Pixel Bar Charts: A New Technique for Visualizing Large Multi-Attribute Data Sets without Aggregation

Daniel Keim[1], Ming C. Hao, Julian Ladisch[1], Meichun Hsu,
Umeshwar Dayal
Software Technology Laboratory
HP Laboratories Palo Alto

HPL-2001-92
April 11th , 2001*

E-mail: (ming_hao, mhsu, dayal)@hpl.hp.com, keim@informatik.uni-konstanz.de, julian@ladisch.de

pixel, bar chart, web transactions, visualization

Simple presentation graphics are intuitive and easy-to-use, but show only highly aggregated data and present only a very limited number of data values (as in the case of bar charts), and may have a high degree of overlap which may occlude a significant portion of the data values (as in the case of the x-y plots). In this paper, we therefore propose a generalization of traditional bar charts and x-y-plots which allows the visualization of large amounts of data. The basic idea is to use the pixels within the bars to present the detailed information of the data records. Our so-called *pixel bar charts* retain the intuitiveness of traditional bar charts while allowing very large data sets to be visualized in an effective way. We show that, for an effective pixel placement, we have to solve complex optimization problems, and present an algorithm which efficiently solves the problem. Our application using real-world e-commerce data shows the wide applicability and usefulness of our new idea.

* Internal Accession Date Only   Approved for External Publication

[1] Presently with the Computer Science Institute, University of Constance, Constance, Germany

Pixel Bar Charts: A New Technique for Visualizing Large Multi-Attribute Data Sets without Aggregation

Daniel Keim*, Ming C. Hao, Julian Ladisch*, Meichun Hsu, Umeshwar Dayal
Hewlett Packard Research Laboratories, Palo Alto, CA
(ming_hao, mhsu, dayal)@hpl.hp.com

Abstract

Simple presentation graphics are intuitive and easy-to-use, but show only highly aggregated data and present only a very limited number of data values (as in the case of bar charts), and may have a high degree of overlap which may occlude a significant portion of the data values (as in the case of the x-y plots). In this paper, we therefore propose a generalization of traditional bar charts and x-y-plots which allows the visualization of large amounts of data. The basic idea is to use the pixels within the bars to present the detailed information of the data records. Our so-called *pixel bar charts* retain the intuitiveness of traditional bar charts while allowing very large data sets to be visualized in an effective way. We show that, for an effective pixel placement, we have to solve complex optimization problems, and present an algorithm which efficiently solves the problem. Our application using real-world e-commerce data shows the wide applicability and usefulness of our new idea.

1. Introduction

Because of the fast technological progress, the amount of data which is stored in computers increases very quickly. Researchers from the University of Berkeley estimate that every year about 1 Exabyte of data is generated, with 99.997% available only in digital form. Today, computers typically record even simple transactions of everyday life, such as paying by credit card, using the telephone and shopping in e-commerce stores. This data is collected because business people believe that it is a potential source of valuable information and could provide a competitive advantage.

Finding the valuable information hidden in the data, however, is a difficult task. Visual data exploration techniques are indispensable to solving this problem. In most data mining systems, however, only simple graphics, such as bar charts, pie charts, x-y plots, etc., are used to support the data mining process. While simple graphics are intuitive and easy-to-use, they either:

- show highly aggregated data and actually present only a very limited number of data values (as in the case of bar charts or pie charts), or
- have a high degree of overlap which may occlude a significant portion of the data values (as in the case of x-y plots).

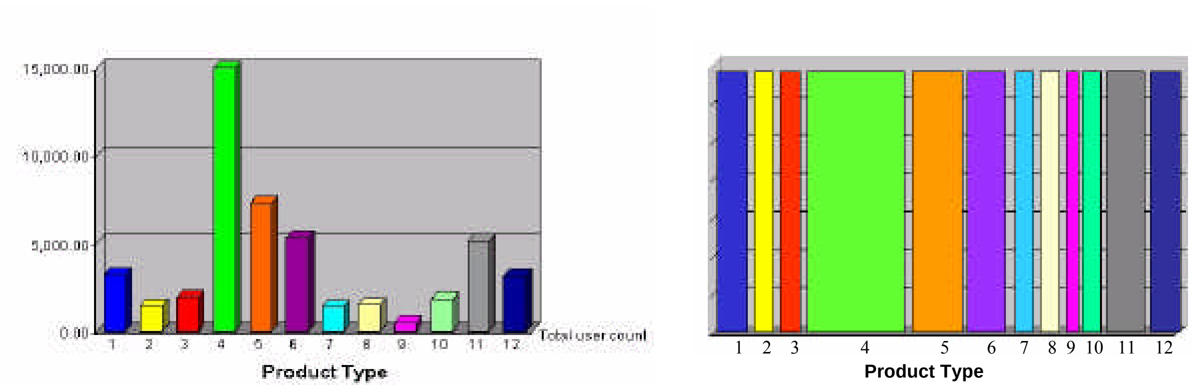
The usefulness of bar charts is especially limited if the user is interested in relationships between different attributes such as product type, price, number of orders, and quantities. The reason for this limitation is that multiple bar charts for different attributes do not support the discovery and correlation of interesting subsets, which is one of the main tasks in mining customer transaction data.

For an analysis of large volumes of e-commerce transactions [Eic 99], the visualization of highly aggregated data is not sufficient. What is needed is to present an overview of the data but at the same time show the detailed information for each data item.

