ReVision: Automated Classification, Analysis and Redesign of Chart Images

Anonymous
Institution
Location
email@host.domain

ABSTRACT

Poorly designed charts are prevalent in reports, magazines, books and on the Web. Yet, most of these charts are only available as bitmap images. Without access to the underlying data it is prohibitively difficult for viewers to create more effective visual representations. In response we present *ReVision*, a system that automatically redesigns visualizations to improve graphical perception. Given a bitmap image of a chart as input, our system applies computer vision and machine learning techniques to identify the chart type (e.g. pie chart, bar chart, scatterplot, etc.). It then extracts the graphical marks and the data encoded by each mark. Using the resulting data table ReVision applies perceptually-based design principles to populate an interactive gallery that enables users to view alternative chart designs and retarget content to different visual styles.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

Keywords: Visualization, chart understanding, information extraction, redesign, computer vision.

INTRODUCTION

Over the last 300 years, charts, graphs, and other visual depictions of data have become a primary vehicle for communicating quantitative information [27]. However, designers of visualizations must navigate a number of decisions, including choices of visual encoding and styling. These decisions influence the look-and-feel of a graphic and can have a profound effect on graphical perception [6]: the ability of viewers to decode information from a chart. Despite progress in the development of design principles for effective visualization [6, 11, 16, 19, 20, 23, 27], many charts produced today exhibit poor design choices that hamper understanding of the underlying data and lead to unaesthetic displays.

Consider the pie chart in Figure 1(a), which depicts data concerning the 2005 research budget of the National Institutes of Health (NIH). The design of this chart could be improved in multiple ways: slices are ordered haphazardly, labels are placed erratically, and label text competes with saturated background colors. Moreover, the chart encodes values as angular extents, which are known to result in less accurate value comparisons than position encodings [6, 16, 23]. Figure 1(b) shows the same data in a redesigned visualization: the bar chart sorts the data, uses a perceptually-motivated color palette [25], and applies a position encoding of values.

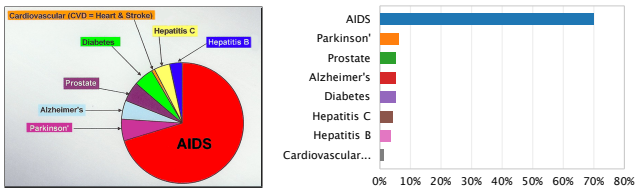


Figure 1: Chart Redesign. (a) A pie chart of NIH expenses per condition-related death. The chart suffers from random sorting, highly saturated colors, and erratic label placement. (b) Plotting the data as a sorted bar chart enables more accurate comparisons of data values [6, 19].

For analysts working with their own data, visual design principles [6, 11, 16, 27] and automated design methods [19, 20] can lead to more effective visualizations. However, the vast majority of visualizations are only available as bitmap images. Without access to the underlying data it is prohibitively difficult to create alternative visual representations.

We present *ReVision*, a system that takes bitmap images of charts as input and automatically generates redesigned visualizations as output. For example, ReVision produces Figure 1(b) as a suggested redesign when given Figure 1(a) as input. Our system identifies the type of chart, extracts the marks and underlying data, and then uses this data in tandem with a list of guidelines to provide alternate designs. ReVision also supports stylistic redesign; users can change mark types, colors or fonts to adhere to a specific visual aesthetic or brand. We make the following research contributions:

Classification of Chart Images. ReVision determines the type of chart using both low-level image features and extracted text-level features. We propose a novel set of features that achieve an average classification accuracy of 96%. Our image features are more accurate, simpler to compute, and easier to interpret than those used in prior work. While we focus on the task of chart redesign, our classification method is applicable to additional tasks such as indexing and retrieval.

UIST'11, October 16-19, 2011, Santa Barbara, CA, USA.

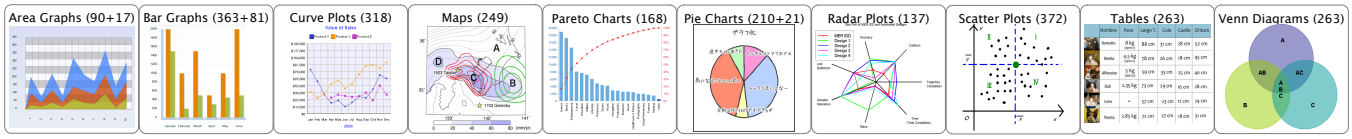


Figure 2: Our 10-category chart image corpus. Numbers in parentheses: (strictly 2D images + images with 3D effects).

Extraction of Chart Data. We present a set of chart-specific techniques for extracting the graphical marks and the data values they represent from a visualization. Our implementation focuses on bar and pie charts. We first locate the bars and pie slices that encode the data. We then introduce heuristics for associating these marks with related elements such as axes and labels. Finally, we use this information about the graphical marks to extract the table of data values underlying the visualization. Limiting the classification corpus to bar and pie charts without 3D effects or non-solid shading, we achieve an average accuracy of 64.5% in extracting the marks, and an average accuracy of 66.5% in extracting the data for charts whose marks were successfully extracted. We note that the classification corpus contains a wide diversity of charts with respect to image resolution and quality. We detail the challenges of robustly extracting data from such chart images.

We demonstrate how these contributions can be combined with existing design guidelines to automatically generate a variety of alternative visualizations. In the process of pursuing these goals, we also compiled a corpus of more than 2,500 chart images labeled by chart type. We are making this corpus publicly available in the hope of spurring continued research on computational visualization interpretation.

RELATED WORK

Our ReVision system builds on three areas of related work.

Classifying Visualization Images

Classification of natural scenes is a well-studied problem in computer vision [1, 3]. A common approach is to use a "bag of visual words" representation [28] where words are low-level image features (e.g., gradients, local region textures, etc.). Each input image is encoded as a feature vector using these words. Machine learning methods then classify the feature vectors and corresponding images into categories.

Researchers have developed specialized techniques for classifying chart images by type (e.g., bar chart, pie chart, etc.). Shao and Futrelle [22] first extract high-level shape types (e.g., line segments) from vectorized representations of charts and then use these shape types as features for classifying six kinds of charts. However, their approach relies on accurate vectorization of charts, which frequently is not available.

