c_i and a heavy bonus value to c_j.

The accumulated values at the end of a reclassifying round are used to modify the emission tables learned by the Baum-Welch. Further, based on this, we also split the nodes if they are classified into different classes in the cluster. The user helps in the debugging process till the clusters become better defined, as shown in Figure 3.6. After four rounds of refinement based on the tokens with high presence in multiple clusters, we get a good model for classification. The time taken between iterations depends on the complexity and implementation of the algorithms used to learn the model, and segment the strings – in this case the Baum Welch, and Viterbi algorithm respectively. In the address dataset example used, a user fairly familiar with New York addresses can train the model in about 10 minutes.

3.6 Extension to Image classification

In this section, we extend our approach to the problem of categorizing image data according to their shape. We look at an image dataset consisting of 14,000 women's

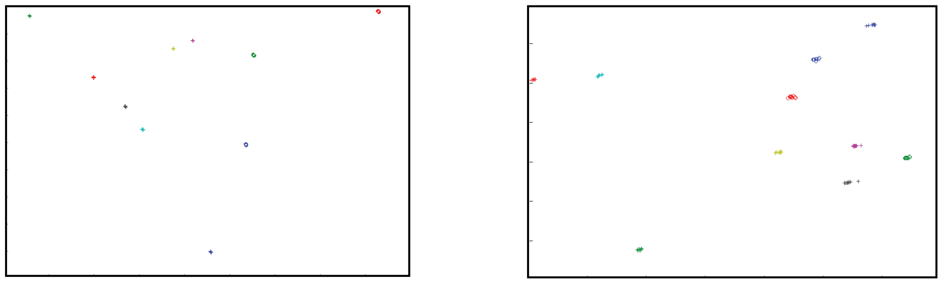

(a) (b)

Figure 3.8: The labeled points in the shoe dataset projected onto 2D using Multi-Dimensional Scaling (MDS). (a) All 3 types of feature vectors have equal weight. (b) Only the texture feature vector is used.

shoes which we would like to classify into 10 different categories. The feature-vector for each shoe includes a SIFT feature-vector [15], a Texture feature-vector, and a Color feature-vector. The number of dimensions in these feature-vectors is 100, 16, and 125 respectively. We calculate the distance or dissimilarity between two shoes as their Euclidean distance in high-dimensional space. To make sure that each column gets a similar contribution, we normalize each column. Given a value $fvec_j(i)$ as the ith column (feature-vector) of the jth object, and $\mu(i)$, $\sigma(i)$ as the mean and standard deviation of the ith column:

1. Subtract the mean, $\mu(i)$ from each entry in the ith column to give *fveci'(j)*
2. Divide *fveci'(j)* by $\sigma(i)$.
3. Add back the original mean so that the final distribution has a mean μ(i), and a standard deviation of 1.

$$normalized\_fvec_j(i) = \frac{fvec_j(i) - \mu(i)}{\sigma(i)} + \mu(i) \tag{3.3}$$

$$final\_fvec_j(i) = \frac{normalized\_fvec_j(i) * w_k}{ndim(k)} \tag{3.4}$$

$1 \leq k \leq 3$, k is the feature vector to which the i^{th} column belongs

We would like the user to be able to again assign the relative weight values to the 3 different feature-vectors. To make sure that one type of feature vector does not

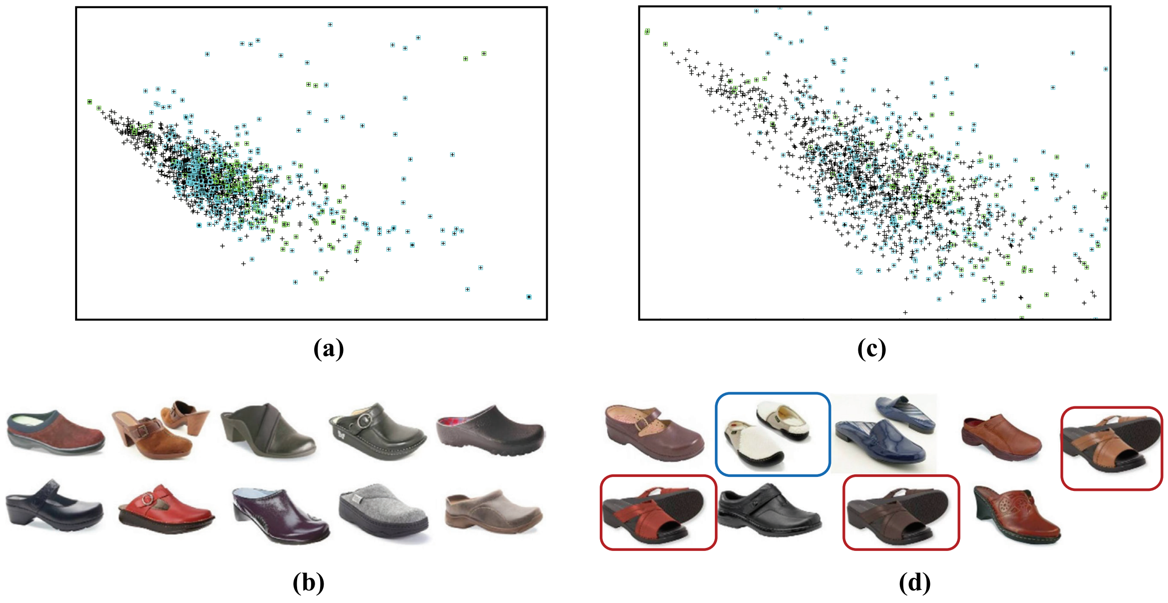


Figure 3.9: Results for exploring the “clogs” subset. (a) Initial classification of the clogs into shoes with-hole and without-hole (b) The top 10 most uncertainly classified clogs (c) Scatter-plot showing the classification after classifying the 1st, 2nd, 6th and 7th shoes in part b (d) The top 10 most uncertainly classified clogs based on the updated model in part c. The shoes highlighted in red are open-toed strapped shoes, and one highlighted in blue has one shoe facing backward

contribute more to the distance matrix value just because it has a higher dimensionality, we need to normalize again (by dividing by the number of dimensions), and then multiplying by the user provided weight.

These feature-vectors are then concatenated horizontally to make a 241 dimensional vector for each shoe. Then a small selection of 100 points is labeled by the user. This is less than 2.5% of the total dataset size. These labeled pointes are run through MDS and plotted in 2D. The user can now adjust the feature weights to improve the separation between clusters. However, for this dataset, the clusters are pretty well separated, as seen in Figure 3.8(a). In fact, even when we set the weights for the SIFT and Shape feature vectors to 0, the clusters still remain pretty well separated as shown in Figure 3.8(b). Next, we fit each of the 10 user clusters to a high dimensional Gaussian model. The presence of 0’s in certain vector spaces prevents normal fitting procedures. However, by adding a small ε-value to the covariance values ensures that they are non-zero and positive. All of the μ and σ values are concatenated into two lists, and a Gaussian Mixture Model (GMM) is created using these lists. This model is used to calculate the posterior probabilities of the total unlabeled points with all points receiving a probability of 90% being added to their respective cluster.

Now, we dig in further into the dataset to further split some of the categories into sub-categories. When we zoom into the cluster containing shoes categorized as clogs, we see that shoes with and without holes are completely mixed up. We start with labeling 30 shoes with holes/no holes. Focusing on the clogs cluster we run the same procedure creating two Gaussian distributions and creating a GMM with their combined statistics. While calculating the posterior probabilities we set the certainty to 100% or 1.0. A certainty of 1 is not unreasonable since points with lower probabilities in high dimensions have been shown to misclassify. From this initial run of the 30 labeled shoes we see the clusters with 392 blue labels (no holes) and 112 green labels (holes). From here the images of the top 10 most uncertain shoes are then displayed in Figure 3.9(c). Labeling the most uncertain shoe to the cluster with no holes produces a new model image (Figure 3.9(b)) containing 417 blue labels (no holes) and 112 green labels (holes). This result demonstrates two important things - first it shows the machine's inability to determine shoes with no holes, as this cluster has not improved. However more importantly it shows the benefit of the user input. By labeling one most uncertain point we were able to increase the "no holes" cluster from 392 to 417 points. Labeling the next most uncertain point (which happens to also be no holes) produces 442 and 112 respectively. So with two labeled points we were able to improve the accuracy of the model by 50%.

In a different run, the user is allowed to label multiple points at a go. Here the user categorizes the 1st and 2nd most uncertain points to the no holes cluster and the 6th and 7th most uncertain to the holes cluster (see Figure 3.9(c)). After relearning the model we obtain the same results with 418 shoes in the no holes cluster and 114 in the holes cluster. This is consistent with the first round of user input. However the new most uncertain points are interesting as you can see in Figure 3.9(d) that there are clearly groups of shoes the program has trouble classifying. Of the top 10 most uncertain there are three open toed strapped shoes (5th, 6th and 8th shoes). There are also two shoes with one facing backward (2nd shoe) giving the appearance of a hole and thus the uncertainty. This shows the visual importance of "visual active learning" so now these types of shoes, particularly the three open toed shoes could potentially be handled differently or at least labeled early on to help improve the model faster.

3.7 Conclusion

In this chapter we have presented a general approach to visualization-assisted model learning for data segmentation and classification tasks. The driving motivation for our approach is that the typical manual tagging of data is very resource intensive. Further, even if one uses just a small dataset for tagging to boot-strap the learning process, if the chosen subset is not a good representative of the entire dataset, then the models learnt might not be robust. On the other hand, completely automated methods which take a brute force approach by using large feature vectors for classification have the drawback of being very time intensive, and small misclassifications can cause the model learnt to be less than ideal. In this case user interaction at various stages can help the machine stay on track, and so will improve the speed as well.

Our other main contribution is giving users control over the weights of various feature vector components. Better feature vectors lead to a good initial clustering. If the feature vectors are bad representatives of the token, the user will have to classify most tokens by hand, thereby minimizing the abstraction of data done by presenting tokens in a clustered format. In able to assign scores to feature vectors, a good approach will be to measure the number of tokens a user has to individually move around between clusters – fewer such operations indicates a better feature vector.

Chapter 4. Iconizer: A Framework to Identify and Create Effective Representations for Visual Information Encoding

Visualization is a powerful means to support analytical reasoning, decision making, and the human information discourse. It is motivated by the fact that humans are able to detect patterns vastly better with visual representations than with alphanumeric ones, which in turn enables more complex reasoning processes. However, currently the degree of semantics encoded in these visual representations is still quite limited. So far, the majority of visual communication occurs by ways of spatial groupings, plots, graphs, data renderings, photographs and video frames. The use of icons as a form of information encoding has been explored to a much lesser extent. Typically graphs in visual analytics applications use standard icons from a fixed icon gallery or an actual picture. Here we describe a framework that uses a dual domain approach involving natural language text processing and global image databases to help users identify metaphors suitable to visually encode abstract semantic concepts. The visual representations resulting from these visual encodings can then be used as stand-alone expressive icons or as clip art in visualization and visual analytics applications. A preliminary version of this work was presented as [27].

4.1 Introduction

Much scientific evidence indicates that visual encodings, if chosen well, are more efficient and effective than textual ones. Their decoding is more time-efficient because human visual processing can rapidly and effortlessly interpret what is seen, make inferences, and explain causal histories [37,52]. In addition, the retention of visually encoded content in short-term memory tasks is higher and visual encodings are more space-efficient. Experiments with a database on terrorist attacks in the 1980s have shown that iconized data occupies 10 times less space than text. Thus the time for an analyst to browse, analyze and search iconized data over textual data can be potentially reduced by an order of magnitude [50]. Along these lines it has also been demonstrated that people can work with multi-dimensional icons twice as fast as with text [87].In fact, some have called visual language a ‘prosthesis’ for some of the limitations of human thought since it supports and augments the severe limitations of working memory [38]. Hence, the potential of visual information encoding is undoubtedly high, but the encodings must be selected wisely to reach these reported performance rates. In this chapter, we present a framework and infrastructure that assists in a user’s effort to select or create good iconic representations for the visual encoding of concepts.

In particular we wish to take advantage of the fact that there exist iconic or canonical views of objects, as has been demonstrated in the psychology literature for human perception. In their seminal work, Palmer et al [67] find that humans agree on canonical views of objects and that recognition is faster for these views. We propose to find representative examples for hierarchies of object classes. However, instead of having humans hand select these canonical images to use as visual encodings − a tedious process when the collections are large − we propose that these biases in preference will emerge naturally when mining large collections of images taken and posted to the internet by people.

Icons (in computing) have been around since the 1970s to make computer interfaces easier to understand for novice users, mapping concepts to standardized visual representations that are meant to be intuitive. However, currently the majority of icons are symbolic representations of applications that need to be memorized by the user. Clip art, on the other hand, aims to be more descriptive and is meant for illustration. Many companies sell copyrighted clip art catalogues, but most people use the free standard clip art libraries embedded in popular applications, such as MS Office and Windows. A very narrow set of clip art is used in practice, often only marginally matching the situation at hand. A third option for selecting iconic representations is to use web search to gather illustrations or images fitting a desired concept. However, most of the time, especially for more complicated or specific concepts, this results in limited success because multiple queries or lengthy searching must be performed to match a concept exactly. The framework we present provides a computer aided system that allows users to quickly and effectively design clip art that is well targeted to their particular concept of interest, in a style matching their personal preference (or that of their intended audience). In other

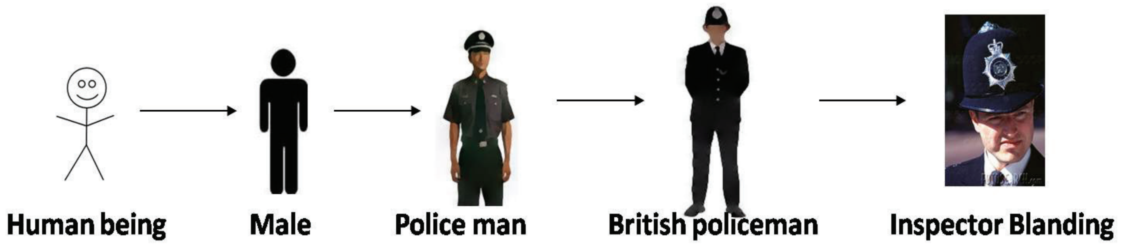


Figure 4.1: Specialization from an unknown person to Inspector Blanding

words, our system can make every amateur a decent clip art designer, with a unique personal style. We achieve these capabilities by extending and synthesizing techniques rooted in non-photorealistic rendering and computer graphics, image processing, web-scale content-based image retrieval and natural language processing.

The ability to design well-targeted expressive clip art in a cohesive illustrative style enables applications at a scale much grander than singleton icons (We use the notion of icon and clip art interchangeably – both are considered to represent visual information encodings). We may use them for the illustrations of documents, books, manuals, and the like, and they are also applicable to visualize taxonomies of objects and even more general concepts. Further, they can replace or at least complement textual annotations within mind-maps or the node links diagrams often used in analytical reasoning tasks, making these representations exceedingly more expressive. Likewise they can also augment social network graphs, where commonly a person is just represented as a generic icon labeled with a name, say 'Bill' (or a picture of that person). These graph representations show little or no adjunct information pertaining to the icon's subject. We may ask, what other relevant information do we know about Bill that could help us in making further conclusions about the information conveyed within the graph? Is there any missing information worth investigating or mining the data for? The major problem with this generic representation is that it cannot visually express this pertinent information, or the lack thereof.