In this paper, we describe a new visualization technique called *pixel bar chart*. The basic idea of pixel bar chart is to use the intuitive and widely used presentation paradigm of bar charts, but also use the available screen space to present more detailed information. By coloring the pixels within the different bars according to the values of the data records, very large amounts of data can be presented to the user. To make the display more meaningful, two parameters of the data records are used to impose an ordering on the pixels in the x- and y-directions. Pixel bar charts can be seen as a generalization of bar charts. They combine the general idea of x-y plots and bar charts to allow an overlap-free, non-aggregated display of multi-attribute data.

Since pixel bar charts use each pixel to present one data value, they belong to the class of pixel-oriented techniques. Other pixel-oriented techniques include the spiral technique [KK 94], the recursive pattern technique [KKA 95], and the circle segments technique [AKK 96]. Other classes of information visualization techniques include *geometric projection techniques* (e.g. [Ins 85, ID 90]), *icon-based techniques* (e.g.,

*Presently with the Computer Science Institute, University of Constance, Constance, Germany
keim@informatik.uni-konstanz.de ; julian@ladisch.de



a). Equal-Width Bar Chart b). Equal-Height Bar Chart

Figure 1: Regular Bar Charts

[PG 88, Bed 90]), *hierarchical techniques* (e.g., [LWW 90, RCM 91, Shn 92]), *graph-based techniques* (e.g., [EW 93, BEW 95]), which in general are combined with some *interaction techniques* (e.g., [BMMS 91, AWS 92, ADLP 95]) and sometimes also some *distortion techniques* [SB 94, LRP 95].

2. From Bar Charts to Pixel Bar Charts

A common method for visualizing large volumes of data is to use bar charts. Bar charts are widely used and are very intuitive and easy to understand. Figure 1 illustrates the use of a regular bar chart to visualize customer distribution in an e-commerce sales transaction. The height of the bars represents the number of customers for 12 different product categories.

Bar charts, however, require a high degree of data aggregation and actually show only a rather small number of data values (only 12 values are shown in Figure 1). Therefore, for data exploration of large multidimensional data, they are of limited value and are not able to show important information such as:

- data distributions of multiple attributes
- local patterns, correlations, and trends
- detailed information, e.g., each customer's profile (age, income, location, etc.)

2.1 Basic Idea of Pixel Bar Charts

Pixel bar charts are derived from regular bar charts (see Figure 1a). The basic idea of a pixel bar chart is to present the data values directly instead of aggregating them into a few data values. The approach is to represent each data item (e.g. a customer) by a single pixel in the bar chart. The detailed information of one attribute of each data item is encoded into the pixel color and can be accessed and displayed as needed.

One important question is: how are the pixels arranged within each bar? Our idea is to use one or two attributes to separate the data into bars and then use two additional attributes to impose an ordering within the bars (see Figure 2 for the general idea). The pixel bar chart can therefore be seen as a combination of the traditional bar charts and the x-y diagrams.

Now, we have a visualization in which one pixel corresponds to one customer. If the partitioning attribute is redundantly mapped to the colors of the pixels, we obtain the regular bar chart shown in Figure 1a (Figure 1b shows the equal-height-bar-chart" which we will explain in the next section). Pixel bar charts, however, can be used to present large amounts of detailed information. The one-to-one correspondence between customers and pixels allows us to use the color of the pixels to represent an additional attribute of the customer - for example, sales amount, number of visits, or sales quantity.

In Figure 3a, a pixel bar chart is used to visualize thousands of e-commerce sales transactions. Each pixel in the visualization represents one customer. The number of customers can be as large as the screen size (about 1.3 million). The pixel bar chart shown in Figure 3a uses product

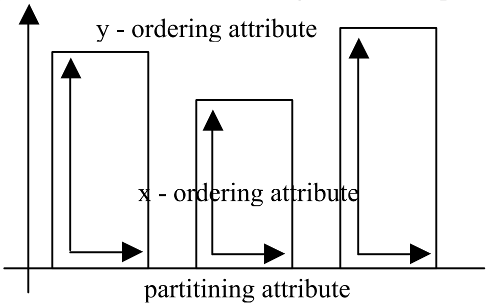
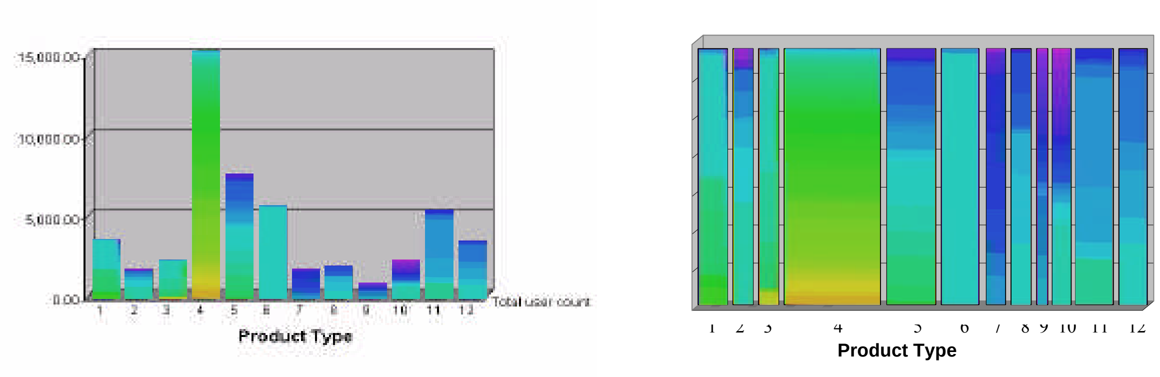


Figure 2: Pixel Bar Charts



a). Pixel Bar Chart

b). Space Filling Pixel Bar Chart

Figure 3: Pixel Bar Charts

type as the partitioning attribute and number of visits and dollar amount as the x and y ordering attributes. The color represents the dollar amount spent by the corresponding customer. High dollar amounts correspond to bright colors, low dollar amounts to dark colors.