Prasad et al. [21] classify bitmap images of charts drawn from five common categories. They apply a series of preprocessing and segmentation operations which require many input-dependent parameters and complicate generalization of their approach to more visualization categories. Additionally, they use common features [9, 10] to summarize high level properties of image regions and compare them using a multi-resolution pyramidal histogram scheme. The connection between these features and the types of graphical marks in the images is ambiguous. We are interested in image features at the level of graphical marks, as we hypothesize that they may predict the chart type more accurately.

Extracting Marks from Charts

A few researchers have investigated techniques for extracting marks from charts. Zhou and Tan [29] combine boundary tracing with the Hough transform to extract bars from bar charts. Huang et al. [17, 18] generate edge maps, vectorize the edge maps, and use rules to extract marks from bar, pie, line, and low-high charts. However, they focus on extracting the mark geometry rather than the underlying data. Their techniques also rely on an accurate edge map, which can be difficult to retrieve from many real-world charts, due to large variance in image quality. We designed our techniques to be more robust to such variance.

Automated Visualization Design

Researchers have applied graphical perception principles to automatically produce effective visualizations. Mackinlay's APT [19] generates charts guided by rankings of visual variables for specific data types (e.g., categorical, ordinal, or quantitative data). Stolte et al.'s Polaris [24] is a system for generating small multiples [27] displays based on user queries of a multidimensional data table. Mackinlay et al. [20] later extend this work to support a broader range of automated visualization designs. These systems assist users in producing visualizations directly from data; we seek to first extract data from chart images and then redesign the visualizations to improve graphical perception. Our current work is complementary to these earlier systems: once we have extracted the data table we could feed it into any of these systems to generate improved alternative designs.

SYSTEM OVERVIEW

ReVision is comprised of a three stage pipeline: 1) chart classification, 2) mark and data extraction and 3) redesign. In stage 1, Revision classifies an input image according to its chart type. This stage uses a corpus of test images to learn distinctive image features and train classifiers. In stage 2, ReVision locates graphical marks, associates them with text labels, and extracts a data table. In stage 3, ReVision uses the extracted data to generate a gallery of alternative designs.

STAGE 1: CLASSIFYING CHART IMAGES

While classification of natural images is a well-studied computer vision problem, chart images are unlike natural images in several important ways. First, marks within data graphics are more discrete and frequently repeated than textured regions in photographs. Second, there are many areas with constant color so pixel neighborhoods with very low variances are common. Finally, text occurs more frequently and prominently than in natural scenes. The following sections describe how we account for these differences to achieve classification accuracy superior to previous work [21].

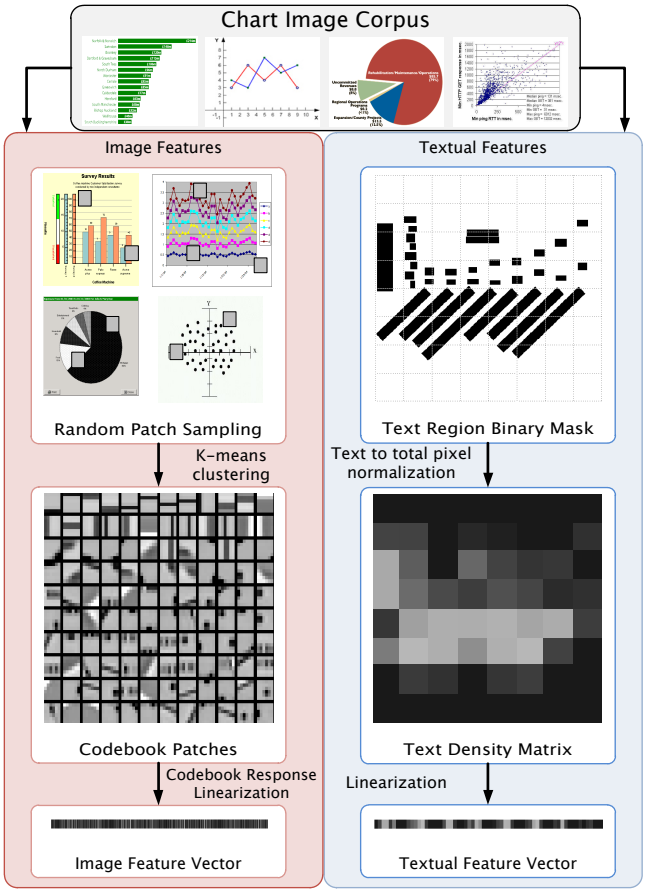


Figure 3: The ReVision classification pipeline, including both image (left) and textual (right) features.

Learning Low-Level Image Features

We capture the patterns of recurring graphical marks by constructing image features directly from sample pixel patches (small rectangular image regions). Such patches are predictive of chart categories and are easier to compute and interpret than more complex descriptors. As we will show, using image patches also achieves higher classification accuracy. We follow the approach of Coates et al. [7], who found that random patch sampling and K-means clustering achieve state-of-the-art performance for many types of images. We determine parameter values by evaluating classification accuracy in a series of cross-validation experiments.

Image Normalization. We normalize images to a constant size of $D \times D$ pixels to ensure homogeneous sampling and equivalence of each exemplar in the corpus. We preserve the aspect ratio of the original image by padding with the background color. We convert color images to grayscale, as generally color is not indicative of visualization category. We considered image sizes from $D = 32$ up to $D = 512$ and found that $D = 128$ achieves the best classification performance. We believe this is due to the reduction of noise and compression artifacts in the down-sampled images (the average image size in our collections is roughly 300×400 pixels).

Patch Extraction. We extract square patches from the images by uniform random sampling for 100 locations per image. We tested patch sizes over the range from 2×2 up to 20×20 pixels and found that a patch size of 6×6 pixels gave the best classification performance. The optimal patch size was consistently about 5% of the normalized image dimensions. To filter out frequently occurring constant color regions, we reject sample patches with variance less than 10% of the maximum pixel value.

Patch Standardization. For classification, absolute pixel values are not as important as variation within a patch. Thus, we normalize the contrast of each patch by subtracting the mean and dividing by the standard deviation of the pixel values in that patch. This normalization ensures that a single appropriately weighted patch can represent patches with different absolute intensities but similar variations. We perform a standard "whitening" procedure on the entire patch set which reduces the cross-correlation between patches and has been shown to improve classification performance [7].

Patch Clustering. Given an extracted patch set, we perform K-means clustering to obtain a set of "centroid" patches that correspond to the most frequently occurring patch types. In practice we set the number of centroids K to 200 as we found that larger numbers of centroids did not achieve better performance. The centroid patches constitute a feature set or "codebook" which we can use to describe our image corpus.

