First publ. in: Visual database systems 3 (VDB 3), visual information management, proceedings of the Third IFIP 2.6 Working Conference on Visual Database Systems, 1995, held in Lausanne, Switzerland, in March 1995 / ed. by Stefano Spaccapietra ... London : Chapman & Hall, 1995, pp. 203-214

Issues in Visualizing Large Databases

D. A. Keim, H.-P. Kriegel
Institute for Computer Science, University of Munich
Leopoldstr. 11 B, D-80802 Munich, Germany,
phone: (+49) 89-2180-6267 (-5268), fax: (+49) 89-2180-5246,
e-mail: {keim, kriegel}@informatik.uni-muenchen.de

Abstract
In this paper, we describe our concepts to visualize very large amounts of data. Our visualization techniques which have been developed to support querying large scientific databases are designed to visualize as many data items as possible on current display devices. Even if we are able to use each pixel of the display device to visualize one data value, the number of data items that can be visualized is quite limited. Therefore, in our system we consider only those data items which are 'close' to a user-specified query. The data items are arranged according to their distance from the query. Multiple windows are used for the different attributes of the data and the distance of each of the attributes from the query is represented by color. After briefly describing the basic ideas of our visualization techniques, in this paper we focus on two specific issues which arise in connection with our visualization techniques: First, we compare the *data space* which is the amount of information available in the database to the *visualization space* which is the amount of information that can be visualized by our visualization techniques. Second, we discuss the usefulness of different color models for our distance-to-color mapping and define a new color model which produces adequate color scales.

Keywords
Visualizing Large Databases, Database Exploration, Data Mining, Visual Query Systems, Interfaces to Database Systems, Data Space and Visualization Space, Color Models

1 INTRODUCTION

The progress made in hardware technology allows today's computer systems to store very large amounts of data. The available storage space is easily filled with data which is often automatically recorded via sensors and monitoring systems. Today, even simple transactions of every day life, such as paying by credit card or using the telephone, are typically recorded by using computers. Even larger amounts of data are generated by automated test series in physics, chemistry or medicine, and satellite observation systems are expected to collect one terabyte of data every day in the near future (Frawley, Piatetsky-Shapiro, Matheus 1991). Usually, many parameters are recorded, resulting in multidimensional data with a high dimensionality. The data of all areas mentioned so far is collected because people believe that it is a potential source of valuable information providing a competitive advantage (at some point).

Finding the valuable information hidden in them, however, is a difficult task. With today's database systems and their query tools, it is only possible to view quite small portions of the data. If the data is presented textually, the amount of data which can be displayed is in the range of some one hundred data items, but this is like a drop in the ocean when dealing with data sets containing millions of data items. Having no possibility to adequately query and view the large amounts of data which have been collected because of their potential usefulness, the data becomes useless and the database becomes a data 'dump'.

For the exploration of very large databases to be successful in the near future, we believe that it is essential to make the human being an integral part of the data analysis process. It will be important to combine the best features of humans and computers. The intelligence, creativity and perceptual abilities of humans which are unmatchable need to be supported by computers which are best suited to do searching and number crunching. Some five years ago, a broader community of researchers recognized the potentials of visualization techniques to analyze and explore large amounts of data. With visualization techniques, larger amounts of data can be presented on the screen at the same time, colors allow the users to instantly recognize similarities or differences of thousands of data items, the data items may be arranged to express some relationship and so on. Over the last years, many techniques for the visualization of multidimensional data have been developed. It seems, however, that many of the techniques do not provide adequate support for the flood of data we are facing today. Since, on the other hand, the technology for generating, collecting and storing data is available, the gap between the amount of data that should be visualized and the amount of data that can be visualized is growing. Therefore, a major research challenge is to find human-oriented ways to help the user in exploring large amounts of multidimensional data.

Our work focuses on visualization techniques which use color and dense displays to visualize large multidimensional data sets. Our techniques support querying large databases by query-dependent visualizations of the data. In (Keim, Kriegel, Seidl 1994), the interactive query and visualization interface of a first prototype system is presented. Today, a complete re-implementation of the systems in C++ / MOTIF exists which runs on HP 7xx machines under X-windows. In addition to the basic technique presented in (Keim, Kriegel, Seidl 1994), the current version of the *VisDB* system implements several extensions and new visualization techniques which are briefly introduced in section 2 (see Keim 1994 for a detailed discussion). In this paper, we focus on two issues which arise in connection with our visualization techniques. In section 3, we compare the *data space* which is the amount of information available in the database to the *visualization space* which is the amount of information that can be visualized by our visualization techniques. In section 4, we discuss the usefulness of different color models for our distance-to-color mapping and define a new color model which produces adequate color scales. Section 5 summarizes our approach and points out some of the open problems for future work.