2.2 Space-Filling Pixel Bar Charts

One problem of traditional bar charts is that a large portion of the screen space can not be used due to the differing heights of the bars. With very large data sets, we would like to use more of the available screen space to visualize the data. One idea that increases the number of displayable data values is to use equal-height instead of equal-width bar charts. In Figure 1b, the regular bar chart of Figure 1a is shown as an equal-height bar chart. The area (width) of the bars corresponds to the attribute shown, namely the number of customers.

If we now apply our pixel bar chart idea to the resulting bar charts, we obtain space-filling pixel bar charts which use virtually all pixels of the screen to display customer data items. In Figure 3b, we show an example of a space-filling pixel bar chart which uses the same partitioning, ordering, and coloring attributes as the pixel bar chart in Figure 3a. In this way, each customer is represented by one pixel.

Note that pixel bar charts generalize the idea of regular bar charts. If the partitioning and coloring attributes are identical, both types of pixel bar charts become scaled versions of their regular bar chart counterparts. The pixel bar chart can therefore be seen as a generalization of the regular bar charts but they contain significantly more information and allow a detailed analysis of large original data sets.

2.3 Multi-Pixel Bar Charts

In many cases, the data to be analyzed consists of multiple attributes. With pixel bar charts we can visualize attribute values using multi- pixel bar charts which use different color mappings but the same partitioning and ordering attributes. This means that the arrangement of data items within the corresponding bars of multi-pixel bar charts is the same, i.e., the colored pixels corresponding to the different attribute values of the same data item have a unique position in the bars. In Figure 4, we show an example of three pixel bar charts with product type as the partitioning attribute and number of visits and dollar amount as the x and y ordering attributes. The attributes which are mapped to color are dollar amount spent, number of visits, and sales quantity.

Note that the pixels in corresponding bars in multiple bar charts are related by their position, i.e., the same data record has the same relative position with each of the corresponding bars. It is therefore possible to relate the different bar charts and detect correlations.

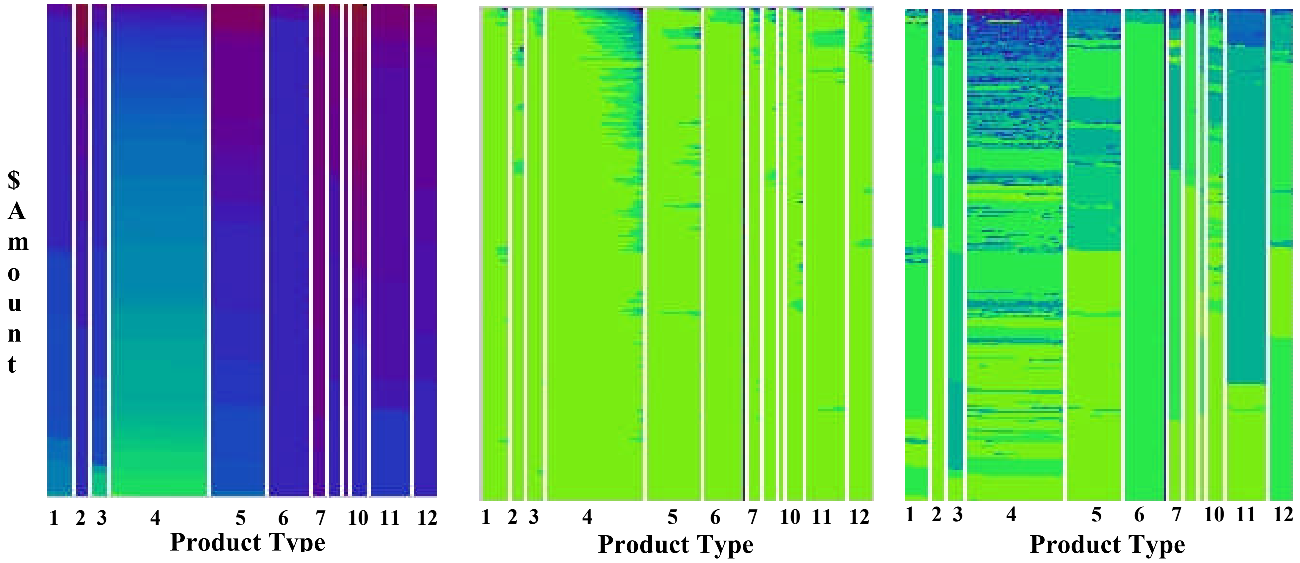
3. Formal Definition of Pixel Bar Charts

In this section we formally describe pixel bar charts and the problems that need to be solved in order to implement an effective pixel placement algorithm.

3.1 Definition of Pixel Bar Charts

For a general definition of pixel bar charts, we need to specify the:

- dividing attributes (for between-bar partitioning)
- ordering attributes (for within-bar ordering)
- coloring attributes (for pixel coloring).



a) C=dollar amount

b) C=number of visits

c) C=quantity

Figure 4: Multi Pixel Bar Chart Chart *(D_x= Product Type, D_y=$\perp$, O_x=number of visits, O_y=dollar amount, C)*

In traditional bar charts there is one dividing attribute which partitions the data into disjoint groups corresponding to the bars. In space-filling bar charts, the bars correspond to a partitioning of the screen according to the horizontal axis (x).