This chapter is structured as follows. Section 4.2 presents the general philosophy behind our methodology, which is rooted in a joint lexical and visual analysis. Section 4.3 presents previous work in the area of visual languages and icon generation as well as the background of our approach. Section 4.4 describes our approach in detail, and the results are presented in Section 4.5. Section 4.6 presents some discussion on our system, and Section 4.7 ends the chapter with conclusions.

4.2 Overall Motivation and Philosophy

We aim to find Visual Information Encodings (VIEs) that are intuitive, that is, are already part of one's visual vocabulary. This avoids the need for laborious memorization of a set of dedicated symbols for iconic communication, which in fact has prevented visual languages from reaching mainstream applications. VIEs are relatively easy to find for most objects and some actions because they can be observed in real life and are already part of one's visual vocabulary – yet their interpretation and aesthetics still leave much room for artistic freedom in determining the best VIE design. However, as with visual languages, the greatest challenge comes from determining good VIEs for abstract concepts. Take for example the concepts 'travel'. When asked, people will offer a wide variety of possible VIEs for these concepts, and such variety is also reflected in the query results with popular image search engines, such as Google Images. We hence desire a VIE that reaches the broadest consensus among a given, sufficiently wide population. One way to measure such consensus is via formal user studies, which unfortunately is infeasible on a broad scale given the large variety of abstract concepts and candidates for their VIEs.

We propose a different approach, circumventing the need for an active solicitation of user responses to specific candidate VIEs. Instead we exploit existing public lexical databases augmented with aggregated statistical information in the form of lexical triggers and couple these with public image search engines. These triggers are computed by analyzing thousands of documents and looking for commonly occurring co-occurrence relations (words that often appear together). Thus we resort to employing methods that statistically analyze data that humans have produced (in place of interrogating them directly) and which in some form represent their view of the world. These relations in essence often provide an encoding of relations between words. Examples are 'sky → blue' or 'travel → passport'. These sorts of concept relations are not encoded in traditional semantic databases like WordNet [23], but are to some degree captured through statistical analysis of large textual corpora. Nevertheless, this knowledge is still far from complete,

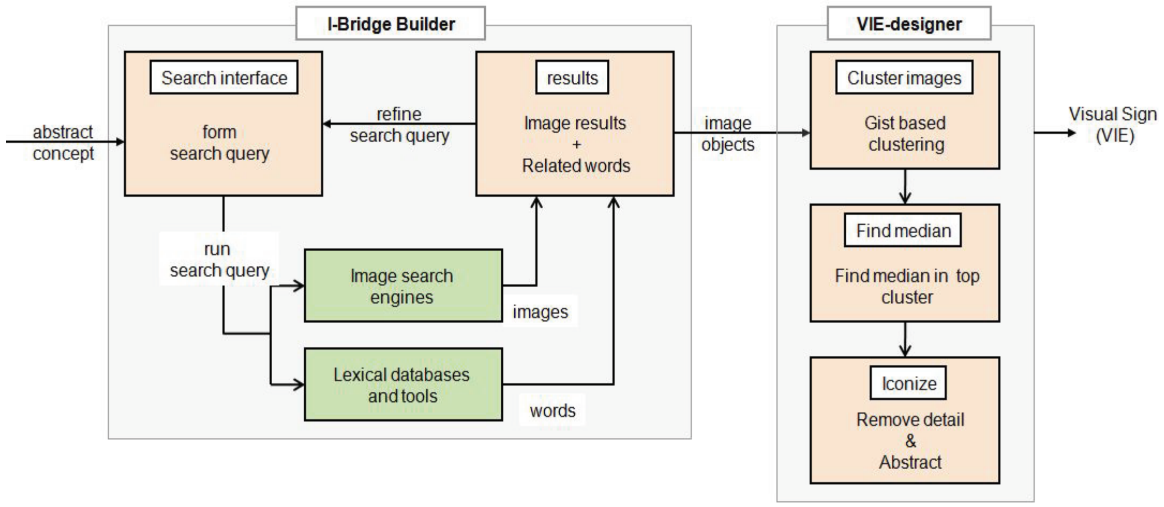


Figure 4.2: System block diagram

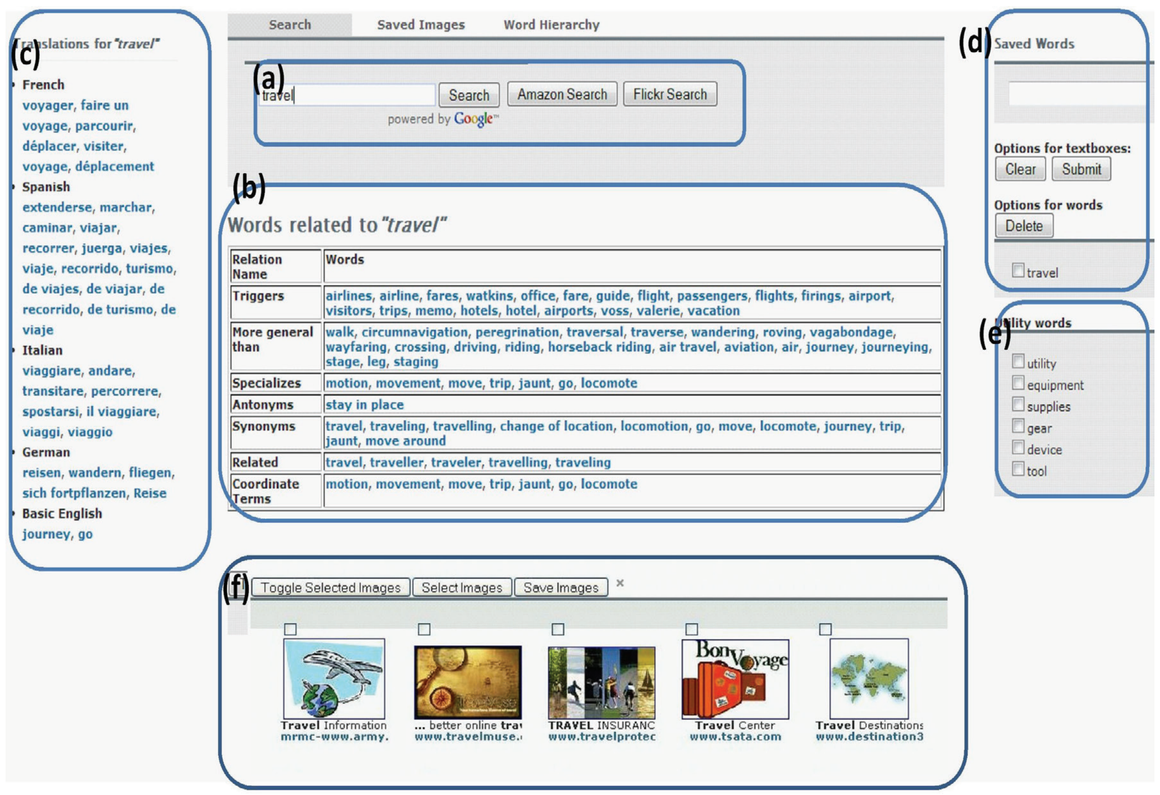
and in addition, we require a similar notion on the image side, that of visual triggers, to map concepts to suitable VIEs. This notion of a visual trigger is not readily available from any current image database. We propose a dual-domain approach to (incrementally) build and format these visual triggers. We note that this is an extremely large undertaking as a whole and all we can do here is to propose a methodology by which this could be done. Our approach uses the available lexical triggers to allow users to interactively explore concept space, select suitable representative concepts, and finally create icons.

As mentioned, concepts can range from very specific to fairly general. Figure 4.1 illustrates this via an example that shows a conceptual zoom across many conceptual levels, here from ‘Man’ all the way to a specific person ‘Inspector Blanding’. First, we may only have data supporting a certain level of concept semantics (we may only know that the person is a policeman of unknown nationality), and the subsequent lack of visual information might instigate analysts to obtain this missing. Second, we may only require information depiction at a certain semantic level. For example, the intent may be to only depict the level’s categorical information, or the depiction of any further detail may simply consume too much pixel real estate. Third, we may be required to hide (or anonymize) further detail because it is classified. Therefore, we provide a hierarchy of possible visual encodings and allow the user to select the level of specificity they wish to depict.

It should be fairly clear that these conceptual zooms are not just multi-resolution representations obtained by low-level abstraction, that is, by intensity or gradient domain filtering. Rather, they are semantic zooms (categorical refinements or generalizations within an object hierarchy). Therefore, we require semantics-informed abstractions. This represents an entirely new dimension to image filtering and abstraction. Current image abstraction methods use low-level image processing methods and are unaware of the object pictured in the image. They retain the strong features and remove weak detail. In contrast, we desire to retain the semantically strong features, that is, those that appear in many instances of a certain object category, and remove the semantically weak features. Again, this requires establishing some amount of visual understanding of object categories, but in this case we are less prone to error since the vaguer the concept the more semantic detail is abstracted away.

4.3 Related Work and Background

Visual languages range from icon algebra [44] to the encoding of all information into multi-frame artist developed cartoon-like renditions [13]. The set of icons is typically fixed, developed manually, and can be composited. Setlur et al. [79] propose the innovative notion of Semanticons to create new file icons by abstracting terms occurring in the file or file name along with a commercial database of images. None of these applications exploits any semantic analysis, nor do they make use of the large publicly available lexical and image databases to broaden the semantic base for abstraction and enable VIE learning from these rich resources. Other related work includes that of [75] who enable the automatic creation of collages from image collections, aiming to compose a single image with blended collection highlights. Here the user has no control over the layout of images inserted into the collage. Alternatively, Photo Clip Art [46] provides an interface that enables insertion of photo-realistic objects into new images, correctly constrained to be in a natural looking context within the resulting image. Their goals however are different than ours in that their visual objects are meant to enrich or compose graphical scenes and collages, without placing special emphasis on conveying specific semantic information. Finally, also related to our work is research targeting the automatic illustration of text, which however has mostly been done via application of 3D graphics



Translations for *"travel"*

- **French**
  voyager, faire un voyage, parcourir, déplacer, visiter, voyage, déplacement
- **Spanish**
  extenderse, marchar, caminar, viajar, recorrer, juerga, viajes, viaje, recorrido, turismo, de viajes, de viajar, de recorrido, de turismo, de viaje
- **Italian**
  viaggiare, andare, transitare, percorrere, spostarsi, il viaggiare, viaggi, viaggio
- **German**
  reisen, wandern, fliegen, sich fortpflanzen, Reise
- **Basic English**
  journey, go

Words related to *"travel"*

Relation Name	Words
Triggers	airlines, airline, fares, watkins, office, fare, guide, flight, passengers, flights, firings, airport, visitors, trips, memo, hotels, hotel, airports, voss, valerie, vacation
More general than	walk, circumnavigation, peregrination, traversal, traverse, wandering, roving, vagabondage, wayfaring, crossing, driving, riding, horseback riding, air travel, aviation, air, journey, journeying, stage, leg, staging
Specializes	motion, movement, move, trip, jaunt, go, locomote
Antonyms	stay in place
Synonyms	travel, traveling, travelling, change of location, locomotion, go, move, locomote, journey, trip, jaunt, move around
Related	travel, traveller, traveler, travelling, traveling
Coordinate Terms	motion, movement, move, trip, jaunt, go, locomote

Figure 4.3: Overview of the user interface. (a) Query-box (b) Related words (c) Translations to foreign languages, and Basic English (d) History + query builder (e) Utility words (f) Image results

engines. The systems by [31,32] annotate their renderings according to the underlying text, while Word-Eye [16] analyzes descriptive text according to a set of hand-coded rules to produce 3D scene renderings. All systems require a library of 3D models to which their illustrations are limited. The Story-Picturing-Engine [JWL06] has a greater gamut, indexing image search engines with exact lexical terms occurring in the text, but no further semantic interpretation, integration, or visual abstraction is made.

Visual signs are at the heart of visual communication. Resemblance Theory [61] measures the degree to which observing a sign initiates a perception process resembling the perception carried out when actually viewing the depicted object. It is clear that no picture can ever produce a perception exactly like that produced by the portrayed object. For example, a picture of a person can never fully describe the real person – it can only show certain aspects or properties which remind us of the depicted person or what he/she stands for (for example a picture of Andy Warhol may signify just him or the entire pop art culture). In the former case the picture is described as being iconic, while in the latter it is indexical [82]. The recognition of either assumes knowledge and experience on the viewer's part. Such recognition processes are triggered by observing any pictorial artifact, be they symbols, data visualizations, or real-world images. Of these cases, it should be obvious that symbols and data visualizations tend to be more arbitrary and thus more difficult to index and harder to recall (at least for non-experts), while real-world images are more readily accessible, intuitive, and requires less learning.

One high level challenge we face is how to find representative or iconic images for a given query term. For this purpose we can exploit the vast number of images available on the Internet – an extremely rich source of visual data that must be carefully sifted to find images relevant to a user's needs. The sheer number of images presents challenges not experienced in traditional well organized or labelled collections of images. Recent work has explored methods for choosing the most representative or canonical photographs for a given location [81] or monument [52] through clustering, where hard geometric constraints can be used since the object is a single instance seen from multiple viewpoints. These methods are less useful here since we apply our system to object categories that vary quite widely in appearance. Other methods postulate that the most relevant images for a search term tend to be those that are more aesthetically pleasing, since returning a poor quality image is probably never aligned with a user's needs [72,19]. We propose a human-computer integrated approach that combines textual analysis with image clustering techniques and human aided image selection.

Another area with a similar high level goal to our work is the problem of image classification for content based image retrieval (CBIR). These systems utilize purely image based information for content analysis and retrieve images by measuring their similarity to a given query image (an in depth review is presented in [18]). The most successful approaches typically integrate a variety of color, texture, shape, or region based cues. However, the problem of content based retrieval is extremely challenging and far from solved for most object categories, and these systems are often helped by having a user in the loop to guide the search process via relevance feedback [30,34,104]. We take such an approach here.

4.4 Approach

Our overall system is depicted in Figure 4.2. It consists of two main components, the I-bridge builder and the VIE designer. As mentioned, many abstract or general concepts do not have direct visual encodings, and thus have low iconicity. If our concept has an iconic sign then our job is fairly easy. Else, we require an indexical sign (I-sign). I-signs must use a good ontological metaphor by which the abstract concept is represented as something concrete, such as an object, substance, container, person, or some visual action. Good metaphorical mappings improve distance relations in conceptual space, moving the concept closer to visual decoding. The first part of our framework is designed to build this bridge crossing conceptual space – we call it the iconicity-bridge or I-bridge. We use large lexical databases to derive potential I-bridges (the language-based I-Bridge Builder in Figure 4.2). Our second step is the I-sign's illustrative abstraction into a VIE that represents the most central visual theme of a concept (the graphics-based VIE-Designer). This involves clustering the images, finding the median in each cluster, and finally abstracting the median so that it captures the common features of all objects in the cluster. In the following we describe each of these components in detail, followed by additional components motivated by rules formulated in visual semiotics.