These codebook patches capture frequently occurring graphical marks such as lines, points, corners, arcs and gradients. By extracting codebooks from each visualization category independently we see that the patch set reflects the occurrence frequencies of marks characteristic of that category. As an example, in Figure 4 we compare the centroids for 3 categories from the corpus of [21]. We invite the reader to guess which categories are represented.[1]

Using Low-Level Image Features for Classification

To classify an input image, we first normalize it to a 128×128 grayscale image and extract 6×6 sized patches centered on each pixel. We then perform the following steps:

Codebook Patch Response. For each extracted patch, we determine the nearest codebook patch by Euclidean distance. We thus obtain a D^2 centroid response map over the entire image, where D is the normalized image dimension.

Feature Vector Formulation. We reduce the dimensionality of the codebook patch response map by dividing the image into quadrants and summing up the activations for each codebook patch in a given quadrant. We thus obtain a $4K$-length feature vector for each input image.

Classification. We use the feature vectors to perform classification using standard machine learning methods. We experimented with many classification algorithms including logistic regression, AdaBoosted decision trees [14] and Support Vector Machines (SVMs) [8]. We find that SVMs using a quadratic kernel function provide the best performance.

[1] Bar Charts (left), Pie Charts (middle), Scatter Plots (right). Bar corners, pie arcs and scatter points are prevalent within their respective categories.

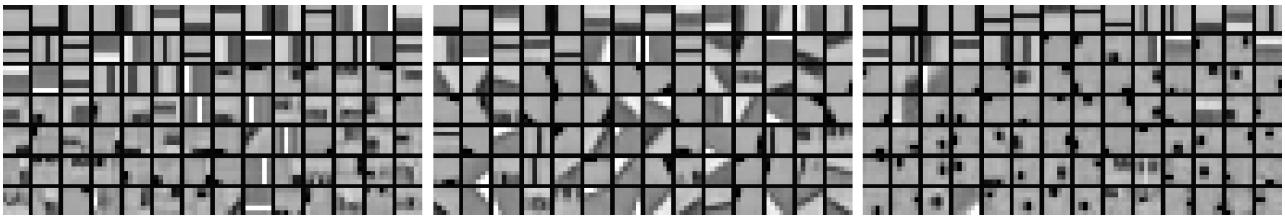


Figure 4: Codebook patches obtained from 3 different visualization categories. We invite the reader to guess which categories were sampled. The answers are in the footnote at the end of the section "Learning Low-Level Image Features."

Using Text-Level Features to Improve Classification

While image features offer powerful cues to determine the category of a chart image, text-level features also contain useful information. The position and size of text regions in chart images may correlate with the type of visualization. Interestingly, prior work [17] argues for *removing* text regions from chart images to reduce noise. We show that textual information can further improve classification accuracy.

We designed a tagging interface to annotate Prasad et al.'s corpus [21] with the position, size, angular orientation and content of text regions in each image. Our tool extracts the text image region and performs OCR; the user can correct the OCR output if desired. While manual text annotation is tedious, it is amenable to crowdsourcing; we designed the tagging UI with this scenario in mind. Furthermore, connected components analysis or more sophisticated approaches [4] could be used to extract text region bounding boxes automatically. These methods could allow fully unsupervised textual and image feature extraction from a visualization corpus.

To capture the distribution and density of text regions in an input image, we construct a binary mask indicating which pixels belong to text regions. We subdivide the mask into $N \times N$ blocks, calculate the proportion of text region pixels in each block, and linearize these values into an N^2 element vector. The procedure is illustrated in Figure 3. We set $N = 8$, as we have found that this value maximizes classification accuracy. However, at this resolution the exact positions, sizes and relative orientations of text regions are abstracted. To retain this information we compute normalized 10-bin histograms of text region length, width, center position, pairwise orientation, and pairwise distance between text region centers. These histograms are concatenated with the text density vector to create a final textual feature vector.

Classification Performance

We use two collections of chart images to evaluate our system. The first corpus comes from previous work on visualization image classification and was kindly provided by the authors of [21]. The corpus consists of 667 images collected from the Internet and categorized into 5 categories: bar charts (125), curve plots (121), pie charts (134), scatter plots (162) and 3D surface plots (125). We use this corpus to compare our results against the prior state-of-the-art.

We also compiled a new corpus containing 2,601 chart images labeled with 10 categories using a semi-automated tool for Google Image search. Example images from each category are given in Figure 2. We use this larger corpus to evaluate how our system scales to classifying images from larger sets of visualization categories.

	Prasad [21]	Image	Text	All	Binary
Bar	90%	85%	57%	89%	95%
Curve	76%	75%	50%	83%	92%
Pie	83%	93%	84%	95%	97%
Scatter	86%	91%	64%	91%	97%
Surface	84%	90%	71%	94%	97%
Average	**84%**	**88%**	**66%**	**90%**	**96%**

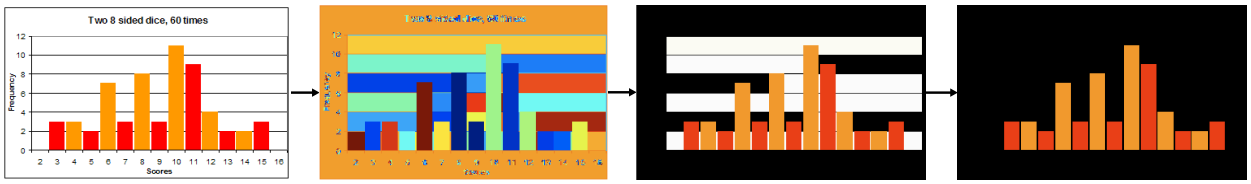
Table 1: Classification accuracy on the corpus in [21]. We compare prior results [21] to our method using image features, text features, and both in multi-class SVMs. The last column shows results for both features using binary SVMs.

	Multi-Class	**Binary**
Area Graphs	88%	98%
Bar Graphs	78%	95%
Curve Plots	73%	91%
Maps	84%	97%
Pareto Charts	85%	97%
Pie Charts	79%	97%
Radar Plots	88%	93%
Scatter Plots	79%	93%
Tables	86%	97%
Venn Diagrams	75%	97%
Average	**80%**	**96%**

Table 2: Classification accuracy on our 10-category corpus (2,601 images) using only image features.