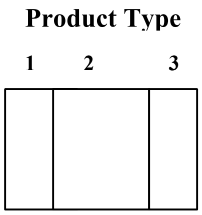


Figure 5: Dividing attribute on x-axis *(e.g., D_x = Product Type)*

We may generalize the definition of space-filling pixel bar charts by allowing more than one dividing attribute, i.e. one for the horizontal axis (D_x) and the one for the vertical axis (D_y).

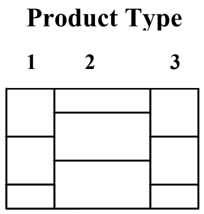


Figure 6: Dividing attributes on x- and y-axis *(e.g., D_x = Product Type, D_y= Region)*

Next, we need to specify an attribute for ordering the pixel in each pixel bar. Again, we can do the ordering according to the x- and the y-axis, i.e., along the horizontal (O_x) and vertical (O_y) axes inside each bar.

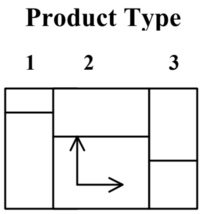


Figure 7: Ordering attributes on x- and y-axis *(e.g., O_x = Dollar Amount, O_y= Quantity)*

Finally, we need to specify an attribute for coloring the pixels. Note that in multi-bar charts we may assign different attributes to colors in different bar charts, which enables the user to relate the different coloring attributes and detect partial relationships among them. Note that the dividing and ordering attributes have to stay the same in order to do that.

Let $DB = \{d_1, ..., d_n\}$ be the data base of n data records, each consisting of k attribute values $d_i = \{a_1^i, \ldots, a_k^i\}$, $a_l^i \in A_l$, where A_l is the attribute name of value a_l. Formally, a pixel bar chart is defined by a five tuple:

$$<D_x, D_y, O_x, O_y, C>$$

where $D_x, D_y, O_x, O_y, C \in \{A_l, \ldots, A_k\} \cup \perp$ [1] and D_x/D_y are the dividing attributes in x-/y-direction, O_x/O_y are the ordering attributes in x-/y-direction, and C is the coloring attribute.
The multi-pixel bar charts of sales transactions shown in Figure 4, for example, are defined by the five-tuple

<product type, ⊥, no. of visits, dollar amount, C>

where C corresponds to different attributes, i.e., number of visits, dollar amount, quantity.

3.2 Formalization of the Problem

The basic idea of pixel bar charts is to produce dense pixel visualizations which are capable of showing large amounts of data on a value by value basis without aggregation. The specific requirements for pixel displays are:

- dense display, i.e., bars are filled completely
- non-overlapping, i.e., no overlap of pixels in the display
- locality, i.e., similar data records are placed close to each other
- ordering, i.e., ordering of data records according to O_x, O_y.

To formalize these requirements we first have to introduce the screen positioning function

$$f : A_1 \times \ldots \times A_k \to Int \times Int,$$

which determines the x-/y-screen positions of each data record d_i, i.e., $f(d_i) = (x, y)$ denotes the position of data record d_i on the screen, and $f(d_i).x$ denotes the x-coordinate and $f(d_i).y$ the y-coordinate. Without loss of generality, we assume that $O_x = A_1$ and $O_y = A_2$. The requirements can then be formalized as:

1. Dense Display Constraint

The dense display constraint requires that all pixel rows (columns) except the last one are completely filled with pixels. For equal-width bar charts, the width w of the bars is fixed. For a partition p consisting of $|p|$ pixels, we have to ensure that

$$\forall i = 1..w, \forall j = 1..\lfloor |p| / w \rfloor : \exists d_i \; with \; f(d_i) = (i, j)$$

For equal-height bar charts of height h the corresponding constraint is

$$\forall i = 1..\lfloor |p| / h \rfloor, \forall j = 1..h : \exists d_i \; with \; f(d_i) = (i, j)$$

2. No -Overlap Constraint

The no-overlap constraint means that a unique position is assigned to each data record. Formally, we have to ensure that two different data records are placed at different positions, i.e.,

$$\forall d_i, d_j \in DB : i \neq j \Rightarrow f(d_i) \neq f(d_j).$$

3. Locality Constraint

In dense pixel displays the locality of pixels plays an important role. Locality means that similar data records are placed close to each other. The partitioning in pixel bar charts ensures a basic similarity of the data records within a single bar. In positioning the pixels within the bars, however, the locality property also has to be ensured. For the formalization, we need a function $sim(d_i, d_j) \longrightarrow [0\ldots1]$ which determines the similarity of two data records and the inverse function of the pixel placement function f^1, which determines the data record for a given (x,y)-position on the screen. The locality constraint can then be expressed as

$$\sum_{x=1}^{w} \sum_{y=1}^{h-1} sim(f^{-1}(x, y), f^{-1}(x, y+1)) +$$
$$\sum_{x=1}^{w-1} \sum_{y=1}^{h} sim(f^{-1}(x, y), f^{-1}(x+1, y)) \to \min$$

Note that in general it is not possible to place all similar pixels close to each other while respecting the dense display and no-overlap constraints. This is the reason why the locality constraint is formalized as a global optimization problem.