4.4.1. The I-Bridge Builder

A good I-bridge is an association that is deeply rooted in our semantic understanding of the world. Such associations consist of pure lexical classifications, such as synonyms, antonyms, hyponyms (specializations), and hypernyms (generalizations) as well as statistical knowledge about co-occurrence relations between terms in documents or spoken language (so-called trigger relations). The former is captured in public lexical databases such as WordNet [23], while we use Lexical FreeNet [7] to provide the statistical knowledge component. Basic English [65] gives us the equivalent of a word in restricted English – e.g. 'bombshell' translates to 'great shock'. It also provides information about the direction in which an adjective or emotion becomes more intense. Utility or helper terms are also provided which when combined with the original query often help narrow image results to a particular visual meaning. For example utilizing the 'gear' utility with the query 'travel' might direct a user toward images of suitcases or backpacks. Lastly language translators are used to translate concepts that are polysemous (have multiple meanings), or are brand names in English.

In our interface (Figure 4.3), all of these lexical and statistical associations are exposed to the user as tools for enabling conceptual connections and exploring the space of an input concept. This results in a powerful interactive interface to help the user build an effective I-bridge. Figure 4.3(a) shows the query input box. Figure 4.3b displays semantically related words from WordNet and statistically related words from Lexical FreeNet. Figure 4.3(c) provides translations of the query concept into four languages plus Basic English, while Figure 4.3(d) maintains a history of the explored query concepts during an I-bridge building session. In Figure 4.3(e) helper terms such as 'equipment' or 'tool' are provided to enable focusing on particular visual senses.

Finally, Google image search is simultaneously used to discover images or objects commonly associate with the current query – our visual triggers. For example, the top results for the term 'travel' might contain images of airplanes, suitcases, beaches, and maps. These image triggers are the visual equivalent of textual triggers in the sense that they are visual representations that tend to co-occur with a query, as compared to textual triggers which are words that tend to co-occur with a query. Figure 4.3(f) displays the top results from Google image search for the current query.

After the initial display of results, the user can do several things. In a perfect world, he would find several relevant images in the first go and store them to the saved images panel. Otherwise, he can continue browsing the lexical space until the images reflect his desired concept. Our interface provides an abundance of ways to browse this space. For example, the user might want to explore concepts semantically or statistically related to his query (Figure 4.3(b)). Alternatively, he could use the query builder to make complex queries. For example, suppose the user has submitted a query of the word 'art'. The results returned might being too 'artsy' for his taste and not represent the high level concept 'art' in a simple, concise manner. An image query of 'art' + 'supplies' (a utility word from Figure 4.3(e)) gives more concrete results such as images depicting colored pencils, crayons or paint. Combining multiple search terms in the query input box can also be quite useful. For example, the word 'climb' triggers 'ladder', but an image depicting a ladder in isolation does not clearly represent the activity of climbing.

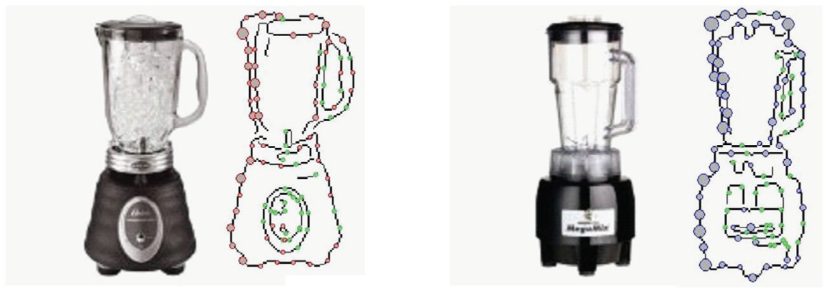
Whereas, a combined search for ‘climbing + ladder’ results in Google image results showing people climbing ladders. Also, the image search results themselves can suggest good I-bridges. For example, in the image for travel, the image results include an airplane, a map and compass, suitcases, and a person on the beach. Depending on the person’s perception and preferences, he might choose one of these visual senses to use as the I-sign candidate, or further explore a concept indicated by the trigger by performing additional searches.

In order to construct a good VIE for the user’s candidate I-sign, we will require a large number of relevant images to mine for the most iconic visual representation. In our experience with the I-bridge builder (and Google Image Search) we have observed that more specialized queries tend to return image sets that are more coherent. For example, the query ‘man’ returns a diverse set of images with many depictions, while the image set for ‘police man’ is more homogeneous. This reveals a powerful strategy for I-sign learning: Join all the image instances obtained with specializations of the target term, obtained through our semantic and statistical textual analysis, and then use this collection to build the VIE. Finally, more descriptive queries can be derived by examining the information stored in the search concept’s frame and those connected to it. We explore these and other strategies to derive a comprehensive query formation framework for robust I-signs.

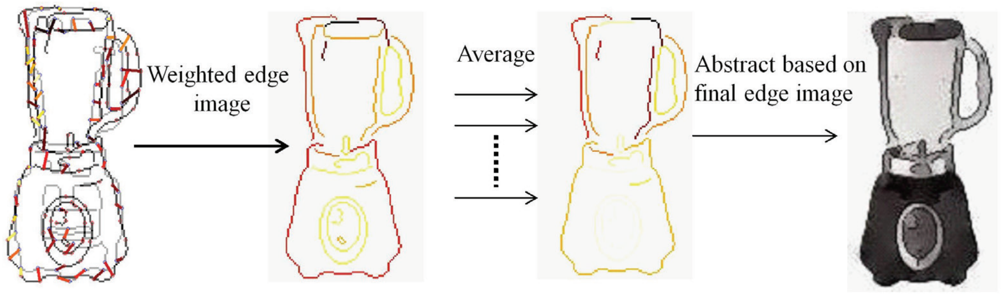
As mentioned previously, we use Google Image Search as our image source. To get a diverse and comprehensive set of images for a query, we download the top 200 results. Though we could go further down in the ranking, later results tend to become much more noisy and unreliable. Instead, to increase the size of our data set while maintaining high quality, we translate our queries into 4 other languages and collect the top 200 results from each translated term.

4.4.2. The VIE-Designer

Constructing visual equivalents of model-based abstractions requires a semantic abstraction of these I-signs, which as mentioned above goes much beyond the image-based abstraction methods available today. More concretely, we seek a picture of the given concept that unifies all of the concept’s known facts (within the current world context), but abstracts away the unknown facts. In images, a ‘fact’ is expressed as some



(a) Images with corresponding edge images. The edge images have the random points highlighted. The green points are the ones with no good match.



(b) Given the one-to-one matching between points on the two edge images, we create a weighted edge image for the median. Repeating this with other blenders gives us the average weighted edges. This helps us design the final "icon"

Figure 4.4: This figure demonstrates how the edges of the median image are matched to the others to find the edges which match across objects

visual feature, or collection of features. Let us define the notion of feature noise. An image set that bears such noise is a set of images that share some features (facts), but also contain a wide selection of other random features (unknown facts). We can construct an average or exemplar image for a category by looking for features in common across a set of queries. For example, we may only know that our suspect is a 'male, blue-collar worker', so we could formulate a corresponding search query of this term, and in addition also use our lexical resources to retrieve images of various instances of the term, here mechanics, plumbers, electricians, and so on. The noise can then be removed by finding the level of image-based abstractions at which all images appear similar.

In this section, we present our algorithm to extract the basic icon for a set of images. Given a set of images belonging to the same category, we can find the most common shape features among them, and produce an icon.

Initially we calculate the gist feature vectors of all the images. As a pre-processing step, we use the gist feature vectors to find duplicates, and discard them. Next, we cluster the feature vectors and select cluster exemplars using affinity based clustering

[24]. This method requires us to enter pair wise similarities as well as the "preference" for a particular image to be an exemplar, i.e. a cluster centre. The Euclidean distances between the gist feature vectors serve as the dissimilarity values. For the "preference" values, we use the Google Image Search rank, since images appearing earlier in the ranking tend to better reflect the search term and be of higher quality. Given the resulting clusters, we order the images within each cluster based on their similarity to the exemplar. Finally, we select the cluster used to build our final VIE. This choice is based on two criteria – the size of the cluster and the average distance from the exemplar.

As mentioned above, the resulting VIE should preserve the details present in all examples of this VIE while removing those details specific to any particular example of the VIE. Thus, to construct a final VIE, we build on the exemplar from the selected cluster and utilize image processing techniques to abstract away unnecessary details. More specifically, we represent the exemplar and each image within the exemplar's cluster with a set of 100 random edge points described using shape context features [8]. Next we perform shape based alignment between the exemplar and each of the images within the same cluster using the Hungarian algorithm to find the optimal one-to-one matching between the point sets (Figure 4.4(b)). We then remove bad matches based on: a) low feature similarity, and b) points that are matched across large distances in image space (indicating false matches between points in very different parts of the object).

At this point, we are left with point pairs that are highly similar. We assign scores to the examplar points based on their similarity to the matching points in the other non-exemplar image. Further, the remaining points on the exemplar edges are assigned scores equal to the nearest exemplar point on a connected edge. This gives us a complete weighted edge image for the exemplar. We show an example for blender in Figure 4.4 - displaying the original edge image, matched edge points, and final weighted edge image. In this figure, the color map goes from white to black via yellow and orange. We can see here that the outer edges are red/orange and black, indicating edges present in both the exemplar and the non-exemplar image. We repeat this step by matching the exemplar to all the other images within the exemplar's cluster. In the end, the final weighted edge image is the mean of the weighted edge images calculated to each non-exemplar blender.

The final weighted edge image helps us design the abstracted version of the object. We use Poisson-blending [69] guided by the noise-free edges to create an abstracted illustration [66,88] by removing the features at higher levels. Finally, we add back the edges calculated in the previous step to give it a more defined and iconic look.

4.5 Results and Applications

Figure 4.5 demonstrates the exemplar-finding algorithm (we have already illustrated the population-driven exemplar abstraction in the section above). Here we use the problem of finding the average car within a collection of many cars. We find the following nine subcategories for a car, since these represent car categorization by shape (and not size alone): sedan, coupe, convertible, sports car, SUV, van, pickup truck, station wagon, and minivan. For each category we use GIST-based affinity clustering to get a set of representative visual triggers. The user selects the best cluster(s) from each category, and finally we cluster together the top ranked selected images using our shape context based clustering. This gives us the average car and shows the utility of clustering at multiple scales. At the end of clustering, a sedan emerges as the average car.

Next, in Figure 4.6, we demonstrate our entire algorithm (exemplar-finding and abstraction) for the construction of taxonomy visualization. Here we seek to assign an icon to each level, instead of just to the leaves. We first calculate the icons for the leaf nodes using the algorithm outlined above. For the inner nodes with only leaf nodes as children, we form a collage of at most the top four children under it. This order can simply be calculated based on the popularity – for example on Amazon.com, the sub-categories always appear in the order of popularity. For example, when someone selects the category "Small Appliances", they are probably looking for a Coffee maker. Further, when forming the collage, the more important categories are allocated more space. As we move further up the taxonomy tree, one node will have many subcategory trees underneath it. In order to keep the icons compact and representative, we just percolate the icon for the top sub-category upwards. For example, under small appliances, "Coffee tea and espresso" is a subcategory. Under "Coffee, tea and espresso", Coffee-maker is the top subcategory. So we percolate the icon for the Coffee-maker upwards, and it is used in forming the icon for "Small Appliances".

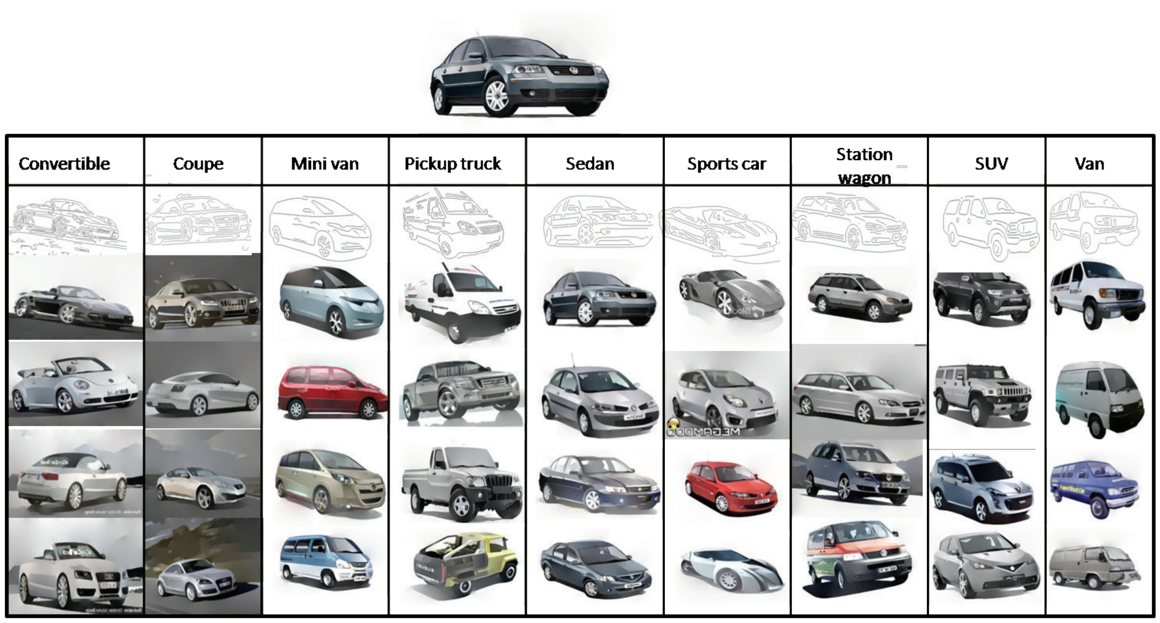


Figure 4.5: Icon from car – it is built by finding the combined median of the 9 sub-categories shown here. The images under each sub-category are the top results for that car. The top row shows edge images at the scale at which the different cars tend to look most similar.

The final figure, Figure 4.7, presents a collection of results and extensions, using a social network / node-link diagram example. It compares a standard graph seen online and in various publications dealing with social network data with a representation derived using our system. Graphs and networks have become increasingly ubiquitous with the appearance of internet-based social networks and databases and also within visual analytics applications. Current work on graph drawing in these applications has mostly focused on coding relationships via spatial arrangements and less on the visual encoding of the semantics associated with the nodes and links. Typically, text, color, and line properties are used to label nodes and edges, augmented with generic icons of limited semantics (such as the factory icon or the group-of-people meeting icon) or photographs (such as faces of terrorists or pictures of locations). The icons generated in our system can help overcome these limitations by placing more expressive icons in the graph. The example shown in Figure 4.7 on the left is a simple social network graph involving three people, their personal attributes, and the relations between them. The three people are – Jon, Ben, and an unknown football player. Among the attributes we see that the addresses of Jon and Ben are known. The relations show that Jon and Ben exchange a call, and that

Jon admires a certain (unknown) football player. The graph on the right adds more information icons derived via various means and extensions of our framework, which we shall explain in the following.

Generalization of sub-categories: We assume that we do not know what Jon and Ben look like, but we know their professions, so we can show the icons for those professions. Jon is a mechanic which does not reach a good consensus in the retrieved images. So we first clustered images of mechanics, plumbers, electricians, and the like, and then used our exemplar image abstraction framework to determine the average mechanic. Similar we can find a VIE for Ben, who is a doorman.