For a known benchmark, we use the corpus of visualization images collected by Prasad et al. [21]. We first tested the performance of the learned image features for classification by training a multi-class SVM classifier using a quadratic kernel through 5-fold cross-validation on the 5 category image corpus. In each fold, we randomly pick $4/5$ of the images for training and the remainder for testing. Results are then averaged over the 5 folds. Table 1 summarizes how our results compare to the ones reported by Prasad et al. (compare the first two columns). Our weighted average classification accuracy is 88%, exceeding the reported average accuracy of 84% of Prasad et al. by 4%. Incorporating text features further increases our accuracy to 90%. We also use the same procedure on our larger 10-category corpus where we obtain an average classification accuracy of 80% (see Table 2).

As both graphical marks and structural elements may be used across multiple chart categories (e.g., in hybrid chart types), we relax the constraint of disjoint class categorization. Binary classifiers are more appropriate for a system like ReVision, where we can evaluate which positive categorizations are valid in the data extraction part of the pipeline through er-



(a) Input chart (b) Connected components (c) Rectangular components (d) Bars

Figure 5: Bar extraction procedure. We compute connected components (shown false-colored) and discard non-rectangular components. We keep rectangles whose color differs from outer colors (see inset figure on this page) and that touch the inferred x-axis.

ror estimation and failure detection. To perform multi-label categorization we construct a bank of binary 1–vs–all classifiers. We train a binary SVM for a single category by using in-category images as positive examples and all others as negative examples. The results obtained using this method are given in the rightmost columns of Tables 1 and 2: we achieve an average accuracy of 96% on both corpora.

STAGE 2: MARK AND DATA EXTRACTION

After categorizing charts by type, ReVision proceeds to locate graphical marks and extract data. Our current implementation focuses on mark and data extraction algorithms for bar and pie charts, two of the most popular chart types.

We have found that chart images from the web are often heavily compressed and noisy. To reduce compression artifacts and noise we first apply the bilateral filter [26] to each bar or pie chart image given by our chart classifier. The bilateral filter smooths away small variations in color but retains sharp edges in the image. Next we perform *mark extraction* which locates geometric marks such as bars, pie slices, and axes by fitting models of expected shapes. Then we perform *data extraction* which infers the mapping between the data space and the image space, associates labels with marks and then extracts a data tuple for each mark.

Our algorithms are based on a few simplifying assumptions:

- Charts contain 2D marks and do not contain 3D effects.
- Each mark is solidly shaded using a single color. Marks are not shaded using textures or steep gradients.
- Bar charts do not contain stacked bars.
- Bar chart axes appear at the left and bottom of the chart.
- Marks encode a data tuple, where at least one dimension is quantitative and one dimension is categorical. For example, in a bar chart the height of a bar represents a quantitative value, while a category label may appear below.

Based on these assumptions we develop simple, robust data extraction algorithms while still encompassing a significant number of real-world charts. For example, 42% (52/125) of the bar and 43% (53/125) of the pie charts in the Prasad corpus [21] follow these assumptions.

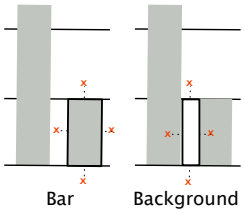
Mark Extraction

In this stage, we locate the marks. For bar charts, we extract the bars and axes. For pie charts, we extract the pie location and slice borders. We discuss each in turn.

Bar Charts

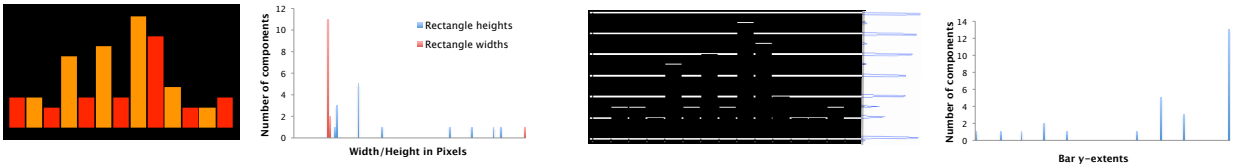
Figure 5 shows the steps in our bar extraction algorithm. We identify candidate bars by looking for rectangular connected components and then use the location of the candidate bars to extract the axes.

We first extract connected components from the filtered image by grouping adjacent pixels of similar color – i.e., whose L_2-norm in normalized RGB space is less than 0.04 (Figure 5b). We then find rectangular connected components by computing how much each component fills its bounding box. If component pixels fill more than 95% of the bounding box, we classify the component as a rectangle; otherwise, we discard the component. We also discard very small and thin rectangular component (i.e., width or height less than 2 pixels) as such small components are unlikely to represent bars (Figure 5c).



The remaining set of rectangles are likely to include all of the true data encoding bars, but may also include background rectangles formed by gridlines. In Figure 5c the remaining background rectangles are shown in white, along with the true bars. A key difference between the background rectangles and the true bars is that the true bars are surrounded on all sides by background, while background rectangles are likely to adjoin other background regions. We test for background rectangles by selecting one point outside each of the four sides of each rectangular component and compare the colors of these points to the average color of the pixels inside the rectangle, as shown in the inset figure. If any one of the sampled points has the same color as the average interior color, we classify the rectangle as part of the background and discard it; otherwise, we assume it is a candidate bar.

Bar charts may be oriented horizontally or vertically. Since bars encode data using length they vary in one dimension and remain fixed in the other dimension (e.g., vertical bars vary in height but maintain a fixed width). To identify which dimension varies most we build histograms of the widths and heights of the candidate bars and find the mode of each histogram (Figure 6a). The histogram with the strongest mode represents the constant dimension and gives us the orientation of the chart: vertical if the width is constant, horizontal if the height is constant. For brevity, in the remainder of this section we will describe our algorithms assuming a vertical



(a) Candidate bars / width and height histograms

(b) Y-gradient image / candidate bar Y-value histogram

Figure 6: Inferring the orientation of the chart and locating the x-axis. (a) Extracted bars and histograms of widths and heights of the bars. Note that the mode of the width histogram (red) is larger than the mode of the height histogram (blue), which indicates a vertical chart. (b) Inferring the x-axis. At left, the y-difference image and projections along the rows. At right, a histogram of the y-values of the bars. The mode of the y-value histogram matches a peak in the row projection imaged, which is the location of the x-axis.

bar chart, but our system works with horizontal bar charts using analogous algorithms.