2 THE BASIC IDEA OF OUR VISUALIZATION TECHNIQUE

Visualization of data which have some inherent two- or three-dimensional semantics has been done even before computers could be used for visualization, and since using computers for this purpose, a lot of interesting visualization techniques have been developed by researchers working in the graphics field. Visualization of large databases which can be seen as arbitrary multidimensional data, however, is a relatively new research area. Researchers in the graphics/

visualization area are currently exploring techniques in different application domains. Examples are shape coding (Beddow 1990), worlds within worlds (Feiner, Beshers 1990), parallel coordinates (Inselberg, Dimsdale 1990), treemaps (Turo, Johnson 1992) iconic displays (Pickett, Grienstein 1988; Bergeron, Meeker, Sparr 1992), or dynamic methods as presented in (Buja et al. 1991) or (Shneiderman 1994). In most of the approaches proposed so far, the number of data items that can be visualized on the screen at the same time is quite limited (in the range of 100 to 1000 data items), but it is a declared goal to push this limit (Treinish et al. 1992). In dealing with databases consisting of millions or even billions of data items, our goal is to visualize as many data items as possible at the same time to give the user some kind of overview of the data. The obvious limit for any kind of visual representation is the resolution of current displays which is in the order of one to three million pixels, e.g. in case of our 19 inch displays with a resolution of 1024 x 1280 pixels there are about 1.3 million pixels. The basic idea of our visualization techniques is to use each pixel of the screen to give the users visual feedback about the data, allowing them to easily focus on the desired data and to understand the influence of the query parameters.

In dealing with databases consisting of millions of data items with multiple attributes (often ten and more), we had to find an adequate way of restricting the amount of data to be visualized to a number that can be displayed on the screen. In our approach, we only visualize the data items which are ‘close’ to a user-specified query. The ‘closeness’ is determined by using distance functions for each of the attributes. The distance functions are datatype and application dependent. Examples for distance functions are the numerical difference (for metric types), distance matrices (for ordinal and nominal types), lexicographical, character-wise, substring, phonetic or semantic difference (for strings) and so on. In the specification of the query, not all of the attributes have to be used. If k of the m attributes are used in the specification of the query, then the query region itself can be seen as an (m-k)-dimensional space with some extension into the other m dimensions. Attributes that are not used in the specification of the query region have basically no impact on the visualization.

Having calculated the distances for each of the attributes which are part of the query, the distances are combined into the overall distance. Important aspects such as normalizing and weighting the distances of the different attributes, the formulas used to calculate the overall distances, and the heuristics used to reduce the number of displayed data items are described in (Keim, Kriegel, Seidl 1994). The overall distances are then sorted, resulting in a one-dimensional distribution ranking the data items according to their distance with respect to the query. The basic idea for visualizing the data items is to map the value ranges of the different attributes to color and represent each data item by multiple pixels being colored according to the distance values for each of its attributes. In the following, we briefly introduce the three visualization techniques which may be used for presenting the colored pixels on the screen. A more detailed description of the techniques including several examples is given in (Keim 1994; Keim, Kriegel 1995).

Spiral Technique

The basic idea for visually displaying the data on the screen is to present the one hundred percent correct answers in the middle of the window and the approximate answers rectangular spiral-shaped around this region. To relate the visualization of the overall result to visualizations of the different selection predicates (dimensions), we generate a separate window for each selection predicate of the query and arrange them next to each other. In the separate windows, we place the pixels for each data item at the same relative position as the overall result for the data item in the overall result window. By relating corresponding regions in the different windows, the user is able to perceive data characteristics such as multidimensional clusters or correlations. Additionally, the separate windows for each of the selection predicates provide important feedback to the user, e.g. on the restrictiveness of each of the selection predicates and on single exceptional data items. An example visualization of about 70 000 three-dimensional test data items is provided in Figure 6.

Axes Technique

The axes arrangement improves the spiral technique by including some feedback on the direction of the distance into the visualization. The basic idea is to assign two dimensions to the axes and to arrange the data items according to the direction of their distance; for one dimension negative distances are arranged to the left, positive ones to the right and for the other dimension negative distances are arranged to the bottom, positive ones to the top. The partitioning of the data into four subsets provides additional information on the position of data items with respect to the axes dimensions. Since the quadrants which correspond to the four subsets are not equally filled, the number of data items which may be visualized is slightly lower. In the example visualization provided in Figure 7, the same data set is used as in Figure 6.