4. Ordering Constraint

The last constraint which is closely related to the locality constraint is the ordering constraint. The idea is to enforce a one-dimensional ordering in x- and y-direction according to the specified attributes $O_x = A_1$ and $O_y = A_2$. Formally, we have to ensure

$$\forall i, j \in 1..n : a_1^i > a_1^j \Rightarrow f(d_i).x > f(d_j).x$$

$$\forall i, j \in 1..n : a_2^i > a_2^j \Rightarrow f(d_i).y > f(d_j).y$$

Note that ordering the data records according to the attribute and placing them in a row-by-row or column-by-column fashion may easily fulfill each one of the two constraints. Ensuring both constraints at the same time may be impossible in the general case. We can formalize the constraint as an optimization problem:

[1] The element ⊥ is used if no attribute is specified.

$$\sum_{x=1}^{w}\sum_{y=1}^{h-1} (f^{-1}(x,y).a_1 - f^{-1}(x,y+1).a_1 + \left|f^{-1}(x,y).a_1 - f^{-1}(x,y+1).a_1\right|)/2 + \sum_{x=1}^{w-1}\sum_{y=1}^{h} (f^{-1}(x,y).a_2 - f^{-1}(x+1,y).a_2 + \left|f^{-1}(x,y).a_2 - f^{-1}(x+1,y).a_2\right|)/2 \to \min$$

Note that there may be a trade-off between the x- and the y-ordering constraint. In addition, the optima for the locality and the ordering constraints are in general not identical. This is due to the fact that the similarity function may induce a different optimization criterion than the x-/y-ordering constraint. For solving the pixel placement problem, we therefore have to solve an optimization problem with multiple competing optimization goals. The problem is a typical complex optimization problem which is likely to be NP-complete and can therefore only be solved efficiently by a heuristic algorithm.

3.3 Pixel Placement Algorithm

For the generation of pixel bar charts, we have to

- partition the data set according to D_x and D_y
- determine the pixel color according to C[2]
- place the pixels of each partition in the corresponding regions according to O_x, O_y .

The partitioning according to D_x and D_y and the color mapping are simple and straightforward to implement, and therefore do not need to be described in detail here. The pixel placement within one bar, however, is a difficult optimization problem because it requires a two-dimensional sort. In the following, we describe our heuristic pixel placement algorithm which provides an efficient solution to the problem. The basic idea of the heuristic pixel placement algorithm is to partition the data set into subsets according to O_x and O_y, and use those subsets to place the bottom- and left-most pixels. This provides a good starting point which is the basis for the iterative placement of the remaining pixels. The algorithm works as follows:

1. For an efficient pixel placement within a single bar, we first determine the one-dimensional histograms for O_x and O_y, which are used to determine the α-quantiles of O_x and O_y. If the bar under consideration has extension w x h pixels, we determine the $1/w,\ldots,(w-1)/w$-quantiles for the partitioning of O_x, and the $1/h,\ldots,(h-1)/h$-quantiles for the partitioning of O_y. The quantiles are then used to determine the partitions $X_1, \ldots, X_w$ of O_x and $Y_1, \ldots, Y_h$ of O_y. The partitions $X_1, \ldots, X_w$ are sorted according to O_y and the partitions $Y_1, \ldots, Y_h$ according to O_x.
2. We can start now to place the pixel in the lower-left corner, i.e., position (1,1), of the pixel bar:
$$f^{-1}(1,1) = \left\{ d_s \mid \min_{d_s \in X_1}\{d_s.a_2\} = \min_{d_s \in Y_1}\{d_s.a_1\} \right\}$$
Next we place all pixels in the lower and left pixel rows of the bar. This is done as
$$f^{-1}(i,1) = \left\{ d_s \mid \min_{d_s \in X_i}\{d_s.a_2\} \right\} \quad \forall i = 1..w$$
$$f^{-1}(1,j) = \left\{ d_s \mid \min_{d_s \in Y_j}\{d_s.a_1\} \right\} \quad \forall j = 1..h$$
3. The final step is the iterative placement of all remaining pixels. This is done starting from the lower left to the upper right. If pixels at positions *(i-1, j)* and *(i, j-1)* are already placed, the pixel at position *(i, j)* is determined as
$$f^{-1}(i,j) = \left\{ d_s \mid \min_{d_s \in X_i \cap Y_j}\{d_s.a_1 + d_s.a_2\} \right\} \qquad \text{if } X_i \cap Y_j \neq \varnothing$$
Because we have placed the data structures as introduced in step 1, the pixel to be placed at position can be determined in O(1) time if $X_i \cap Y_j \neq \varnothing$. If $X_i \cap Y_j = \varnothing$, we have to iteratively extend the partitions X_i and Y_j and consider
$$d_s \in (X_i \cup X_{i+1}) \cap Y_j \,.$$
If this set is still empty, we have to consider
$$d_s \in (X_i \cup X_{i+1}) \cap (Y_j \cup Y_{j+1})$$
and so on, until a data point to be placed is found. Note that this procedure is quite efficient due to the data structure used.

4. The Pixel Bar Chart System

To analyze large volumes of transaction data with multiple attributes, pixel bar charts have been integrated with a data mining visualization system [Hao 99]. The system uses a web browser with a Java activator to allow real-time interactive visual data mining on the web. The web interfaces are based on standard HTML and

[2] We use a colormap which maps high data values to bright colors and low data values to dark colors.