Finally, we extract the axes of the bar chart. We assume a non-stacked bar chart, so every bar must touch the x-axis. Therefore we build a histogram of the extreme y-values of the candidate bars. The mode of this histogram corresponds to the most common extreme y-value and represents the horizontal line that most bars touch. We treat this mode as the x-axis (Figure 6b). To ensure we have the correct axis, we compute row projections of a column-difference image and check for a peak near our tentative x-axis (Figure 6b). We discard any candidate bar that does not touch the x-axis, which leaves us with final candidate set of data encoding bars (Figure 5d).

We do not extract the y-axis for bar charts. As we will show in the data extraction section, we can extract the data value for the bars without knowing the location of the y-axis.

Pie Charts

Mark extraction in pie charts involves two phases. We first fit an ellipse to the pie. We then unroll the pie and locate strong differences in color to locate the pie sector edges.

Although most pie charts are circular, some are elliptical. For greatest generality, we model the pie as an ellipse. We start by computing the gradient of the chart and threshold to find pixels near sharp changes in color. We set the threshold such that at least $1/30$ of the image pixels are kept as gradient pixels. We use the text region tags from the classification component to remove gradient pixels that result from text.

We then adapt the RANSAC [12] algorithm to fit an ellipse to the gradient pixels. We use RANSAC because of its robustness to outliers. In brief, our algorithm works as follows: we first randomly selects 4 gradient pixels and compute an ellipse that passes through them using Fitzgibbon et al.'s direct least-squares ellipse fitting method [13]. We then find the set of gradient pixels that are at most 10 pixels away from the ellipse and call them *inliers*. Next we check how well the ellipse explains the inliers by computing an ellipse score. The score is a weighted sum of three components: the circularity of the ellipse (ratio of the minor to major axis), how tightly it fits the inliers (average distance of an inlier to the ellipse), and coverage (how much of the ellipse is near inliers). Higher scores indicate better ellipses. Examples of low scoring ellipes are shown in Figure 7. We keep the ellipse if its score is higher than the previous highest score. We iterate this process 20,000 times, which we experimentally found to work well for our corpora.

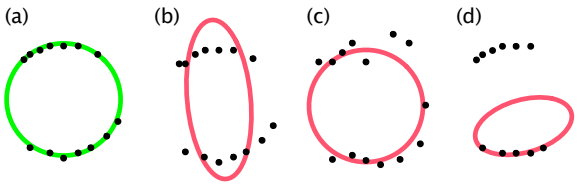
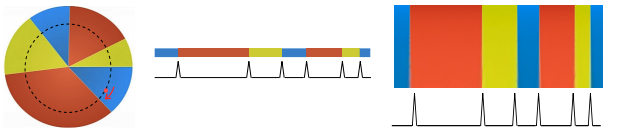


Figure 7: A high scoring ellipse and three low scoring ellipses. (a) An high scoring ellipse maximizes, circularity, goodness of fit and coverage. (b) An ellipse with low circularity score, (c) low fit score, and (d) low coverage score.



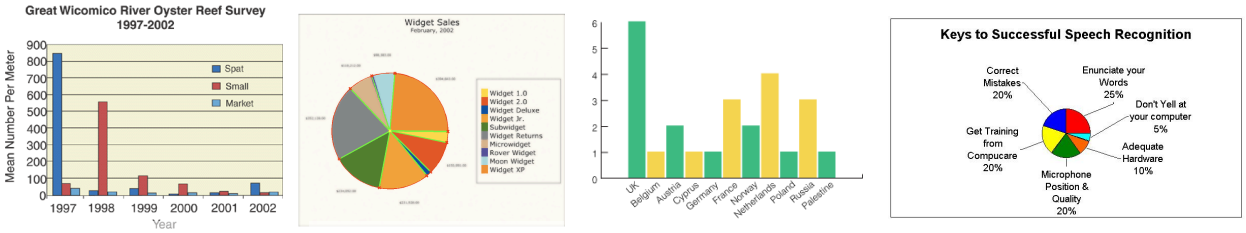
(a) Chart (b) Unrolled ellipse and difference image (c) Unrolled pie using ellipses as different radii

Figure 8: Unrolling the pie. Consider the inner ellipse marked in (a). We unroll the ellipse by sampling points at constant angular intervals (b). Peaks in the horizontal derivative occur at sector edges. To improve our edge estimation, we unroll multiple ellipses and sum their horizontal derivatives (c). Peaks in the summed horizontal derivatives occur at sector edges.

To extract the pie sector edges we first "unroll the pie"; we sample an ellipse located inside the pie at constant angular intervals creating a one-dimensional ellipse image (Figure 8a). We then take the horizontal derivative of this ellipse image to identify strong changes in color. Color changes indicate transitions between pie sectors and therefore the peaks in the derivative gives us estimates for the sector edge locations (Figure 8b). To increase robustness of our estiamtes, we unroll multiple concentric ellipses and sum their horizontal derivatives (Figure 8c). We identify peaks in the summed derivatives by looking for zero crossings of its first derivative. Finally we retain all peaks that are more than one standard deviation above the mean of the summed derivatives.

Data Extraction

In the data extraction step, our goal is to recover the data encoded by each mark. In general, a mark encodes a tuple of data, as stated in our assumptions. We recover these encodings by using the geometry of the extracted marks and



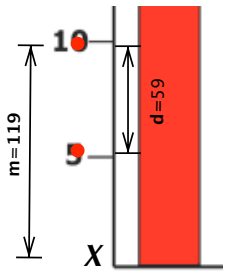
(a) Some bars are too small (b) Some slices are too small (c) Rotated labels cause mislabeling (d) Callouts cause mislabeling

Figure 9: Mark and data extraction failures. If marks are very small, our algorithms fail to extract them (a, b). Data extraction failures occur when marks are mislabeled, e.g., when labels are rotated or if the chart places labels far from their associated marks (c, d).

the text region tags from the classification component. After data extraction, we output a data table that contains an id and data tuple for each mark.