Symbolic processes: Symbolic processes [44] can encode what a participant means or is. It may show a participant posing for the viewer, along with an icon symbolizing the associated information. We can use context to modify the meaning of an icon and provide focus. In our example, Ben is a doorman by profession, and is also very intelligent. The notion of "intelligent" lexically triggers "scientist" which in turn triggers "physicist" and finally "Einstein". Only "Einstein" is a good I-bridge to "intelligent", all others yield highly mixed image results. To guide the focus on "intelligence" we use another trigger, "brain" to yield the final VIE used in the graph. Jon, on the other hand, is a sports car mechanic, which is symbolized with the appropriate object (a sports car) added to the background. Finally, the "admires" relation does not yield a good I-sign, but we can establish, via the lexical databases, that it is a weaker form of the extreme case, "worship" (see below for more detail). This term gives rise to many images of people with hands thrown up into the air. This, however, does not map back to worship unambiguously, even though it consistently appears among the top results of the image search. Our user studies revealed that it reminds people of different things like celebration, relief and so on. To map this image to worship, we used the technique of symbolic processes and provided context by adding the image of a church in the background. Now it looks more like people thanking/praising God. The technique of expressive augmentation, described below, is then used to downscale "worship" to "admire".

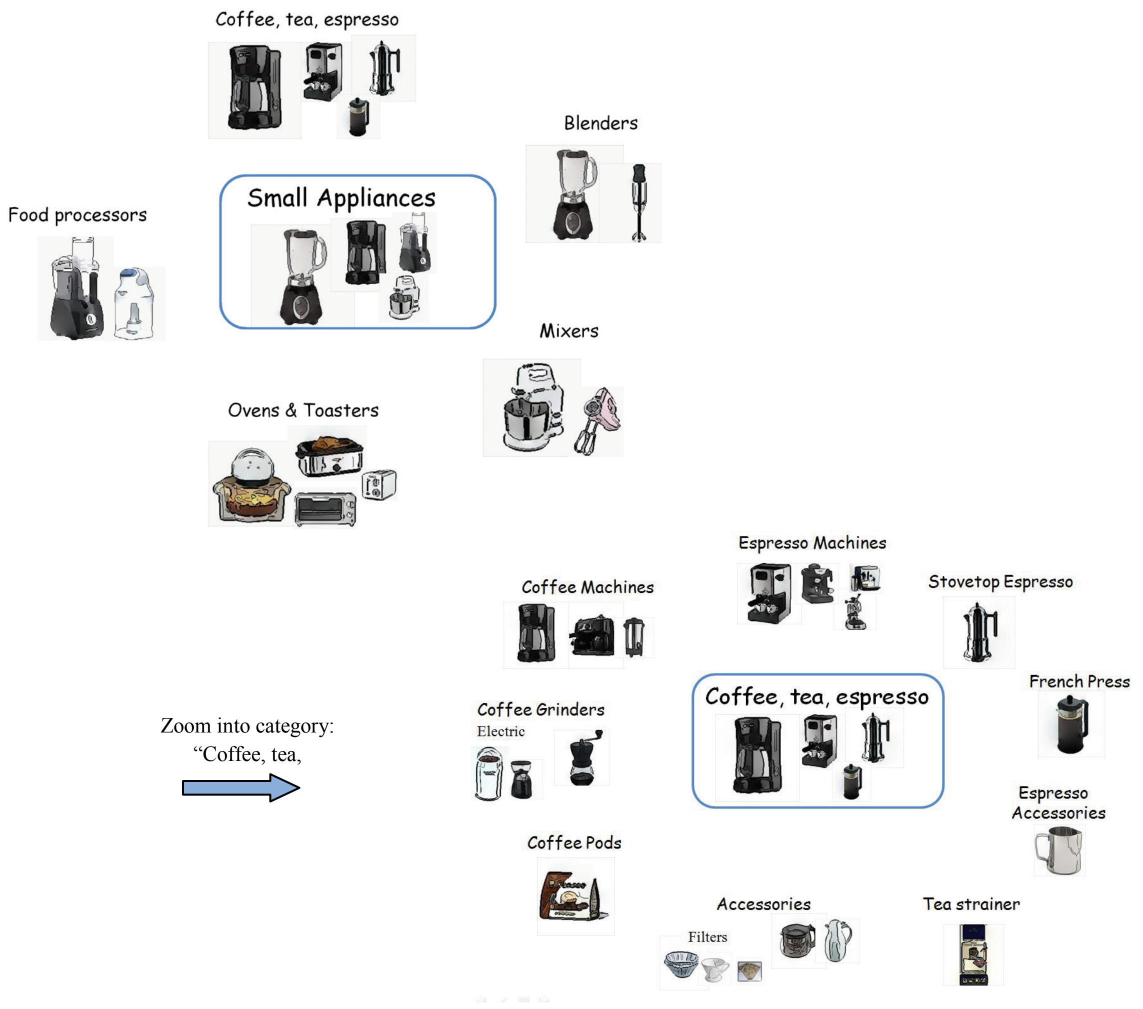


Figure 4.6: Icons showing the taxonomy under the category "Small Appliances". On the right, we zoom in to see the icons for the sub-category "Coffee tea espresso"

Expressive augmentation: We require a fallback method to step in when no method can reach a good consensus. This can occur for example, for subtle semantic nuances within a given concept and for many non-action terms. Spoken language is rich in these, consider for example the difference between "worship" and "admire". Both are positive feelings towards a person in the sense of "liking", but the former is significantly stronger. Some differences can be brought out via illustrative effects but others cannot. We therefore include scale icons (a simple graphical volume indicator) into our renditions, which can indicate the strength of a certain concept with respect to a base concept. We define as a base concept the widely acknowledged root of a given target concept. Basic English can be used to derive this root (which can then be specialized via

WordNet or Lexical FreeNet). We then use the number of single-term levels traversed in the hierarchy as a measure of added strength, assuming that single-term specializations are linear in strength (we do note that this will not always be correct and is subject to personal preferences). The strength of a given term with relation to a base term can be shown by either using an explicit scale, or by changing the size of the base icon with respect to a bounding box. In our specific example, we found that "admire" is halfway between "worship" and so we set the scale to mid-level in Figure 4.7.

4.6 Evaluation and discussion

Our system requires user input and interaction at various stages of the VIE design. To get insight and feedback from multiple people, we had members of our lab interact with the system.

The users identified different concepts they were interested in visualizing, and we built the I-Bridge using our system. Some interesting results were:

- Success → Person climbing a ladder (via google images)
- Oil spill → Images of the duck in the oil spill. This is an example where a current event highly modifies the most relevant icon.
- Renaissance → Triggers artists like Shakespeare and Michelangelo associated with that era. We can represent the concept using either the people, or (especially in the case of Michelangelo) the pieces of art created by them.
- Gothic→ We can represent this concept by using an icon with a person in gothic attire (via Google Images), or by using examples of gothic architecture (via Synonym relation)
- Affinity → Its synonym *kinship* gives us images of family trees
- Countries → The associated terms and images give us the political map, flag, and landmarks (Taj Mahal for India, Great Wall for China) of the country.
- Thrill → Rollercoaster. This indeed represents a good icon for representing the experience of a *thrill.*

During the course of exploration, we identified some shortcomings of our approach, and subsequently devised possible ways to overcome them. The symbolic process and expressive augmentation described above were one outcome of these explorations. As mentioned in the discussion on symbolic processes, the meaning of an icon changes depending on its context, and in fact many concepts can map to the same icon, but it is up to us as designers to make sure that the final visualization contains the right context. As discussed for the concept "worship", Google Image search gave us a set of similar looking images, showing a person with hands thrown up towards the sky. However when looked at as an icon by itself, it might represent anything to a user from relief to extreme joy to celebration. The way to correct this was to add a place of worship (church, temple etc.) as a contextual (symbolic) object to the background.

Another discovery made was that when we get cluttered images or images which have no consensus, we can still attempt to find an icon by taking the route of specialization. As mentioned before, specialized queries often give better image results, and we can combine the results from the various sub-categories to find an icon for the main category. *Car* is a good illustration of this. Even for a simple object like car, the image search results tend to be highly noisy since *car* is such a general, and commonly used term, but by employing specializations like SUV, or convertible the results get much better.

In general, the system works quite well for natural objects or manmade objects found in natural settings like flowers, monkeys, houses, butterflies, etc. However, the drawback with such categories is that the results rarely appear on clean backgrounds. If our VIE must look similar to icons or clipart, we need good methods to separate the object from the background. Since in natural images, the background is not usually very clean, such a process requires human interaction as well. Images obtained in shopping and product databases such as Amazon.com and the like are usually of good quality in this regard.

4.7 Conclusion

In this chapter, we presented an approach which can accomplish the goal of finding a good visual information encoding, or simply, an icon for a concept we are interested in. Such a goal requires the integration of many fields – linguistics, vision, computer graphics, and user interfaces, with a human in the loop. We started by building a bridge from the concept to an I-sign which can represent the concept well.

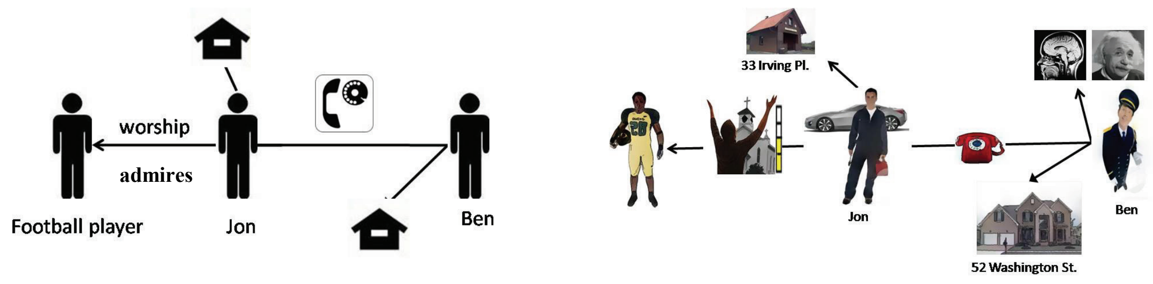


Figure 4.7: The graph on the left shows a commonly seen social network graph. On the right, we show a graph using various icons found using our technique, including symbolic illustration

Given the I-sign, we needed to test if it indeed can result in a Visual Information Encoding (VIE) for the concept by testing for the presence of clusters of images, where each cluster represents a set of visually similar images. If this set of visually similar images exists, we use image abstraction to design our final VIE. At this stage, the user has to again look at the icon, and decide if looking at it triggers the original concept. If this step fails, then it means that some relevant context needs to be added.

We believe that our framework has great prospects in the design of clip art for various applications, such as taxonomies, book illustrations, and the expressive augmentation of graphical node/link diagrams to make these much more engaging and informative. In future work we plan to fully integrate our framework into the graph drawing engine, use abstractions more freely to summarize certain facts and attributes, and use compositions for compact visual story telling with background context and foreground key players. The art of visual advertising and visual semiotics has much to teach us in this in this respect and we plan to extend our work further into these areas.

Given the current status of implementation, we believe that we can deploy our interface and backend processes to a wider circle of users, over the web. Such a community-driven effort will likely result in much more robust icons, and further insight into personal preferences as well. For this study we believe that the conjoint-analysis user study framework recently described in [30] will represent a helpful evaluation suite. We plan to evaluate both usability and performance, in a conjoint manner using three types of experiments – determining I-signs given textual concepts, choosing between two I-signs for a textual concept, and finally given an I-sign, choosing among two alternative textual concepts. Thus, some users will be identifying and composing I-signs while others will be judging them. This will enable us to gain bi-directional insight into the process.

In addition, we also believe that an important theoretical aspect of our work is that it will enable significant strides to be made in the establishment of visual triggers for visual icons and metaphors. As mentioned, this is vastly under-explored territory.

Nevertheless, it goes undisputed that not all concepts have good visual representations and encodings. Sometimes a textual description serves just fine since it, too, can be considered a pattern of symbols, which has been learned very early on in life. This is particularly true for difficult non-object concepts such as “worship” which we have explored in our results. It may be more convenient to simply write this as a sequence of letters. For those concepts that do lend themselves well to visual encodings, we believe that the power of our approach is its ability to communicate possibly quite subtle differences much more efficiently than textual descriptions. Good examples for this are taxonomies, as we have shown with our “small appliances” case study. Finally, our approach also shows its strength when different concept attributes are combined, as for example, to visually encode the concept of “business travel” as a briefcase, “budget travel” as a backpack or a worn-old suitcase, and “relocation travel” as a large suitcase.

Chapter 5. Magic Marker: A Color Analytics Interface for Image Annotation

This chapter presents a system that helps users by suggesting appropriate colors for annotating an image. The color distribution in the image regions surrounding the text area determines what color makes a good choice – i.e. is both eye-catching while glancing at it, and clearly legible while focusing on it. We use a modified version of the perceptual color space CIE-LAB as a basis of our calculations and visualizations. Each point in this space is assigned a preference value called p-map (preference map), where colors with higher values are better choices. This tool works like a "Magic Marker" in the hands of a user, giving them the power to automatically choose a good annotation color, which can be varied based on their personal preferences.

5.1 Introduction

Annotation of an image is a commonly encountered task. It is a process people use to create art, advertisements, presentations or educational tools. While using popular image processing tools like Photoshop, Gimp, etc., users employ a trial and error method before arriving at a suitable choice. None of these tools guide users by pointing out the appropriate set of colors. The most common method to accomplish this task is by choosing colors via the HSV color wheel.

This chapter presents a solution to this annotation problem by designing a user interface, which works as a guide to the user in this color selection task. We build an intuitive user interface which can be easily used by a non-expert, and base our calculations on well known color perception paradigms. While choosing an annotation color, we try to ensure that in the final image, the added markers like text, arrows, boxes etc stand out from the background. Our framework captures known color design rules to form a grading scheme, which along with user preferences can help derive appropriate colorizations.