Grouping Technique

In the grouping arrangement all dimensions for one data item are grouped together in one area. The areas are arranged in a rectangular spiral-shape according to the overall distance of the considered data items. The coloring of the distances for the different dimensions may be the same as in the spiral and axes arrangements. The generated visualizations, however, are completely different from those of the spiral and axes arrangements. The visualizations generated by using the grouping arrangement consist of only one window with many areas visualizing all dimensions of the considered data items instead of many windows, each providing a visual representation of only one dimension. An example visualization of about 1000 six-dimensional data items is provided in Figure 8.

In trying to visualize larger amounts of data than possible with the techniques described so far, an important potential is to consider time as an additional dimension. For many applications it is natural to consider time sequences of visualizations, describing features which are changing over time. In the terminology of our system, this could be described as moving the query region along the time dimension. Most traditional systems for visualizing time series consider in each step only the data items at a certain point of time. Contrarily, with our visualization techniques we consider all data items which are ‘close’ to the query region including data items with differing time values as long as their overall distance with respect to the query is low enough. Our idea for visualizing larger portions of the database is to generalize the technique of generating sequences of visualizations by moving the query region. Instead of moving the query region along the time axis, the user may choose an arbitrary path through m-dimensional space. Obviously, the semantics of the derived visualizations are different. If moving the query region along some parameter, e.g. the temperature in an environmental database, the user may get insight in the corresponding distributions of the other parameters such as ozone or CO_2. The user may also choose more complicated paths through m-dimensional space, e.g. by varying two parameters such as temperature and ozone at a time. The specification of more complicated paths, however, is not straightforward and it is not clear how the user will be able to deal with the complicated semantics introduced by complex paths. An open question is which paths provide visualizations that are 'easy' to perceive and allow the user to find interesting properties of the data. Despite these open questions, generating time series of visualization holds a great potential since neither the number of data items that can be visualized nor their dimensionality is limited, and the visualizations may help the user to get important information out of the automatically generated visualization sequences.

3 DATA SPACE VERSUS VISUALIZATION SPACE

In trying to visualize very large databases, it is interesting to compare the amount of information available in the database to the amount of information that can be visualized by using current display technology. A table of a (relational) database can be seen as a set $\{e_1, ..., e_n\}$ of m-dimensional data items $[e_i = (d_{i1}, ..., d_{im})]$. The portion of m-dimensional space which is (partially) filled with data from the database is the smallest orthogonal subspace that contains all data items of the database, which can be formally expressed as

$$\prod_{j=1}^{m} \left[\operatorname*{MIN}_{i=1}^{n} \{d_{ij}\} \ , \ \operatorname*{MAX}_{i=1}^{n} \{d_{ij}\} \right] .$$

If all of the domains are simple discrete domains such as integer, the size of the considered m-dimensional space may be determined by calculating the differences between the minima and maxima for each of the dimensions and computing their cross-product. In general, the size of the considered m-dimensional space may be determined by using the cardinality of each dimension, which is given by the number of possible values between its minimum and maximum. For continuous data types such as real, the cardinality can only be determined by making the domain discrete. The number of distinguishable values depends on the precision of the datatype, which itself depends on number of bits used to represent each data value.

To define our general notion of data space, we use Dom_j for the domain of dimension j and $|Dom_j|$ for the cardinality of the domain (j = 1 ... m). Analogous to the term visualization space, which denotes all visualizations, the term data space is used to denote all possible instances of databases with m dimensions and n elements. The data space can be defined as the number of all power sets with n elements in m-dimensional space:

$$Data\ Space = \left(\prod_{j=1}^{m} |Dom_j| \right)^n$$

The visualization space may be defined as the number of different visual representations that are possible on the screen. If 'Col' is the number of distinguishable colors and (X * Y) the size of the screen used, then the visualization space may be defined as:

$$Visualization\ Space = Col^{(X \times Y)}$$

Note that the number of visual representations that are pre-attentively perceptible as being different is much lower. If two visual representations which differ in just a few pixels are compared to each other, one would generally find it very difficult to find the differences, especially if the differing pixels have similar colors. Additionally, the anatomy and physiology of the human vision system restricts the number of distinguishable visual representations. Effects such as accommodation, blind spots, afterimages, lateral inhibition, and mach bands (regions near sharp transitions that are perceived brighter or darker) prevent the human vision system from perceiving visualizations as being different (Hendee, Wells 1993). Still, the remaining visualization space is very large and no visualization technique completely uses it. Visualization techniques usually have specific arrangements and colorings of the pixels which restrict the number of possible visual representations considerably. Later in this section, we derive a formula for the visualization space with regard to our visualization techniques.