Bar Charts

To recover the data from a bar chart, we infer first the mapping between image space and data space. While we assume a linear mapping, we believe our data extraction techniques are amenable to other mappings (e.g. logarithmic) . The linear mapping is fully defined by a scaling factor between image space and data space, and the minimum value (usually the data value at the x-axis) . We first recover the scaling factor by considering the y-axis labels. We identify the y-axis labels that encode data values by finding text regions that are equidistant from the leftmost bar and line up vertically. We then estimate the scaling factor using each pair of value labels, as illustrated by the inset figure. The distance between labels "5" and "10" is $d = 60$, and the estimated scaling factor is $60/(10-5) = 12$. We take the median of the estimated scaling factors computed for each pair of labels to be the chart's scaling factor. For this chart, it was 12.5.



We then find the minimum value. We begin with the y-axis label vertically closest to the x-axis. If this label is 0, we set the minimum value to 0. Otherwise, we compute the minimum value using a similar procedure to computing the scaling factors. For each label, we estimate a minimum value by using the y-distance from the label's center to the x-axis and the chart scaling factor. For example, consider the inset figure above. Here, we assume a chart scaling factor of 12 and use the location of the "10" label to estimate the minimum value. We find $X = 119/12 - 10 = -0.08$. The actual minimum is 0, so we are not exact, but close. We set the chart's minimum value to the median of these minimum value estimates. For this chart, the median minimum value was -0.2.

Finally, we assign a category to each bar by associating it with the nearest label below the x-axis.

Pie Charts

Sectors in a pie chart frequently encode two pieces of data: they represent percentages of a whole and a level of a categorical variable. We compute the percentages they encode directly from the recovered borders of the sectors. We then assign a label to each sector by using the nearest text label to the ellipse arc that spans the sector, which we approximate using short line segments.

Extraction Results

We used the subset of the corpus of Prasad et al. [21] that met our assumptions, resulting in 52 bar charts and 53 pie charts. We assume a priori knowledge of the chart type, noting that our classification component provides this information.

We report mark extraction and data extraction accuracy results separately, as accurate data extraction depends on accurate mark extraction. We were able to successfully extract marks for 35/52 (67%) of bar charts and 33/53 (62%) of pie charts. Most of the failures were caused by very small marks (Figure 9a, b). We discard or smooth small elements in both bar and pie charts to combat image artifacts, but this sometimes results in legitimate marks being missed. In addition, the small size of many of the charts exacerbate the problem; the average dimension of the charts we used was 411 pixels.

Using the charts from which we were able to extract all marks, we were able to successfully extract the data tuples for 24/35 (69%) of bar charts and 21/33 (64%) of pie charts. We used a simple closest-label heuristic to associate marks with text labels. This heuristic may fail when labels are rotated (Figure 9c), or when marks are small and are labeled with callouts (Figure 9d).

Some charts contained data value labels for each mark (e.g., the exact percentage a pie sector represented), which gives us ground truth values from which we can estimate extraction error. Of our successful extractions, 11 bar charts and 17 pie charts had data value labels. For bar charts, our extracted values were on average within 6.8% of the original data. For pie charts, our extracted values were on average within 4.6% of the original data, and within 0.33 absolute percentage points. These are reasonable error rates given the small sizes of many of the charts.

STAGE 3: REDESIGN

The output of the data extraction process is a relational data table. ReVision uses this data to populate a gallery of alternative visualizations (Figure 10). The gallery displays a variety of visualizations sorted by *effectiveness* [19]; we use rankings of visual encodings from prior work [19] informed by graphical perception experiments (c.f., [6, 16, 23]).

ReVision chooses different visualizations depending on the

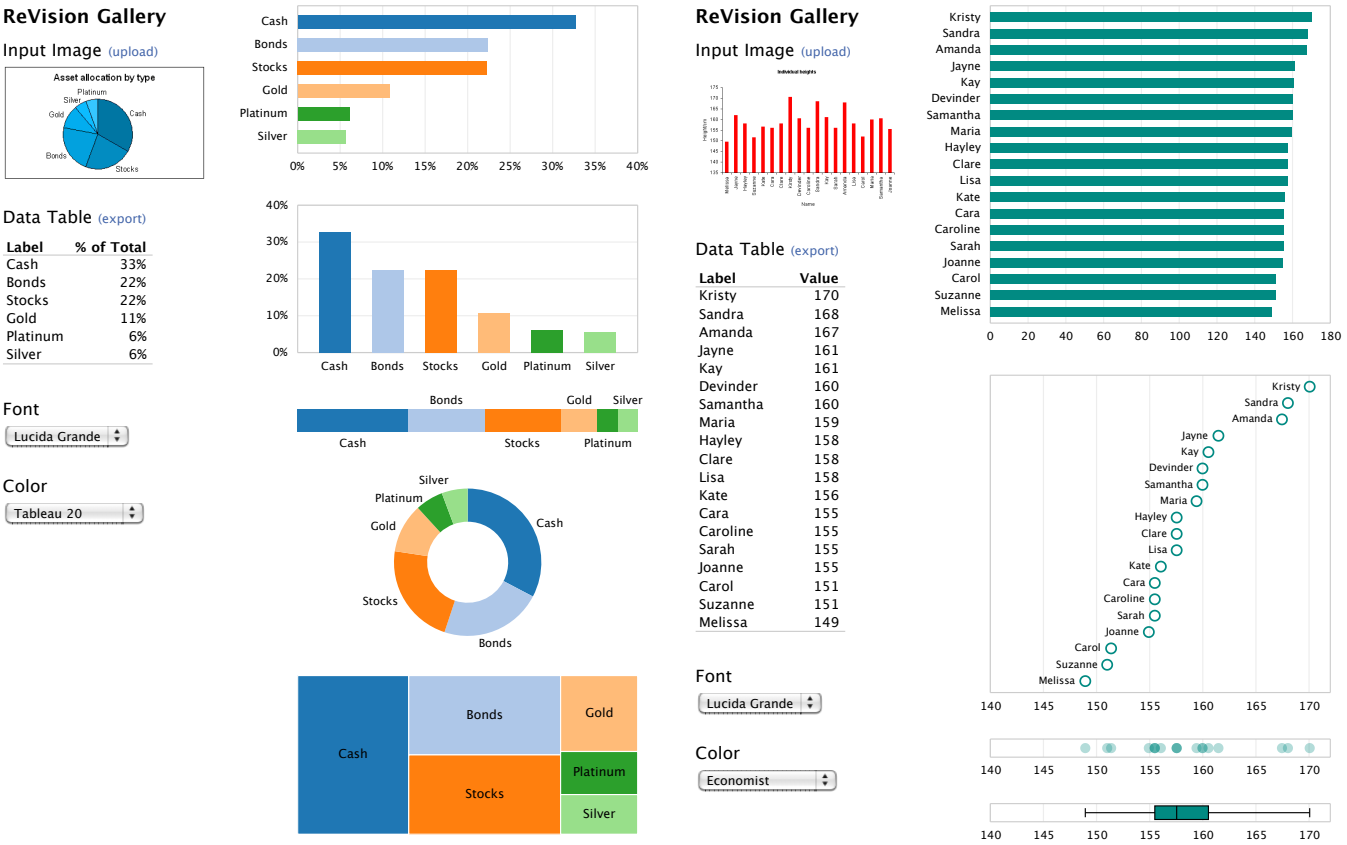


Figure 10: ReVision Design Galleries. Given an extracted data table, the gallery presents a variety of chart types, sorted by proposed perceptual effectiveness rankings [19]. Users can also select and compare color schemes and typefaces.