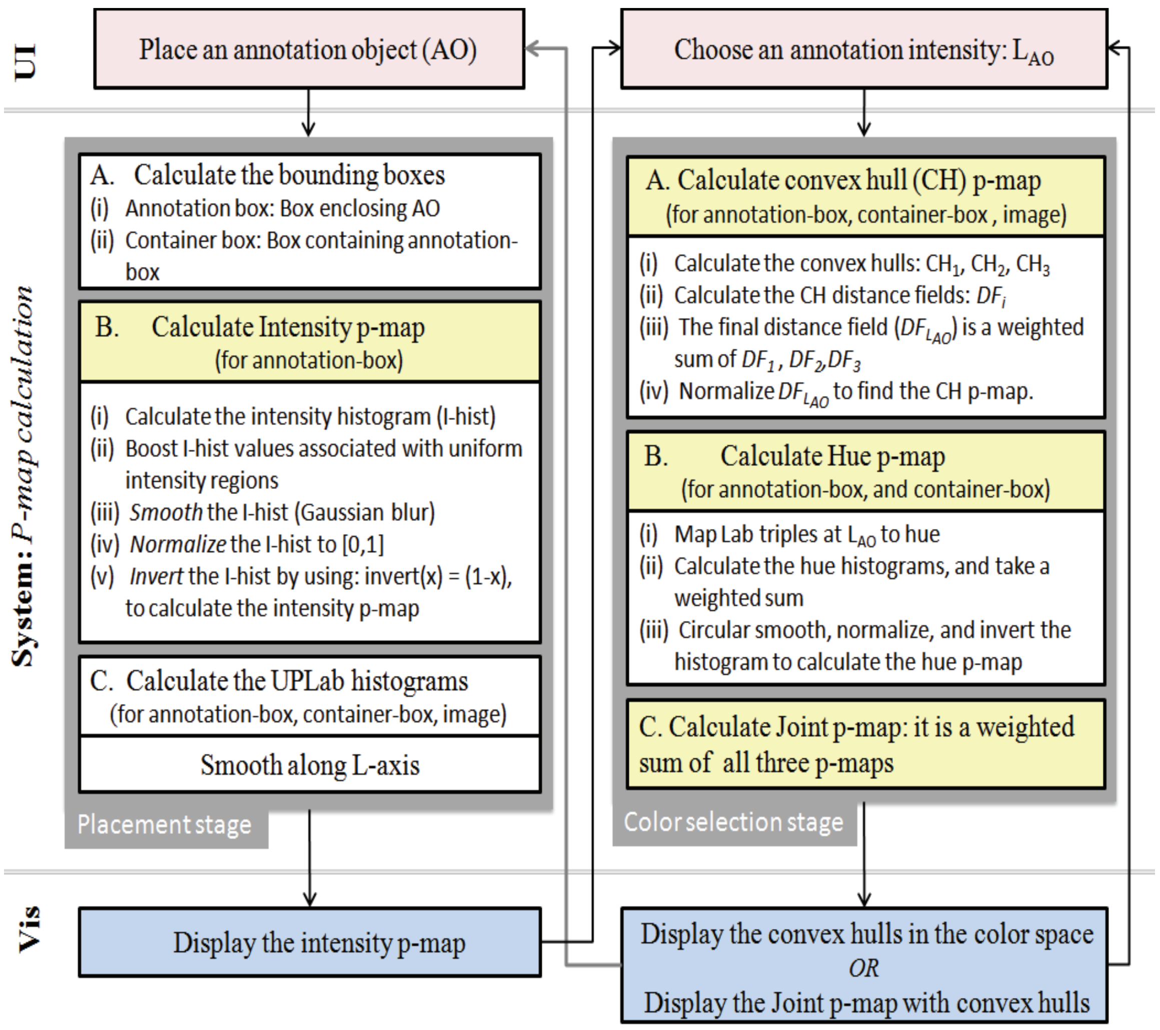

UI
Place an annotation object (AO)
Choose an annotation intensity: L_{AO}
System: P-map calculation
A. Calculate the bounding boxes
(i) Annotation box: Box enclosing AO
(ii) Container box: Box containing annotation-box
B. Calculate Intensity p-map (for annotation-box)
(i) Calculate the intensity histogram (I-hist)
(ii) Boost I-hist values associated with uniform intensity regions
(iii) *Smooth* the I-hist (Gaussian blur)
(iv) *Normalize* the I-hist to [0,1]
(v) *Invert* the I-hist by using: invert(x) = (1-x), to calculate the intensity p-map
C. Calculate the UPLab histograms (for annotation-box, container-box, image)
Smooth along L-axis
Placement stage
A. Calculate convex hull (CH) p-map (for annotation-box, container-box , image)
(i) Calculate the convex hulls: CH_1, CH_2, CH_3
(ii) Calculate the CH distance fields: DF_i
(iii) The final distance field ($DF_{L_{AO}}$) is a weighted sum of DF_1 , DF_2,DF_3
(iv) Normalize $DF_{L_{AO}}$ to find the CH p-map.
B. Calculate Hue p-map (for annotation-box, and container-box)
(i) Map Lab triples at L_{AO} to hue
(ii) Calculate the hue histograms, and take a weighted sum
(iii) Circular smooth, normalize, and invert the histogram to calculate the hue p-map
C. Calculate Joint p-map: it is a weighted sum of all three p-maps
Color selection stage
Vis
Display the intensity p-map
Display the convex hulls in the color space
OR
Display the Joint p-map with convex hulls

Figure 5.1: The flowchart shows how the Magic-marker system calculates, and updates the preference map (p-map), once the user loads an image into it

The annotation problem includes problems like layering, highlighting and blending with background text. In our current work, we focus on picking the right color for text. Text has the interesting property that it has a high level of detail, but is familiar to people at the same time. We focus on legibility, i.e. each letter should be clearly visible, and recognizable on its own. This is different from testing the rate of speed-reading, since in that process, users tend to use the mental model of the language to fill up the hazy or illegible letters.

Multiple studies have been done on readability or legibility of text given a foreground-background color combination. The results show that the subjective opinion is often based on the aesthetic and stereotypic presumptions and may thus differ from the objectively measured performances [33]. This tells us that it is valuable to have an interactive tool which guides the user towards a good sub-space; the user has to make the final choice based on personal preferences and the target audience.

A large amount of the work on legibility has been done by web page designers. Their task is simpler since they deal with a single colored background with uniform texture. However, their studies uncover a wealth of information regarding the importance of luminance contrast and chromatic contrast.

A good review of the past studies on readability and legibility on posters and CRT displays is presented in the paper by Humar et al [39]. Most results suggest that a luminance contrast accounts for most of the variance in typical legibility experiments. Travis et al [93] did experiments on detection and discrimination of words and non-words on multi-colored displays. The results showed that when the luminance contrast between text and background color was 0, a near perfect reading was still possible. This important finding means that purely chromatic differences may be sufficient for the visual system to maintain word identification. They explained the previous results by saying that typical displays produce much larger multiples of threshold luminance-contrast than threshold chromatic-contrast.

Experiments on reading speed on a color monitor [48] found that when both color and luminance contrast are present, there is no sign of additive interaction, and performance is determined by the form of contrast yielding the highest reading rate. Studies done on legibility on multicolor CRT displays [83] confirm this to some extent by saying that chromaticity contrast and luminance contrast are additive only under specific conditions. However, their results give more importance to luminance contrast by saying that chromaticity contrast can neither improve legibility if an acceptable level of luminance contrast is already present, nor substitute for luminance contrast.

Even today color difference (ΔE) equations assume that luminance and chromatic differences are additive — they usually use a weighted Euclidean distance approach. This clearly shows the lack of conclusion on the combined influence of luminance contrast

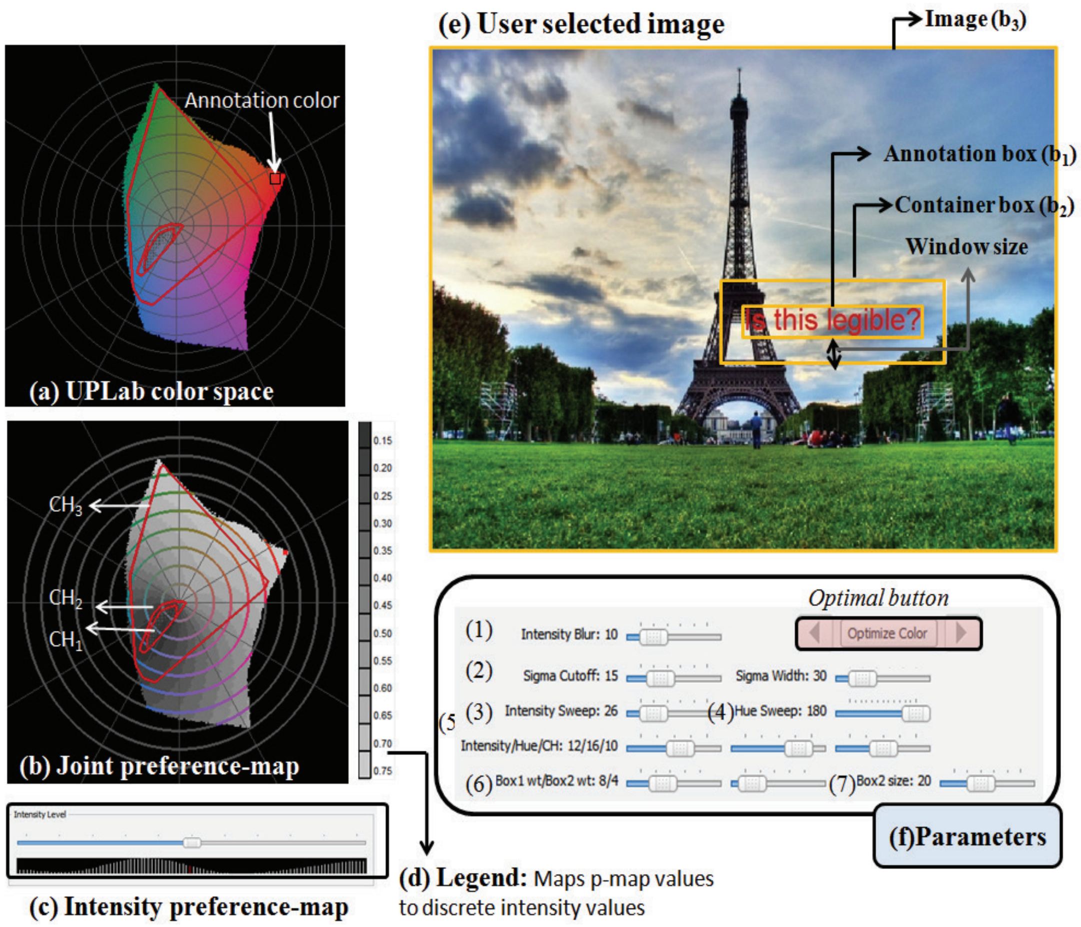


Figure 5.2: Illustrates the different components of the user-interface

and chromatic contrast on legibility. Our tool is therefore meant to be used interactively, to quickly get a choice of optimal label colors, from which users can choose one according to their preferences. We encode rules from color perception, and legibility studies to decide upon the optimal colors.

The rest of the chapter is divided as follows. Section 5.2 looks at some related work. The contributions of this work are discussed in section 5.3. In section 5.4, we discuss the theory behind our distance calculation and interface design. In section 5.5, we discuss the working details of the magic marker. In section 5.6 we look at some results. This is followed by the evaluation in section 5.7, where we do conjoint analysis [30] to learn the appropriate parameter values. Finally we conclude in section 5.8, and discuss ways to apply this tool in different applications.

5.2 Background work

In this section we look at some of the related work. First we look at how color can be represented to closely imitate the way humans perceive them. Next, we look at some methods that study the interaction of color. These papers concentrate on two aspects – generating color maps/palettes for mapping data to color, and designing interfaces where users can directly manipulate the colors in their visualization.

5.2.1. Color Spaces

The Munsell color chart [63] is used to evaluate the perceptual qualities of color spaces. Munsell describes color in terms of hue, value and chroma; hue corresponds to dominant wavelength, value to brightness and chroma to colorfulness. Unlike saturation, which is a statement of colorfulness as a fraction of the maximum possible for a given hue and value, chroma is an absolute measure of colorfulness. The maximum possible chroma differs among hues – for example, the maximum chroma for red is much greater than for green.

5.2.2. Color Design

The landmark texts by Itten [41] and Wong [98] provide great insight on the human perception of color and its aesthetic aspects. Much information is also available in books by Stone [85] and Ware [96]. Color mapping is the well studied topic of mapping data points to color based on human perception, cognition, and color theory. *PRAVDAColor* [9] helps users select color maps for mapping data points to color in scientific visualization. The Color Brewer [11] contains expert designed color palettes for mapping cartographic scalar data. These tools are either take a lot of tweaking to come up with a suitable palette, or are pre-designed by experts. The recent paper by Wijffelaars et al. [97] takes the expert out of the loop, and generalizes this process. Relevant also is the interactive color palette tool proposed by Meier et al [59], designed to support creative interactions for graphics designers.

While designing color, we often want to ensure that the final image looks natural, and aesthetically pleasing to the users. In [74] it was demonstrated that color maps should preserve a monotonic mapping in luminance since they are perceived as more *natural* by the human observers.

One popular design aspect is color harmony, which defines sets of colors that are aesthetically pleasing to human perception. In search of an intuitive 2D representation for visual designers, Itten then arranged the harmonic colors into a color wheel, which flattened this color space to mostly variations in hue. This system was used by Cohen et al [15] to quantify the color harmony of an image and shift the hues towards a harmonic setting. Wang et al [95] also use color harmony based rules while creating their framework to help users select colors in scene objects. The users can specify their hues of choice, while the system assists by making suggestions based on aesthetics, and by optimizing the luminance and contrast channels. In [64] we see a framework that allows users to manipulate colors directly in a 3D perceptual color space, instead of using multiple iterations using 2D color manipulation tools. This system assists users in object highlighting and annotation in color-rich scenes.

5.3 Contributions

In this section, we discuss some of the shortcomings of the current state of the art, and how we address these issues in our work.

5.3.1. Global vs. local effects

The work by Bauer [6] tells us that as long as we pick a color outside the convex hull formed by the colors (in CIE-LAB space) of a set of displayed data points, we can easily spot this target color. However, when we consider the case of a photographic image, an annotation occupies only a small portion of it. Therefore, the annotation color is highly influenced by the colors in the background surrounding it. For example if the person standing next to you wears a combination of red and green, it will strike as being mismatched. However, if there are two people standing next to each other – one in red, and the other in green, the color clash already reduces a bit, and as they move further away, this clash slowly disappears. Also, when we look at Figure 5.2(b), the outermost convex-hull (CH_3) occupies almost the entire color-space, leaving little choice for the users. However CH_1, the convex-hull formed by the colors surrounding the text is much

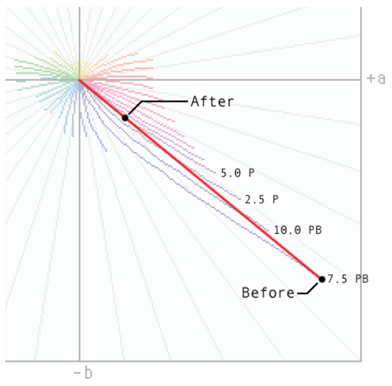


Figure 5.3: *Blue turns to purple*

smaller, and is the set which truly represents the colors we must avoid. We take this local vs. global effect into consideration by giving different weights to different regions of the image depending on their distance from the annotation.

5.3.2. Conflict in color mixing

The studies on legibility and readability of web-pages have shown us that there is no clear consensus on the interaction of luminance and chromatic contrast. The way to design a good system to assist with annotation therefore requires an intuitive user interface, where the user is provided with informed choices, and he can make the final decision. The use of a visual interface to guide users in their task by providing good alternatives is a hallmark of visual analytics.