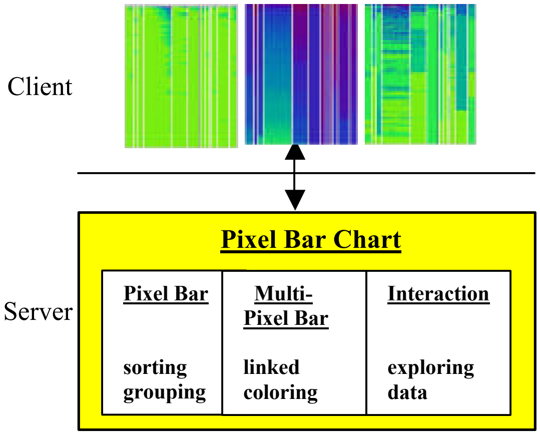


Figure 8: System Architecture & Components

Java applets, which are used to explore relationships and to retrieve data within a region of interest. The server is integrated with the data warehouse and the mining engine. The user at the client side visually explores the data by dynamically accessing the large multi-attribute transactions with complex relationships through HTML pages in a web browser.

4.1 System Architecture and Components

The pixel bar chart system connects to a data warehouse server and uses the database to query for detailed data as needed. The data to build the pixel array is kept in memory to support real-time manipulation and correlation. As illustrated in Figure 8, the pixel bar chart system architecture contains three basic components:

1. Pixel array ordering and grouping
   A pixel array is constructed from the pixel bar chart five tuple specification. One pixel represents one data record, i.e., a customer. The partitioning algorithm assigns each data record to the corresponding bar according to the partitioning attribute(s). The pixel placement implements a simplified version of the heuristic algorithm presented in subsection 3.4.

2. Multiple linked pixel bars
   In multi-bar charts, the position of the pixels belonging to the same data record remains the same across multi-pixel bar charts for correlation. The colors of the pixel correspond to the value of the selected attributes (such as price, number of orders, etc.).

3. Interactive data exploration
   This system provides simultaneous browsing and navigation of multiple attributes.

4.2 Interactive Data Analysis

Interactivity is an important aspect of the pixel bar chart system. To make large volumes of multi-attribute data sets easy to explore and interpret, the pixel bar chart system provides the following interaction capabilities:

- visual querying
- layered drill-down / detail-on-demand
- multiple linked visualizations
- zoom in and out of the pixel bar charts

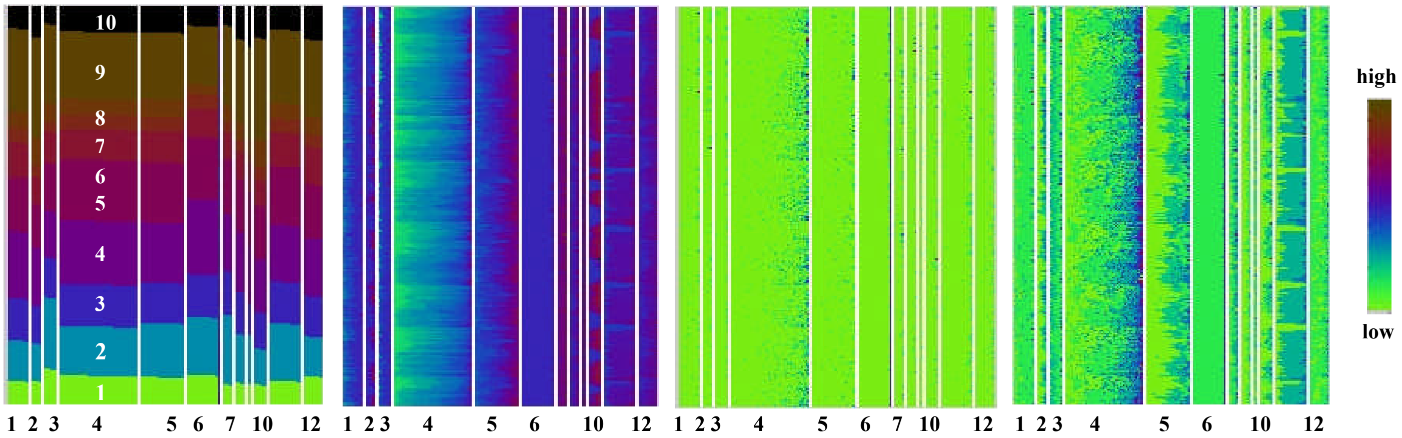
The attributes used for partitioning *(Dx, Dy)*, ordering *(Ox, Oy)*, and coloring *(C)* can be selected and changed at execution time. For identifying correlations, a subset of data items in a pixel bar chart can be selected to get the pixels corresponding to related attribute values highlighted within the same display. A drill-down technique allows the viewing of all related information after selecting a single data item. When multi-bar charts are presented, pixels reside at the same location across all the charts with different attributes. In addition to discovering correlations and patterns, the user can select a single data item to relate all its attribute values.

5. Application and Evaluation

The pixel bar chart technique has been prototyped in several e-commerce applications at Hewlett Packard Laboratories. It has been used to visually mine large volumes of sales transactions and customer shopping activities at HP shopping web sites.