input chart type and extracted data. For the input pie chart in Figure 10(a), the gallery presents bar charts to support part-to-part comparisons and divided bar, donut, and treemap charts to support part-to-whole judgments [23]. For the input bar chart in Figure 10(b), ReVision generates a bar chart and labeled dot plot to support comparison of individual values and small dot and box plots to enable assessment of the overall distribution. Note that the y-axis of the input chart does not start at zero; ReVision's bar chart correctly incorporates a zero baseline, while more appropriate charts (dot and box plots) are used to focus in on the data range.

In addition, users can select and compare choices of font and color palette. ReVision includes palettes designed for well-spaced, perceptually discriminable colors [15, 25], as well as palettes from popular charting tools and news magazines. We generate charts using Protovis [2]; viewers can export the Protovis definitions for subsequent modification or reuse. Alternatively, users can export the data to create their own charts using tools such as Microsoft Excel or Tableau [20].

LIMITATIONS AND FUTURE WORK

In its current state, ReVision is suitable for generating alternative visualizations for many published bar and pie charts. However, more work is needed to overcome limitations of our classification and extraction methods.

With respect to classification, we currently employ a manual approach to annotating text regions. To achieve fully automatic processing of textual features, we would need to employ text identification and extraction algorithms. We do not yet use the text itself for classification; it is possible that such information might further improve classification accuracy.

Another direction for future research is to explore category inference for unlabeled image sets. Different categories of chart images give rise to qualitatively different codebook patches. We might leverage this to design a system to infer chart categories in an unsupervised fashion. These categories might be matched to known categories or labeled by hand.

ReVision might better leverage image features detected during classification to inform data extraction. Using the feature distributions over the image to predict the positions of graphical marks might assist more robust data extraction methods.

Our mark extraction algorithms are ripe for additional research. Our extraction techniques do not yet handle textured marks or 3D effects (e.g., due to gradient shading or perspective) both of which are common in our corpora. As 3D effects and textural patterns have been shown to hamper graphical perception [5, 11], these charts are ripe for redesign. Unfortunately, recognizing general 3D effects and textures is beyond the scope of our current techniques.

Our extraction algorithms do not yet parse legends. This

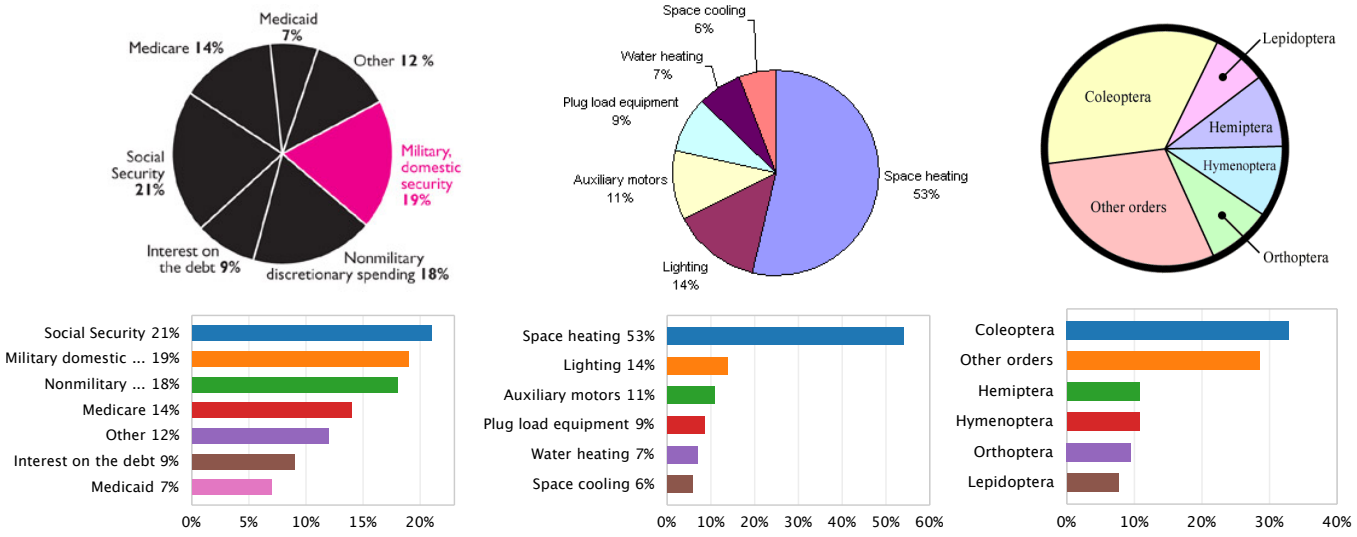


Figure 11: Example ReVision redesigns for input pie charts.