5.3.3. Color space

The human vision is designed such that it is natural to describe colors as locations in a 3D space. The tristimulus values of a color are the amounts of three primary colors needed to match that test color. Tristimulus values are often given in the CIE 1931 color space, in which they are denoted X, Y and Z. While the CIE tristimulus values provide a complete color description, they do not correspond to the way humans perceive colors. In particular, distances between tristimulus triples do not reflect the degree to which we perceive the colors to differ. Therefore, the CIE introduced two perceptual color spaces in 1976 – the CIE-Luv and CIE-Lab spaces. The CIE-Luv space was used in [97] to design their color palettes. However, though these spaces are designed so that the distance in the color space directly corresponds to the difference as perceived by humans, they do not

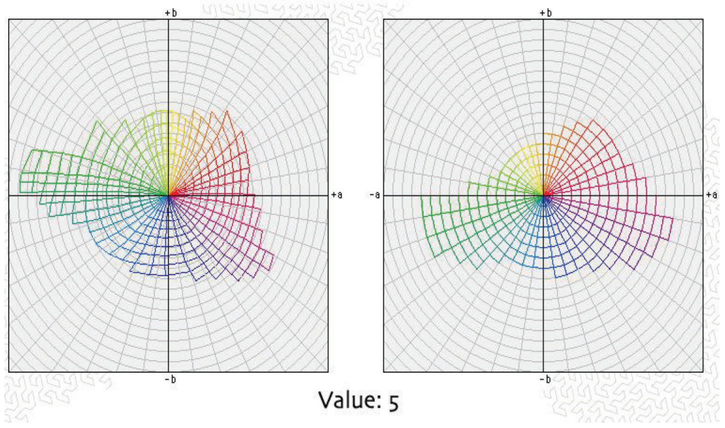


Figure 5.4: The transform of the CIE-Lab space to UP-Lab space for the Munsell value 5.

contain the colors arranged uniformly along the hue, and chromatic channels. The Munsell data set, as discussed in Section 5.2.1 is an example of a color space which separates luminance, hue and chroma into 3 orthogonal axes. Therefore, for our calculations, and user interface, we use a modified CIE-Lab space, such that it satisfies the properties of the Munsell color set.

5.4 Overview

The main task for our system is to find rules which assign legibility scores, i.e. p-map values to each point in our 3D color space. In this section, we look at how these rules are derived.

5.4.1. Minimum contrast for legibility

Web-page designers have long studied the ways to choose a good foreground-background combination for maximum legibility. Though placing text on an image is far tougher since the background color is rarely uniform, we can still learn from their work. Since results from their studies show that a certain luminance contrast assures legibility, we must make sure that we achieve the minimum contrast from *most* colors in the background.

According to the studies by Maureen Stone [86], a contrast (ΔL) of 20 is legible for text; contrast of 30 is easily readable; and contrast of 60 is robustly readable. This information comes in handy when we assign a score to each intensity level based on luminance contrast.

5.4.2. Perceptually uniform CIE-LAB space

We require a perceptually uniform color space for our system since this would lend to an intuitive system for a user. In the CIE-Lab, colors at similar distance, have similar perceptual differences. However the space itself is not a perceptually uniform space. This is because colors sharing the same Lab hue angle do always not share the same apparent hue. For example if you move inwards along a line joining a bright blue to the origin, the apparent hue changes to purple: this is the well known “blue turns purple” problem in gamut mapping (see Figure 5.3). Furthermore, the angles between hues are not equal, and colors of constant chroma are not equidistant from the neutral axis. To rectify this, we use the non-linear mapping ICC profile provided by Bruce Lindbloom [14]. This mapping is similar to the gamut mapping working done by Marcu [58].

The color space ICC profile performs a nonlinear mapping of Lab so that all properties of the Munsell color set are met. This means that when *CIE-Lab* colors are transformed through this Lab-to-Lab profile, the resulting Uniform Perceptual Lab (or "*UP Lab*") colors have these properties:

1. All colors of constant Munsell hue have the same *UP Lab* hue angle.
2. All Munsell hues are evenly distributed around the hue wheel.
3. All colors of constant Munsell chroma (saturation) lie the same distance from the neutral. Another way of saying this is that chroma rings are perfect circles, centered on neutral.
4. All chroma rings are equally spaced.

During this transformation, the luminance channel and the neutral colors remain unaffected. An illustration of the transformation is shown in Figure 5.4.

5.4.3. Convex hulls

Bauer et al in [6] have shown that given a bunch of colored points laid out as a scatterplot, it is easier to find a target color with a color outside the convex hull (in the CIE-Lab space) of the source colors than inside the convex hull. This holds both when the target was linearly separable in chromaticity only, or in a combination of luminance and chromaticity.

This result gives us a metric which calculates the score for each point in the CIE-Lab space based on their distance from the convex hull.

5.4.4. Theory

Based on the background work done in color perception, and legibility of text on multi-color screens, we form some ground rules which help us in designing the *Magic Marker* user interface. We use three orthogonal *preference-maps* (p-maps) to find the colors which lead to good legibility. Finally, we combine these values by taking a weighted sum, to find the final preference-map value – the *joint p-map.*

Luminance Contrast: We need to make sure that there is a minimum amount of luminance contrast between the text and background. Since the background is not uniform, we calculate the contrast based on the intensity histogram of the background. Our thumb-rule: $\Delta L > 30$ is sufficient for giving an ideal annotation color.

Hue based contrast: Luminance contrast alone is not sufficient to make a good labeling color. We need to make sure that the hue is also sufficiently distinct. For this purpose, we use the *UP-Lab* space to calculate our hue-value. The hue is determined by the angle formed with the origin, i.e. the hue for a *UP-Lab* color with coordinates (L', a', b') is:

$$hue = tan^{-1}(\frac{b'}{a'})$$

This is similar to the LCH_{Lab} space as defined by Bruce Lindbloom. The further away a hue is from the hues in the background, the better it performs.

Convex hulls: As we learn from Bauer et al [6], it is easier to find a target color when it lies *outside* the convex hull formed by the distractor colors. This makes sure that we avoid the hue and chroma combinations present in the background. Also, since this space covers chroma, we do not create a separate preference-map for it. To make sure that we give higher importance to local features, we divide the image into three nested boxes (see Figure 5.2(e)) – the annotation box (b_1) which is the bounding box for the annotation, the container box (b_2) that extends a small amount beyond b_1, and finally the whole image (b_3).

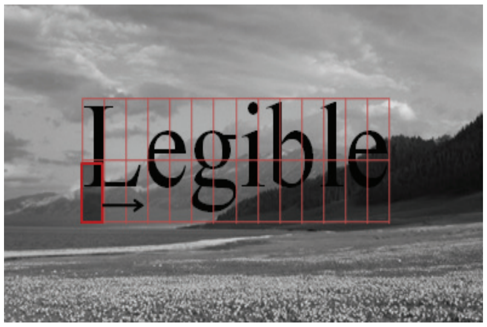


Figure 5.5: Sliding window. We show how uniform intensity is determined by passing a window less than the size of a letter

5.5 Implementation details

A user can upload a picture of choice into the interface, and the system gives a score to each point in the 3-D color space, based on its *legibility potential* at a given location in the image. We work in the *UP Lab* color space for both the score calculation, and the visual interface. To calculate the scores, we create a *preference-map* (p-map), which takes into accounts three values: (a) the convex hull p-map: p_{CH}, (b) intensity based p-map: p_I, and (c) hue based p-map: p_H. For all these scores, we work on the histogram of the image in different dimensions. Further, to avoid sudden changes in the p-map values of neighboring points, and to take neighborhood values into account, we apply the Gaussian filter at different steps. An overview of the system is shown in the flowchart in **Figure 5.1**.

We first discuss how a color's hue, intensity, and location with respect to the convex hulls define its p-map value. Finally, we look at how these values are displayed in the visual interface in Section 5.5.4.

5.5.1. Intensity based p-map

The intensity based p-map assigns each intensity level in the range [0,100], a value based on how well colors in that level will work from the point of view of *luminance contrast* with the background. As we mentioned in section 5.4.1, a contrast of 30 will make text easily legible. We define contrast as the absolute difference between the foreground and background intensities.

Intensities that are farther away from those present in the background are given a higher value. A simple way is to calculate the intensity histogram, normalize it to the range $[0,1]$, and invert it. *Invert* means applying the following function:

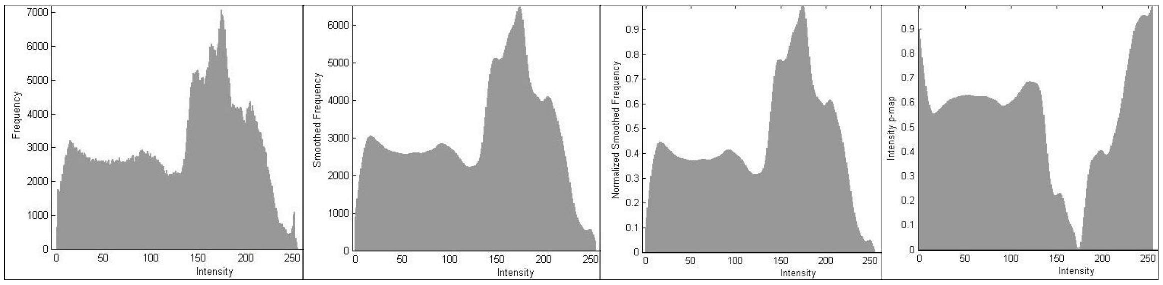
$$invert(x) = 1 - x$$

We also apply a Gaussian filter to the histogram before the normalization step so as to give lower p-map values to intensities in close proximity to those in the background. The parameter *intensity sweep* defines the size of the filter. Hence, p_I is calculated for each intensity level using the following steps: calculate intensity histogram → apply Gaussian filter → normalize → invert. An example of this process is shown in the Figure 5.6.

However, we found that this algorithm didn't work well in cases where there is a small patch in the background with an *almost* uniform intensity (see Figure 5.5). This is because they do not have a sufficiently large presence in the histogram to be penalized heavily; however since they form a continuous block, using a similar intensity for annotation works poorly. We address this by adding a module which passes a sliding block over the text area to find regions with small variations in intensity. The maximum allowed standard deviation to qualify a region as uniform is specified by the parameter *sigma cutoff.* If any such region is found, then the mean intensity, and intensities within a certain width (*sigma width*) are penalized in the intensity p-map. The p-map values for such a region start from zero at the mean, and increase linearly towards the ends.

5.5.2. Hue based p-map

In the previous section we assign each intensity level a p-map value. Within an intensity level, we would like to give low hue p-map values to hues which are present in the background, especially those with *similar* intensity levels. We first find the hue values for each point in our current intensity space. Next we calculate the hue histograms for the annotation box and the container box (see Figure 5.2(e)), and take a weighted sum of the two. This is followed by smoothing using a circular Gaussian filter since hues wrap around in the *UPLab* space. Then we normalize the data to the range $[0,1]$ and apply the invert function to get the hue p-map p_H. The size of the filter is again defined by a parameter: *hue sweep*.



(a) Original Histogram **(b)** Smoothed histogram **(c)** Normalized smoothed histogram **(d)** Final intensity p-map formed after *inversion*.

Figure 5.6: Histograms showing the calculation of intensity p-map

5.5.3. Convex hull based p-map

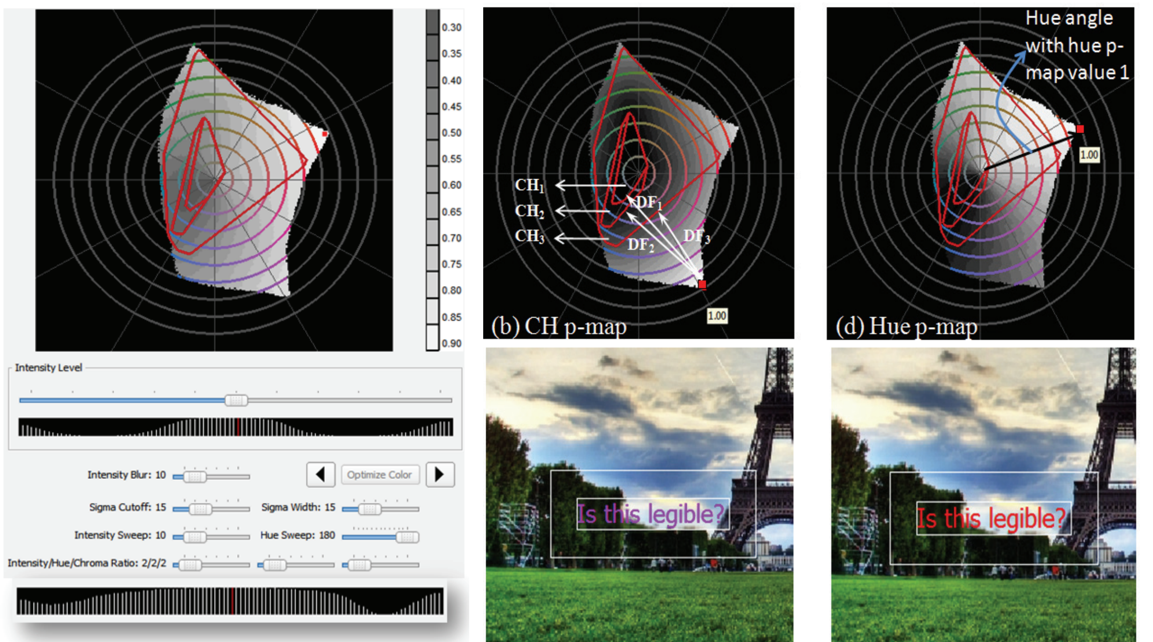
The global convex hull method mentioned in section 5.3.1 appears overly restrictive for annotations which are local in nature, i.e., the color content of a distant image region may not be a distractor here. Further, they work in 2-D space, not taking the intensity into account. We therefore extend this idea by:

(a) Calculating convex hulls separately for each intensity level. In order to prevent sudden changes in the shape of the convex hulls, and to take the occurrence of colors in neighboring intensities into account, we apply a 1-D Gaussian filter (with size *intensity blur*) to the histogram at each chroma or (a, b) value in the *UP-Lab* space.

(b) We create separate convex hulls for three nested boxes in the image. Since the local effect is expected to be greater than the effect of the image as a whole, we give different weights to the boxes. The parameter *window-size* tells us the distance by which b_2 extends beyond b_1 (see Figure 5.2(e)). Also, we give these 3 boxes weights w_1, w_2 and w_3, with $w_1 > w_2 > w_3$. We then calculate the distance field DF(a, b) of all possible (a, b) pairs from the convex hulls as follows: $\sum_i(w_i \times DF_i(a, b))$, where $DF_i(a, b)$ is the distance of point (a, b) from the ith convex hull, CH_i (see Figure 5.2(b)). These distance field values are normalized to the range $[0,1]$ to give us the convex hull p-map values p_{CH}.

The *joint p-map* is calculated as a weighted sum of the scores calculated in the previous three subsections, i.e.

$$\frac{(w_I \times p_I + w_H \times p_H + w_{CH} \times p_{CH})}{(w_I + w_H + w_{CH})}$$



Intensity Blur: 10

Sigma Cutoff: 15

Sigma Width: 15

Intensity Sweep: 10

Hue Sweep: 180

Intensity/Hue/Chroma Ratio: 2/2/2

(a) Joint p-map with all 3 p-maps weighed equally. The intensity p-map peaks in the center. Below, we see the intensity map when sigma-cutoff is change to 0

(c) Annotation color is the color with highest value in CH p-map

(e) Annotation color is the most saturated version of the hue with the highest Hue p-map value

Figure 5.7: (a) The user interface, including the joint p-map at intensity level 50. Next we look how (b) CH p-maps, and (d) Hue p-maps look separately. (c), (e) show the image annotated with the best p-map value restricted to CH, and hue respectively.