5.1 Customer Analysis

The pixel bar chart system has been applied to customer buying patterns and behaviors. In Figure 9, the pixels of the bar chart represent customers making transactions on the web. In the resulting pixel bar chart, customers with similar purchasing behaviors (i.e., product type, geographical location, dollar amount, number of visits, and quantity) are placed close to each other. A store manager can use the visualization to rapidly discover customer buying patterns and use those patterns to target marketing campaigns. Figure 9 shows the four attributes of 106,199 customer buying records. The four pixel bar charts of Figure 9 are constructed as follows:



a) Color: Region b) Color: Dollar amount c) Color: No. of Visits d) Color: Quantity

Figure 9: Multi-Pixel Bar Chart for Mining 106,199 Customer Buying Transactions
(D_x= Product Type, D_y=$\perp$, O_x=dollar amount, O_y=region, C)

- *Product type* is the dividing attribute D_x
  12 product types
- *Dollar amount* is the x-ordering attribute O_x
  Region is the y-ordering attribute O_y
  for 10 United States regions
- *Region, dollar amount, number of visits,* and *quantity* are the four coloring attributes *C*

Many important facts may be discovered in Figure 9 (a, b, c, d). In the bars for the different attributes, the user may observe the following facts:

a) Region attribute
There are 10 different colors to represent 10 different regions (labeled 1-10 in Figure 9a) in the United States. The colored wave indicates the number of customers in each region. After analyzing customer distributions, region 9 (largest area) is found to have the largest number of customers. Region 7 (smallest area) has the least number of customers across all product types.

b) Dollar amount attribute
Product type 5 has the most top dollar amount sales (blue & brown). The dollar amount sales of product types 6 and 7 have a very small variance across all regions (solid blue/brown).

c) Number of visits attribute
The blue color distribution in product type 4 indicates that customers of this product type (consumables) come back more often than customers of other product types.

d) Quantity attribute
The green color of product type 6 indicates that in this category all customers bought the same number of items across all regions. It is also obvious that product type 4 customers have the largest quantities.

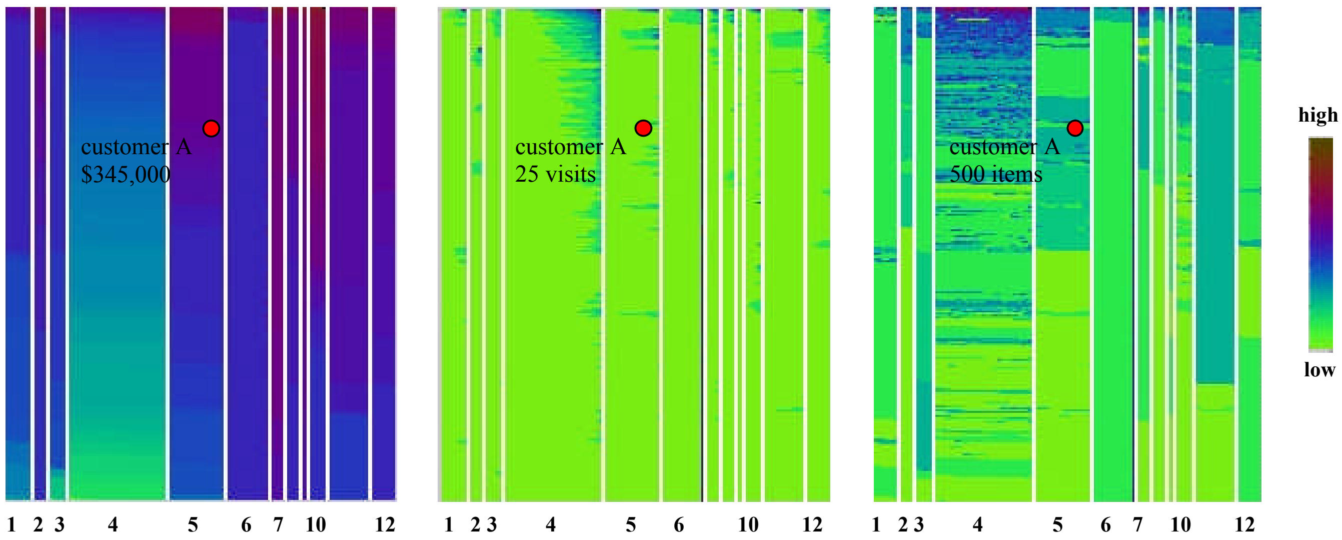
By relating the different bar charts of the multi bar chart of Figure 9, the user may observe for example the following clusters and trends:

- Region 4 has the most customers but region 9 is the most profitable with the most frequent visits and the largest quantities.
- The top dollar amount customers come back more frequently and purchase larger quantities.

5.2 Sales Transaction Analysis

One of the common questions electronic store managers ask is how to use the customer purchase history for improving product sales and promotion. Product managers want to understand which products have the top sales and who are their top dollar amount customers.

An e-commerce manager, for example, needs to answer questions as to which product types have the highest dollar amount customers, how often the customers come back and for which products. These analyses may also be used to determine which products may be impacted when the store issues coupons.



a) Color: dollar amount

b) Color: no. of visits

c) Color: quantity

Figure 10: Multi-Pixel Bar Chart for Mining 405,000 Sales Transaction Records
(D_x= Product Type, D_y=$\perp$, O_x=no. of visits, O_y= dollar amount, C)

While regular bar charts provide only aggregated information on the number of customers by product type (Figure 1), the corresponding pixel bar charts include important additional information such as the dollar amount distribution of the sales. More specifically, the pixel bar chart provides the following additional information:

- Dollar amount versus product distribution
- Each customer's detail information can be drilled down as needed.

Figure 10 illustrates an example of a multi-pixel bar chart of 405,000 multi-attribute web sales transactions. The dividing attribute *(D_x)* is again product type; the ordering attributes are number of visits and dollar amount *(O_x and O_y)*. The colors *(C)* in the different bar charts represent the attributes dollar amount, number of visits, and quantity. From Figure 10, the following information about the web sales can be obtained:

a) Product type 10 and product type 7 have the top dollar amount customers (dark colors of bar 7 and 10 in Figure 10a).

b) The dollar amount spent and the number of visits are clearly correlated, especially for product type 4 (linear increase of dark colors at the top of bar 4 in Figure 10b).

c) Product types 4 and 11 have the highest quantities sold (dark colors of bar 4 and 11 in Figure 10c).

d) By clicking on a specific pixel (A), we may find out that customer A visited 25 times, bought 500 items, and spent $345,000 on product type 5.