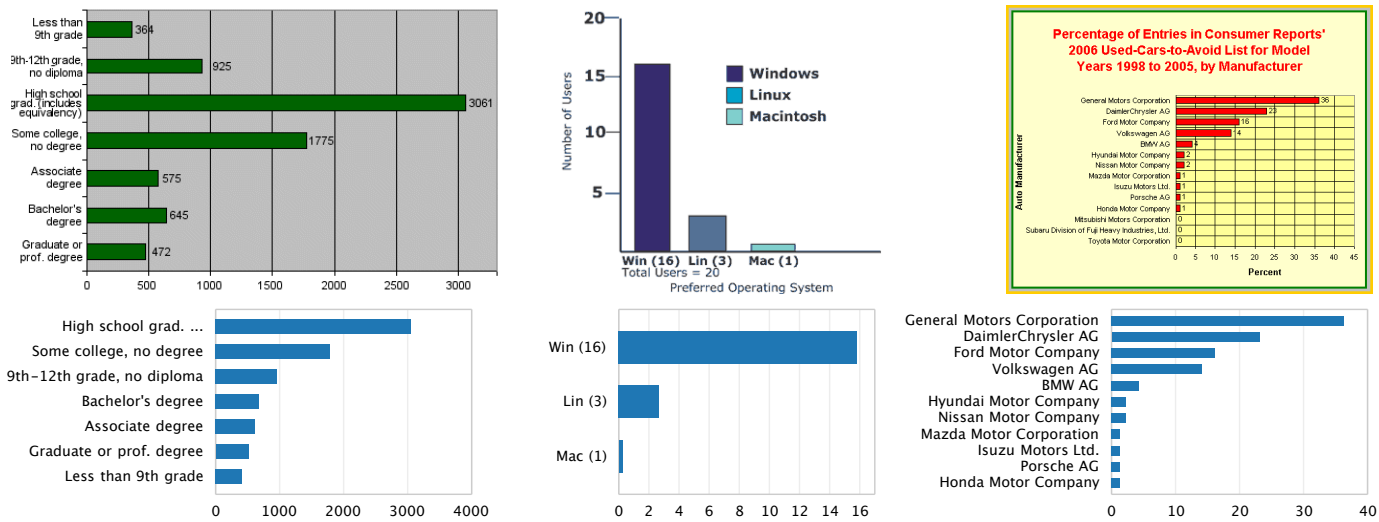


Figure 12: Example ReVision redesigns for input bar charts.

oversight causes failures on charts where the legend provides necessary labeling information, as in in grouped bar or pie charts with categorical color encodings documented in a legend. Finally, we plan to extend our extraction algorithms to additional chart types, such as bubble, line, and donut charts. Parts of our existing algorithms, such as scale estimation and shape fitting, can be readily adapted to these chart types.

CONCLUSION

Although visualizations are becoming more numerous, many could be productively redesigned. We have presented ReVision, a system that classifies charts, extracts the marks and data from the charts, and uses perceptually-based design principles to create automated redesigns. ReVision then presents users with an interactive gallery that allows them to retarget content to different visual styles. Automated redesign is only one of many possible applications for a system that extracts data from charts. Our techniques could be used to enhance information retrieval systems with chart metadata, or support screen-reading of charts for blind users.

In addition to supporting redesign, we believe our approach may eventually contribute insights for developing models of graphical perception. Simple charts – such as the flat, solidly shaded bar charts we consider – are relatively easy to extract. Difficulties arise as additional complexity is introduced, whether due to compression artifacts or design elements. Future research might explore in what ways obstructions to automated visualization interpretation correlate with human decoding of the same images. If fruitful, such efforts may lead to improved methods for automated visualization design [19, 20] from source data.

ACKNOWLEDGMENTS

This section should be left blank for blind review

REFERENCES

1. A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. *Computer Vision–ECCV*, pages 517–530, 2006.
2. M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE Trans Visualization & Comp Graphics*, 15(6):1121–1128, 2009.
3. M. Boutell, C. Brown, and J. Luo. Review of the state of the art in semantic scene classification. *Rochester, NY, USA, Tech. Rep*, 2002.
4. D. Chen, J. Odobez, and H. Bourlard. Text detection and recognition in images and video frames. *Pattern Recognition*, 37(3):595–608, 2004.
5. W. S. Cleveland. *Visualizing Data*. Hobart Press, 1993.
6. W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.
7. A. Coates, H. Lee, and A. Ng. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. *Advances in Neural Information Processing Systems*, 2010.
8. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.
9. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE CVPR*, pages 886–893, 2005.
10. G. David. Distinctive image features from scale-invariant keypoints. *Intl Journal Comp Vision*, 60(2):91–110, 2004.
11. S. Few. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Analytics Press, Berkeley, CA, 2004.
12. M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, June 1981.
13. A. Fitzgibbon, M. Pilu, and R. Fisher. Direct least square fitting of ellipses. *IEEE Trans Pattern Analysis & Machine Intelligence*, 21(5):476 –480, 1999.
14. Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, pages 23–37, 1995.
15. M. Harrower and C. Brewer. Colorbrewer.org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.
16. J. Heer and M. Bostock. Crowdsourcing graphical perception: Using Mechanical Turk to assess visualization design. In *ACM CHI*, pages 203–212, 2010.
17. W. Huang, C. L. Tan, and W. K. Leow. Model-based chart image recognition. In J. Lladós and Y.-B. Kwon, editors, *Graphics Recognition*, volume 3088 of *Lecture Notes in Computer Science*, pages 87–99. Springer Berlin / Heidelberg, 2004.
18. R. Liu, W. Huang, and C. L. Tan. Extraction of vectorized graphical information from scientific chart images. In *Document Analysis & Recognition (ICDAR)*, pages 521–525, 2007.
19. J. D. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans on Graphics*, 5(2):110–141, 1986.
20. J. D. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *IEEE Trans Visualization & Comp Graphics*, 13(6):1137 –1144, 2007.
21. V. Prasad, B. Siddiquie, J. Golbeck, and L. Davis. Classifying Computer Generated Charts. In *Content-Based Multimedia Indexing Workshop*, pages 85–92. IEEE, 2007.
22. M. Shao and R. Futrelle. Recognition and classification of figures in pdf documents. In W. Liu and J. Lladós, editors, *Graphics Recognition. Ten Years Review and Future Perspectives*, volume 3926 of *Lecture Notes in Computer Science*, pages 231–242. Springer Berlin / Heidelberg, 2006.
23. D. Simkin and R. Hastie. An information-processing analysis of graph perception. *Journal of the American Statistical Association*, 82(398):454–465, 1987.
24. C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Trans Visualization & Comp Graphics*, 8(1):52 –65, 2002.
25. M. Stone. *A Field Guide to Digital Color*. A. K. Peters, 2003.
26. C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839 –846, Jan. 1998.
27. E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.
28. J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Workshop on Multimedia Information Retrieval*, pages 197–206, 2007.
29. Y. P. Zhou and C. L. Tan. Hough technique for bar charts detection and recognition in document images. In *Intl Conf on Image Processing*, pages 605–608, sept. 2000.