5.5.4. Visual Interface

The visual interface as shown the user explore the UP Lab space and find colors which have good p-map values for a given image and location. The intensity map is shown separately at the bottom, as a histogram, to guide the user towards intensities providing good luminance contrast. Next, we see a legend at the right, which shows the range of the values present in the joint p-map at the current intensity level. Finally, we see the color space, along with the convex hulls (in red). We also overlay sectors and circles on the color space (Figure 5.2(b) and Figure 5.7(a)(b)(d)), so that a user can easily select colors with the same hue or saturation.

The user can toggle the display to see the joint p-map instead. In this view, we discretize the p- map values into 20 levels, and assign each point an intensity value based on these levels. This helps the user follow the patterns in the p-map much better. The user can hover the mouse over a point to see the exact p-map value. We retain the circles and sectors from the color space display, and fill them with the colors they represent to guide the user as he makes selections on the basis of the p-map. When an expert user is using the system, we can expose the parameters of the system, so that he can tweak them to get better results.

We provide an *optimal button*, which finds all the colors with the highest p-map values. This requires us to find the joint p-map values at all intensity levels. Next we find the maximum p-map value in the entire color-space ($\mathrm{max}_{\mathrm{p-map}}$), and return all the colors with the property:

$$p\text{–}map(l, a, b) > max_{p-map} \times threshold_{opt}$$

The threshold value $threshold_{opt}$ is a real number between 0 and 1. We usually use a value of 0.95 in our interactive system. Once these optimal colors have been found, the user can easily browse through them using the forward/backward buttons shown in Figure 5.7(f). A minor tweak in one of the optimal colors often gives acceptable results.

5.6 Results

The *Magic-marker* tool is very simple to use for even novice users, and responds interactively for most operations. Let us look at a running example to see how the system works.

5.6.1. Case Study

The tool starts up with some initialization steps. The mapping of the CIE-Lab space to *UPLab* space using the look-up table takes a relatively large amount of time (approximately 1-2 minutes). After the system starts running, a user can load any image of his choice. At this point, a default setting is already initialized, along with the default intensity level setting of 50. Once the image has been read in, we can see the intensity p-map values as a histogram, as well as the joint p-map values of the currently selected intensity in a 2-D representation as shown in Figure 5.7 (a). The intensity p-map guides the user towards the intensity levels with higher intensity p-map values. In the present case, we can clearly see that the intensity p-map is highest in the middle region – around the intensity level 50.

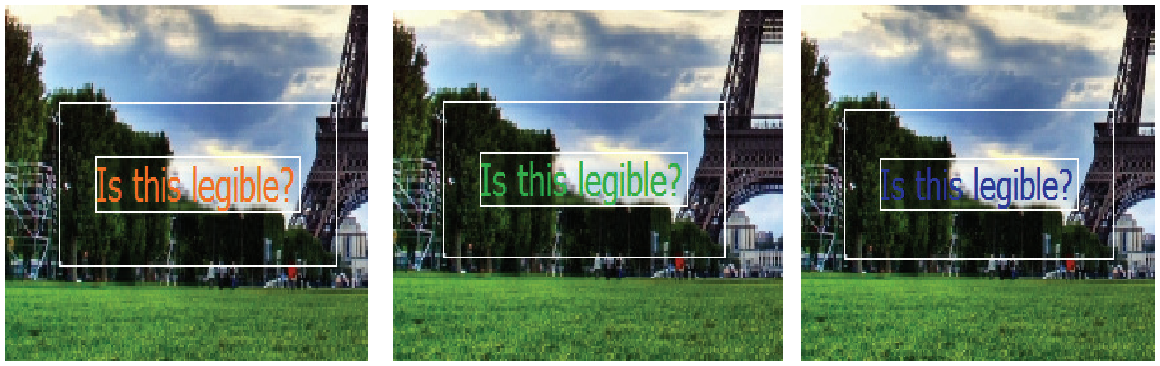


Figure 5.8: This shows some sub-optimal results obtained when the sigma-cutoff parameter is changed to 0. The first two images come from the higher ends of the intensity range, and the third image is from a lower intensity.

Next, we look at the hue p-map and convex-hull p-map separately to understand how the p-map calculation works in those spaces. Convex hull p-maps, as shown in Figure 5.7(b) have the lowest p-map values inside CH_1 (where it is always zero), and they progressively increase as we move outwards from CH_1. Finally, a p-map value of 1 is reached at at-least one of the corner locations. In this example, the maximum is reached at purple, on the lower left corner.

The histogram p-map calculation takes a *hue sweep* value of 180. This means that if there is exactly one color present in the background, the hue p-map values will increase starting from 0, as we move away from the background hue, reaching a maximum value of 1 at the diametrically opposite hue value. Since the hue p-map is already normalized, we will definitely see a hue with a p-map value 1. In Figure 5.7(d), we can see the hue value with p-map 1 highlighted – let us call it huepref-max. The text in Figure 5.7(e) is annotated with a color which is a combination of the current intensity level (50), hue value huepref-max, and the maximum possible chroma for this hue and intensity combination.

This analysis of the individual p-maps shows us that due to the way the CH p-map is calculated, the best colors will always lie on the edge of the color space, i.e. those colors that have the maximum possible chroma for a hue and intensity combination. Intensity maps and hue maps on the other hand restrict the best colors to a certain intensity level or hue value – thus any intensity or hue can be given the highest p-map value. When we combine these p-maps to form the joint p-map, the restriction from CH p-maps will remain.

The best colors as suggested by the hue p-map and CH p-map are clearly distinct – red and purple respectively. However, they are both perceived to perform equally well. We transfer our attention back to the joint p-map, with all three p-maps being weighed equally. Next we press the optimize button, which indeed returns variations of the red and purple annotation colors seen in Figure 5.7.

Our default setting had sigma cutoff set at 15. If we change it to zero, i.e. we no longer care for areas with uniform intensity, we end up with the intensity p-map shown at the bottom of Figure 5.7(a). This intensity p-map gives a good preference value to a much larger number of intensities. This makes it difficult for a user to explore the space, and find an appropriate color. Even the optimal button returns a lot of colors which do not lend themselves to a legible annotation. Some of them are shown in Figure 5.8. The first two have intensities in the higher intensity range (about 60-65), whereas the blue comes from intensity level 23. All these intensities have a low to mid-range score in the first intensity p-map, and are never considered as optimal color ranges. This shows us that identifying regions of uniform intensity in the background is an essential process.

5.7 Evaluation/User studies

We carry out user studies to learn the ideal parameters for our system. Results are automatically generated by sampling a discretized parameter space, and using the optimal button. We use a $\text{threshold}_{\text{opt}}$ value of 1, i.e. only the colors with the highest joint p-map value will be returned.

5.7.1. Visualization Parameters

We carried out legibility based user studies to find out the best parameters for our system. We have 7 independent parameters in our system as shown in Figure 5.2(f).

1. Intensity blur: The filter size for blurring for each chroma or (a, b) value *across* multiple intensity levels - this gives rise to the histogram for the Lab space. It can take the following values: (0, 15, 30).

2. Sigma cut-off: The standard deviation up to which a block is considered as having *almost* uniform intensity (Section 5.4.1). The three levels for this parameter are: (0, 10, 20).

   Sigma width: The width till which the above uniform block has a dampening effect on the intensity score. Sigma width's three levels are: (0, 10, 20). It is dependent on sigma cutoff, and cannot have a non-zero value when sigma cutoff is zero. These two parameters can take a total of ($3^2 – 2$) or 7 unique values.
3. Intensity sweep: Gaussian filter size for blurring the intensity histogram. It takes the values: (0, 15, 30).
4. Hue sweep: Gaussian filter size for blurring the hue histogram. This filter size has the following levels: (0, 90, 180).
5. Intensity/Hue/Chroma ratio: This gives us the weights to combine the separate values. We let the ratio take all values from the space $[1, 2, 3]^3$ except for the redundant combinations: 2:2:2 and 3:3:3 which are the same as 1:1:1. This gives us ($3^3 – 2$), i.e. 25 combinations.
6. The width of the box b_2. This parameter has three levels: (10, 20, 40)
7. The weights w_1 and w_2 for the boxes b_1 and b_2. The value of w_3 is fixed to 1, and $w_1 > w_2$. The two weights can take a value from (2, 4, 8). The 3 valid weight combinations are: (w_1, w_2) = (4, 2), (8, 2), (8, 4).

These discretized values give us a finite number of combinations, so that we can do conjoint analysis to learn the best suited parameters. The levels guide us in calculating the minimum number of questions which must be answered to estimate the parameters accurately. The size of our parameter space is 3^5 x 7 x 25 ~ 40000 and the total number of levels is 3*5 + 7 + 25 = 47.

5.7.2. Discussion

In our study, the users are presented with a pair of images which show the same image annotated in the same location based on different parameters. The user has to select the image in which the text is more legible. On the basis of these results, we carry out conjoint analysis as presented in [30] to jointly study the performance of the parameters. The conjoint analysis study was conducted online, and we had results from approximately 1000 image comparisons. However, most of the results are not statistically significant. The one trend which does stand out is that for window ratios, $(w_1, w_2) = (4, 2)$ performs the worst. The other two values are (8, 4) and (8, 2). Here, we must remember that w_3 was fixed to 1. Therefore, the results show that the outermost box has very little influence on the legibility.

The size of the parameter space is very large, and we need more user studies to sample the parameter space well and get good results. Further, for the user study, we use the output created by the *optimal button*. The size of its result set depends both on the parameter settings, and the threshold value. If there are too few colors in the result set, then we might have missed some preferred colors due to the threshold. On the other hand, if the parameter settings allow a large number of colors to pass the threshold, then randomly picking a single color will not represent the optimality of the setting.

We observed that colors at the extremes of the luminance scale often get selected as *optimal* colors, but do not make a good annotation color due to the absence of *colorfulness*. At high intensities we get white/yellow, and at low intensities black/dark blue. These colors should be given lower p-map values in most cases. A better way of carrying out the user studies would be to generate images with annotation colors taking all values along the *UP-Lab* gamut surface, and conducting the pair wise comparison tests. Parameter settings can then be judged based on the percentage of *good* optimal colors, where *good* colors are those preferred by the users. Further, in order to minimize obvious cases we could make the selection of the two colors to be compared based on weights derived from a very simple distance model (e.g. distance from the average background behind the text). Then clearly illegible annotations would be chosen very rarely.

5.8 Conclusion

In this chapter we present an interactive framework for users to choose appropriate colors for annotating their images. This works better than a static program which can suggest the best colors, since the field of legibility is still under study, and has no fixed conclusions on the interaction between luminance contrast and chroma contrast. Our system is fairly general, and can be easily modified to incorporate any newly acquired knowledge about color perception and luminance/chromatic contrast.

Currently, the most commonly used methods to add annotation to images is to use text that has either an outline, or shadows added for extra contrast. Our method provides a better alternative since the text does not end up taking the center of attention in the image – it is only an annotation. However, one topic which could work as another parameter in our system is the thickness of text. Observations clearly demonstrate that a thicker font leads to better legibility, especially against an image.

A good way to extend this system is by incorporating a learning module into it so that at the end of every successful color assignment, it stores the various parameters which gave the result. As more sessions are carried out, the learned information can help the system modify the weights it assigns to different parameters, hence hopefully providing a more precise scoring scheme. Given such an *intelligent* system, at one point, the only exploration the user will need to do is to find a color scheme which matches his personal preferences.

Chapter 6. Conclusion

To summarize, this dissertation presents the following results:

1. A visual analytics system which allows users to explore high dimensional datasets, and learn inductive models by visually providing positive and negative examples.
2. A visual analytics system which lets users initialize models by controlling the clustering parameters, and feature vector weights. Then the user can further refine the model by labeling the most uncertain data points.
3. A framework that uses a dual domain approach involving natural language text processing and global image databases to help users identify metaphors suitable to visually encode abstract semantic concepts.
4. A system that helps users by suggesting appropriate colors for annotating an image. This works like a "Magic Marker" in the hands of a user, giving them the power to automatically choose a good annotation color, which can be varied based on their personal preferences.

In this report, we see four tools/frameworks which contribute towards the overall task of visual analytics. They improve both the visual aspects of it by providing more expressive representations of information, and the analytic aspects of it by bringing the machine into the analysis or *sense-making* loop.

An initial step towards adding this visual analytic framework to medical emergency room analysis is presented in [103]. Further, a framework to capture and retrieve the visualization knowledge of different people using a web-based interface is presented in [29].

On the communication side, the task of making computers better at communicating is discussed in an upcoming journal article [28].

6.1 Future Work

The work presented in this dissertation is only a starting step towards combining the features of visual analytics to help in the process of machine learning. Even in cases where learning a model is not the ultimate aim, figuring out an appropriate format to store the outcome of an analytic session is useful for any VA application. Having this as a seamless outcome of each analytic session could help take a huge step forward in Visual Analytics. Given such a feature in any VA system would enable it to apply lessons learnt from a given dataset to another dataset involving similar tasks. This especially makes sense since VA systems tend to solve very specific problems – i.e. they are designed to work for a specific data type and problem combination.

Further, extending our ideas presented on *designing* feature-vectors (as seen in Chapter 3) could also help analyze data with no obvious feature vectors. More work in this direction can help design good design parameters so that a user does not have to hand-specify the feature vector components. Having a hierarchical list of parameters and their mathematical specifications can allow users to pick and modify the ones most suited to the dataset and task at hand. For example if *context* is a parameter, the user can select it, and then specify what range constitutes this *context.*

The iconizer project can be used to make both the user interfaces more expressive and the final reports/stories more compact and engaging. Further work in this area requires user studies and inputs to convert this framework into a full-fledged system capable of suggesting icons for any query term. Also, the semantic zooming concept presented in this work has major applications in VA. A generalization of this approach can be used to automatically present the entire dataset in accordance with the security clearance of the people viewing it. For example if a person is cleared to see the profession of a suspect, but is barred from seeing his actual name, then the face of the person can be automatically replaced by an icon representing people in that profession.

References

[1] Aleph. *http://web2.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph.html*.

[2] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi and F. Giannotti. Interactive visual clustering of large collections of trajectories. *Proceedings of IEEE VAST.* , 2009.

[3] M. Ankerst, M. Ester and H. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*(Boston, Massachusetts, United States), , pages 179-188, 2000.

[4] D. Asimov. The grand tour: a tool for viewing multidimensional data. *SIAM Journal on Scientific and Statistical Computing*. 6: 128, 1985.

[5] A. Asuncion and D.J. Newman. {UCI} Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. 2007.

[6] B. Bauer, P. Jolicoeur and W.B. Cowan. Distractor heterogeneity versus linear separability in colour visual search. *Perception(London. Print)*. 25(11): 1281–1293, 1996.

[7] D. Beeferman. Lexical discovery with an enriched semantic network. In *Proceedings of the ACL/COLING Workshop on Applications of WordNet in Natural Language Processing Systems*, pages 358–364, 1998.

[8] S. Belongie, J. Malik and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 24(4): 509-522, 2002.

[9] L.D. Bergman, B.E. Rogowitz and L.A. Treinish. A rule-based tool for assisting colormap selection. In *Proceedings of the 6th conference on Visualization'95*, pages 118-125, 1995.

[10] P. Berkhin. A survey of clustering data mining techniques. *Grouping Multidimensional Data*. : 25–71, 2006.

[11] C.A. Brewer. Color use guidelines for data representation. In *Proceedings of the Section on Statistical Graphics, American Statistical Association*, pages 55–60, 1999.