It is further interesting that there are clusters of darker colors in bar 4 of Figure 10c, which means that there are certain ranges of dollar amount sales for which the quantity tends to be higher than in other segments. This observation is unexpected and may be used to identify the clusters of sales transactions and make use of the information to further increase the sales. Note that the information mentioned above cannot be detected by regular bar charts.

6. Conclusion

In this paper, we presented pixel bar charts, a new method for visualizing large amounts of multi-attribute data. The approach is a generalization of traditional bar charts and x-y diagrams, which avoids the problem of losing information by aggregation or overplotting. Instead, pixel bar charts map each data point to one pixel of the display. For generating the pixel bar chart visualizations, we have to solve a complex optimization problem. The pixel placement algorithm is an efficient and effective solution to the problem. We apply the pixel bar chart idea to real data sets from an e-commerce application and show that pixel bar charts provide significantly more information than regular bar charts.

Acknowledgements:

Thanks to Sharon Beach of HP Laboratories for her encouragement and suggestions, Shu Feng Wei and Brain Ono from HP Shopping for providing data and reviewing the results, and to Graham Pollock of Agilent Laboratories for his review and comments.

References

[ADLP 95] Anupam V., Dar S., Leibfried T., Petajan E.: *'DataSpace: 3-D Visualization of Large Databases',* Proc. Int. Symp. on Information Visualization, Atlanta, GA, 1995, pp. 82-88.

[AKK 96] Ankers M., Keim D. A., Kriegel H.P.: *'Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets',* VISUALIZATION '96, HOT TOPIC SESSION, San Francisco, CA, 1996.

[AWS 92] Ahlberg C., Williamson C., Shneiderman B.: *'Dynamic Queries for Information Exploration: An Implementation and Evaluation',* Proc. ACM CHI Int. Conf. on Human Factors in Computing, Monterey, CA, 1992, pp. 619-626.

[Bed 90] Beddow J.: *'Shape Coding of Multidimensional Data on a Mircocomputer Display',* Proc. Visualization '90, San Francisco, CA, 1990, pp. 238-246.

[BEW 95] Becker R. A., Eick S. G., Wills G. J.: *'Visualizing Network Data',* IEEE Transactions on Visualizations and Graphics, Vol. 1, No. 1, 1995, pp. 16-28.

[BMMS 91] Buja A., McDonald J. A., Michalak J., Stuetzle W.: *'Interactive Data Visualization Using Focusing and Linking'*, Proc. Visualization '91, San Diego, CA, 1991, pp. 156-163.

[Eic 99] Stephen G. Eick: *Visualizing Multi-dimensional Data with ADVISOR/2000*, Visualinsights, 1999.

[EW 93] Eick S., Wills G. J.: *'Navigating Large Networks with Hierarchies'*, Proc. Visualization '93, San Jose, CA, 1993, pp. 204-210.

[Hao 99] Hao Ming, Dayal Umeshwar, Hsu Meichun, D'eletto Bob, Becker Jim, *'A Java-based Visual Mining Infrastructure and Applications'*, IEEE InfoVis99, San Francisco, CA. 1999.

[ID 90] Inselberg A., Dimsdale B.: *'Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry'*, Proc. Visualization '90, San Francisco, CA, 1990, pp. 361-370.

[Ins 85] Inselberg A.: *'The Plane with Parallel Coordinates, Special Issue on Computational Geometry',* The Visual Computer, Vol. 1, 1985, pp. 69-97.

[KK 94] Keim D. A., Kriegel H. P.:*'VisDB: Database Exploration using Multidimensional Visualization'*, Computer Graphics & Applications, Sept. 1994, pp. 40-49.

[KKA 95] Keim D. A., Kriegel H. P., Ankerst M.: *'Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data',* Proc. Visualization '95, Atlanta, GA, 1995, pp. 279-286.

[LWW 90] LeBlanc J., Ward M. O., Wittels N.: *'Exploring N-Dimensional Databases', Proc.* Visualization '90, San Francisco, CA, 1990, pp. 230-237.

[LRP 95] Lamping J., Rao R., Pirolli P.: *'A Focus + Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies',* Proc. ACM CHI Conf. on Human Factors in Computing (CHI95), 1995, pp. 401-408.

[PG 88] Pickett R. M., Grinstein G. G.: *'Iconographic Displays for Visualizing Multidimensional Data',* Proc. IEEE Conf. on Systems, Man and Cybernetics, IEEE Press, Piscataway, NJ, 1988, pp. 514-519.

[RCM 91] Robertson G., Card S., Mackinlay J.: *'Cone Trees: Animated 3D Visualizations of Hierarchical Information',* Proc. ACM CHI Int. Conf. on Human Factors in Computing, 1991, pp. 189-194.

[SB 94] Sarkar M., Brown M.: *'Graphical Fisheye Views',* Communications of the ACM, Vol. 37, No. 12, 1994, pp. 73-84.

[Shn 92] Shneiderman B.: *'Tree Visualization with Treemaps: A 2D Space-Filling Approach',* ACM Transactions on Graphics, Vol. 11, No. 1, 1992, pp. 92-99.