[12] S.K. Card, J.D. Mackinlay and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann. 1999.

[13] M.C. Chuah and S.G. Eick. Glyphs for software visualization. In *5th International Workshop on Program Comprehension (IWPC'97) Proceedings*, pages 183–191, 1997.

[14] CIE Lab to Uniform Perceptual Lab profile is copyright © 2003 Bruce Justin Lindbloom. All rights reserved. *http://www. brucelindbloom. com/index.html*.

[15] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand and Y.Q. Xu. Color harmonization.

Proceedings of ACM SIGGRAPH 2006. 25(3): 624-630, 2006.

[16] B. Coyne and R. Sproat. Wordseye: An automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496, 2001.

[17] P.J. Crossno, D.M. Dunlavy and T.M. Shead. LSAView: A Tool for Visual Exploration of Latent Semantic Modeling. *Proceedings of IEEE VAST.* , 2009.

[18] R. Datta, D. Joshi, J. Li and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. , 2008.

[19] R. Datta, D. Joshi, J. Li and J.Z. Wang. Studying aesthetics in photographic images using a computational approach. *Lecture Notes in Computer Science*. 3953: 288, 2006.

[20] I.S. Dhillon, D.S. Modha and W.S. Spangler. Class visualization of high-dimensional data with applications. *Computational Statistics and Data Analysis*. 41(1): 59–90, 2002.

[21] S.V. Dongen. *Graph clustering by flow simulation*. PhD Thesis, University of Utrecht, The Netherlands. 2000.

[22] R. Eccles, T. Kapler, R. Harper and W. Wright. Stories in geotime. *Information Visualization*. 7(1): 3–17, 2008.

[23] C. Fellbaum and others. *WordNet: An electronic lexical database*. MIT press Cambridge, MA. 1998.

[24] B.J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*. 315(5814): 972-977, 2007.

[25] J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.* 23: 881-890, 1974.

[26] S. Garg, J.E. Nam, I.V. Ramakrishnan and K. Mueller. Model-Driven Visual Analytics. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 19-26, 2008.

[27] S. Garg, T. Berg and K. Mueller. Iconizer: A Framework to Identify and Create Effective Representations for Visual Information Encoding. *IEEE Visualization, 2009.* , 2009.

[28] S. Garg, J. Nam, T. Berg and K. McDonnell. Can Computers Master the Art of Communication? An Excursion with a Focus on Visual Analytics. *(to appear) IEEE Computer Graphics & Applications.* , April 2011.

[29] S. Garg, J. Nam, K. Padalkar, K. Mueller, M. Chan, H. Qu, S. Laue, W. Saleem and J. Giesen. KAV-DB: Towards a Framework for the Capture and Retrieval of Visualization Knowledge over the Web. In *Proc. Schloss Dagstuhl Scientific Visualization Workshop*2010.

[30] J. Giesen, K. Mueller, E. Schuberth, L. Wang and P. Zolliker. Conjoint analysis to measure the perceived quality in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*. 13(6): 1664-1671, 2007.

[31] M. Götze, P. Neumann and T. Isenberg. User-Supported Interactive Illustration of Text. In *Simulation und Visualisierung*, pages 195–206, 2005.

[32] T. Götzelmann, M. Götze, K. Ali, K. Hartmann and T. Strothotte. Annotating images through adaptation: an integrated text authoring and illustration framework. *Journal of WSCG*. 15(1-3): 115–122, 2007.

[33] R. Hall and P. Hanna. The impact of web page text-background colour combinations on readability, retention, aesthetics and behavioural intention. *Behaviour and Information Technology*. 23(3): 183–195, 2004.

[34] J. He, H. Tong, M. Li, H.J. Zhang and C. Zhang. Mean version space: a new active learning method for content-based image retrieval. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 15–22, 2004.

[35] J. Heer, S.K. Card and J.A. Landay. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, 2005.

[36] A. Hinneburg, D.A. Keim and M. Wawryniuk. HD-Eye: Visual mining of high-dimensional data. *IEEE Computer Graphics and Applications*. 19(5): 22–31, 1999.

[37] D.D. Hoffman. *Visual intelligence: How we create what we see*. WW Norton & Company. 2000.

[38] R.E. Horn. To Think Bigger Thoughts: Why the Human Cognome Project Requires Visual Language Tools to Address Social Messes. *New York Academy Sciences Annals*. 1013: 212–220, 2004.

[39] I. Humar, M. Gradisar and T. Turk. The impact of color combinations on the legibility of a web page text presented on CRT displays. *International Journal of Industrial Ergonomics*. 38: 885-899, 2008.

[40] A. Inselberg, B. Dimsdale, I. Center and C.A. Los Angeles. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Visualization, 1990. Visualization'90., Proceedings of the First IEEE Conference on*, pages 361-378, 1990.

[41] J. Itten. *The art of color*. Van Nostrand Reinhold New York. 1969.

[42] F. Janoos, S. Singh, O. Irfanoglu, R. Machiraju and R. Parent. Activity Analysis Using Spatio-Temporal Trajectory Volumes in Surveillance Applications. In *IEEE Symposium on Visual Analytics Science and Technology, 2007. VAST 2007*, pages 3-10, 2007.

[43] J. Jiten and B. Merialdo. Semantic Image Segmentation with a Multidimensional Hidden Markov Model. *Advances in Multimedia Modeling*. 4351: 616–624, 2007.

[44] B. Kovalerchuk. Iconic reasoning architecture for analysis and decision making. *Visual and spatial analysis: advances in data mining reasoning, and problem solving*. : 129, 2004.

[45] J.B. Kruskal and M. Wish. *Multidimensional scaling*. Sage Publications, Inc. 1978.

[46] J.F. Lalonde, D. Hoiem, A.A. Efros, C. Rother, J. Winn and A. Criminisi. Photo clip art. In *ACM SIGGRAPH 2007 papers*, pages 3, 2007.

[47] J. LeBlanc, M.O. Ward and N. Wittels. Exploring n-dimensional databases. In *Proceedings of the 1st conference on Visualization'90*, pages 237, 1990.

[48] G.E. Legge, D.H. Parish, A. Luebker and L.H. Wurm. Psychophysics of reading. XI. Comparing color contrast and luminance contrast. *Journal of the Optical Society of America. A, Optics and Image Science*. 7(10): 2002-2010, October 1990.

[49] D.D. Lewis and W.A. Gale. A Sequential Algorithm for Training Text Classifiers. (Dublin, Ireland), , pages 3-12, 1994.

[50] M. Leyton. *Symmetry, causality, mind*. The MIT Press. 1992.

[51] J. Li, A. Najmi and R.M. Gray. Image classification by a two-dimensional hidden Markov model. *IEEE Transactions on Signal Processing*. 48(2): 517–533, 2000.

[52] X. Li, C. Wu, C. Zach, S. Lazebnik and J.M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. *ECCV (1)*. : 427–440, 2008.

[53] J.W. Lloyd. *Foundations of logic programming*. Springer-Verlag Berlin. 1984.

[54] H. Luo, J. Fan, J. Yang, W. Ribarsky and S. Satoh. Analyzing large-scale news video databases to support knowledge visualization and intuitive retrieval. In *IEEE Symposium on Visual Analytics Science and Technology*2007.

[55] J.U. Mahmud, Y. Borodin and I.V. Ramakrishnan. Csurf: a context-driven non-visual web-browser. In *Proceedings of the 16th international conference on World Wide Web*, pages 31-40, 2007.

[56] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. 35(3): 397–408, 2005.

[57] Y. Mao, J. Dillon and G. Lebanon. Sequential Document Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 13(6): 1208-1215, 2007.

[58] G. Marcu. Gamut mapping in Munsell constant hue sections. In *Proc. of IS&T/SID 6th Color Imaging Conference*, pages 159–162.

[59] B.J. Meier, A.M. Spalter and D.B. Karelitz. Interactive color palette tools. *IEEE Computer Graphics and Applications*. 24(3): 64–72, 2004.

[60] M. Meyer, H. Lee, A. Barr and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*. 7(1): 13-22, November 2002.

[61] C.W. Morris. *Signs, language and behavior*. Prentice-Hall Englewood, NJ. 1946.

[62] S. Muggleton and L.D. Raedt. Inductive logic programming: Theory and methods. *Journal of logic programming*. 19(20): 629–679, 1994.

[63] A.H. Munsell. *A Grammar of Colors*. Van Nostrand Reinhold Company. 1969.

[64] N. Neophytou and K. Mueller. Color-Space CAD: Direct Gamut Editing in 3D.

IEEE Computer Graphics and Applications. 28(3): 88–98, 2008.

[65] C.K. Ogden. *Basic English: a general introduction with rules and grammar*. K. Paul, Trench, Trubner. 1944.

[66] A. Orzan, A. Bousseau, P. Barla and J. Thollot. Structure-preserving manipulation of photographs. *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*. : 103-110, 2007.

[67] S. Palmer, E. Rosch and P. Chase. Canonical perspective and the perception of objects. In *Attention and Performance IX: Proceedings of the Ninth International Symposium on Attention and Performance, Jesus College, Cambridge, England, July 13-18, 1980*, pages 135, 1981.

[68] R. Pawlicki, I. Kókai, J. Finger and others. Navigating in a shape space of registered models. *IEEE Transactions on Visualization and Computer Graphics*. 13(6): 1552–1559, 2007.

[69] P. Pérez, M. Gangnet and A. Blake. Poisson image editing. *ACM Trans. Graph.* 22(3): 313-318, 2003.

[70] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of International Conference on Intelligence Analysis*, pages 2–4, 2005.

[71] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*. 77(2): 257–286, 1989.

[72] R. Raguram and S. Lazebnik. Computing iconic summaries of general visual concepts. In *Proc. of IEEE CVPR Workshop on Internet Vision*2008.

[73] R. Raina, A. Battle, H. Lee, B. Packer and A.Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 766, 2007.

[74] B.E. Rogowitz and A.D. Kalvin. The "Which Blair Project": a quick visual method for evaluating perceptual color maps. In *Proceedings of the conference on Visualization '01*(San Diego, California), , pages 183-190, 2001.

[75] C. Rother, L. Bordeaux, Y. Hamadi and A. Blake. AutoCollage. In *ACM SIGGRAPH 2006 Papers*, pages 847–852, 2006.

[76] S. Rudolph, A. Savikhin and D.S. Ebert. FinVis: Applied Visual Analytics for Personal Financial Planning. (Atlantic City, NY), 2009.

[77] E.A. Rundensteiner, M.O. Ward, Z. Xie, Q. Cui, C.V. Wad, D. Yang and S. Huang. Xmdvtool Q:: quality-aware interactive data exploration. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1112, 2007.

[78] T. Schreck, J. Bernard, T. Tekusova and J. Kohlhammer. Visual cluster analysis of trajectory data with interactive Kohonen maps. In *Proceedings of IEEE VAST*, pages 3–10, 2008.

[79] V. Setlur, C. Albrecht-Buehler, A. Gooch, S. Rossoff and B. Gooch. Semanticons:

Visual metaphors as file icons. In *Computer Graphics Forum*, pages 647–656, 2005.

[80] B. Settles. Active Learning Literature Survey. Technical Report #1648. University of Wisconsin-Madison. 2009.

[81] I. Simon, N. Snavely and S.M. Seitz. Scene summarization for online image collections. In *Proc. of ICCV*, pages 1–8, 2007.

[82] R. Spence. *Information visualization*. Addison-Wesley Reading, MA. 2000.

[83] G. Spenkelink and J. Besuijen. Chromaticity contrast, luminance contrast, and legibility of text. *Journal of the Society for Information Display*. 4(3): 135-144, 1996.

[84] J. Stasko, C. G\örg and Z. Liu. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*. 7(2): 118–132, 2008.

[85] M.C. Stone. *A field guide to digital color*. AK Peters, Ltd. 2003.

[86] M.C. Stone. Color in information display. In *IEEE Visualization*October 2008.

[87] C. Strothotte and T. Strothotte. *Seeing between the pixels: pictures in interactive systems*. Springer-Verlag New York, Inc. New York, NY, USA. 1997.

[88] T. Strothotte and S. Schlechtweg. *Non-photorealistic computer graphics: modeling, rendering, and animation*. Morgan Kaufmann Pub. 2002.

[89] D.F. Swayne, D.T. Lang, A. Buja and D. Cook. GGobi: Evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics and Data Analysis*. 43(4): 423-444, 2003.

[90] S.T. Teoh, K.L. Ma, S.F. Wu and T.J. Jankun-Kelly. Detecting flaws and intruders with visual data analysis. *IEEE Computer Graphics and Applications*. : 27–35, 2004.

[91] D.R. Tesone and J.R. Goodall. Balancing Interactive Data Management of Massive Data with Situational Awareness through Smart Aggregation. In *IEEE Symposium on Visual Analytics Science and Technology, 2007. VAST 2007*, pages 67–74, 2007.

[92] J.J. Thomas and K.A. Cook. Illuminating the path: The research and development agenda for visual analytics. *IEEE Computer Society*. , 2005.

[93] D.S. Travis, S. Bowles, J. Seton and R. Peppe. Reading from color displays: A psychophysical model. *Human Factors*. 32(2): 147–156, 1990.

[94] A. Vinciarelli and S. Favre. Broadcast News Story Segmentation using Social Network Analysis and Hidden Markov Models. In *Proceedings of the 15th international conference on Multimedia*, pages 264-267, 2007.

[95] L. Wang, J. Giesen, K.T. McDonnell, P. Zolliker and K. Mueller. Color Design for Illustrative Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 14(6): 1739-1754, 2008.

[96] C. Ware. *Information visualization: perception for design*. Morgan Kaufmann. 2004.

[97] M. Wijffelaars, R. Vliegen, J.J. van Wijk and E.J. van der Linden. Generating Color Palettes using Intuitive Parameters. In *Computer Graphics Forum*, pages 743-750, 2008.

[98] W. Wong. *Principles of color design*. Wiley. 1996.

[99] L. Xiao, J. Gerth and P. Hanrahan. Enhancing visual analysis of network traffic using a knowledge representation. *IEEE VAST 2006*. : 107-114, 2006.

[100] D. Yang, E.A. Rundensteiner and M.O. Ward. Analysis guided visual exploration of multivariate data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 83–90, 2007.

[101] J. Yang, J. Fan, D. Hubball, Y. Gao, H. Luo, W. Ribarsky and M. Ward. Semantic image browser: Bridging information visualization with automated intelligent image analysis. In *IEEE symp visual analytics science and technology. IEEE, Piscataway*, pages 191–198, 2006.

[102] K.B. Zhang, M. Huang, M. Orgun and Q. Nguyen. A Visual Method for High-Dimensional Data Cluster Exploration. In *Proceedings of the 16th International Conference on Neural Information Processing*, pages 699–709, 2009.

[103] Z. Zhang, S. Garg, A. Dimitriyadi, I.V. Ramakrishnan, R. Zhao, A. Viccellio and K. Mueller. A Visual Analytics Framework for Emergency Room Clinical Encounters. In *IEEE VisWeek Workshop on Visual Analytics in Health Care*(Salt Lake City, UT), 2010.

[104] X.S. Zhou and T.S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*. 8(6): 536–544, 2003.

[105] X. Zhu. Semi-Supervised Learning Literature Survey. *Technical Report 1530*. , 2005.