Now let us compare data and visualization space. To simplify the formula for the data space, we assume that the cardinality of all domains of the data space is constant (dom). The data space can then be defined as

$$Data\ Space' = dom^{(m \times n)} .$$

Now there is an apparent similarity of the formulas for the data and visualization space. Since in general (Col << dom) and ((X*Y) << (m*n)), it is clear that any mapping of the data into the visualization space requires some kind of information reduction. To achieve a good utilization of the visualization space, it is desirable to find a surjective mapping. One possibility for a surjective mapping from the data space into the visualization space is to map the domains to colors [dom $\rightarrow$ Col] and the number of data values in the database to the number of pixels [(m*n) $\rightarrow$ (X*Y)]. In trying to use as much of the visualization space as possible, our visualization techniques use exactly this kind of mapping. The mapping of domains to colors corresponds to the normalization of data values to a fixed number of values; the mapping of the number of data values to the number of pixels corresponds to the reduction of the number of data items to those which are close to the query region.

Now let us have a closer look at the visualization space for our visualization techniques. Since certain restrictions apply in our visualizations, not all of the $Col^{(X \times Y)}$ visual representations are possible. One restriction is that in our visualizations we have a portion for the overall result representing the overall distance sorted in rising order. The sorting of this portion restricts the number of possible visual representations. The restriction can be described by the following recursive equation

$$f(Col, p) = \sum_{i=1}^{Col} f(i, p-1) \qquad \text{with} \qquad \begin{array}{l} f(Col, 1) = Col \\ f(1, p) = 1 \end{array} .$$

A numerical solution for this recurrence equation is the binomial coefficient $\binom{Col + p - 1}{p}$ which can be proved by induction over (Col + p). A second restriction is that our visualization techniques use squares as sections for each of the dimensions. In general, partitioning a screen with fixed resolution $(X \times Y)$ into a fixed number of squares (m + 1) does not allow the usage of all pixels on the screen. Depending on the number of dimensions (m), the size of the squares is limited either by the X- or the Y-dimension of the screen. If the squares are assumed to be completely filled, the number of pixels in each of the squares [P(m)] can be determined as

$$P(m) = \begin{cases} \left(\frac{X}{i}\right)^2 & \text{for m-1} \in [i^2 - i + 1, i^2] \\ \left(\frac{Y}{i}\right)^2 & \text{for m-1} \in [i^2 - 2 \times i + 2, i^2 - i] \end{cases} .$$

$$\boxed{\underbrace{\left(1-\frac{1}{Col^{m+1}}\right)^{(P(m)-H)}}_{\substack{\text{no yellow in all}\\ \text{sections if not a hit}}} \times \underbrace{\binom{Col+P(m)-H-1}{P(m)-H}}_{\substack{\text{overall result section}\\ \text{sorted according to color}}} \times \underbrace{Col^{(m\times(P(m)-H))}}_{\substack{\text{arbitrary color of all remaining}\\ \text{pixels which are no hits}}}}$$

Figure 1 Visualization Space (Spiral Technique).

A third restriction is induced by the yellow pixels that are used to denote data items inside the query region. These pixels are yellow in all m portions of the visualization. Therefore, the number of pixels that can be arbitrarily colored has to be reduced by this number (number of data items fulfilling the query: H). A last restriction is that pixels that are at the same relative position in each of the sections may not be yellow unless they are hits. We therefore have to restrict the number of possible visualizations by a factor of $\left(1-\frac{1}{Col^{m+1}}\right)^{(P(m)-H)}$.

The complete formula for the number of possible visualizations in the case of the spiral technique is given in Figure 1. For the axes and grouping techniques, the arrangement and coloring of pixels is similar and therefore the corresponding formulas are analogous. The major difference in the case of the axes technique is that, in general, not all of the quadrants are completely filled. In the case of the grouping technique, the number of pixels that are used per data item is four (2 x 2) or nine (3 x 3), and therefore the visualization space is correspondingly lower.

At this point, it should be mentioned that in the above definition of data space, two databases which contain the same data items but store them in a different order are considered to be different. Although this may be true in some cases, databases are generally considered to be sets and therefore two databases with identical data items are considered to be the same, independent of the ordering of their elements. This would change the definition of data space to

$$Data\ Space'' = \binom{dom^m+n-1}{n} = \frac{(dom^m+n-1)!}{n!\times(dom^m-1)!} .$$

4 THE COLOR MODEL

Visualizing the data values using color corresponds to the task of mapping a single parameter distribution to color. The advantage of color over grey scales is that the number of just noticeable differences (JNDs) is much higher (Herman, Levkowitz 1992). Finding a path through color space that maximizes the number of JNDs but, at the same time, is intuitive for the application domain is a difficult task. For our purpose of mapping the distances to color, we can restrict the task to a simpler, more solvable problem. From a perceptual point of view, brightness is the most important characteristic for distinguishing colors corresponding to a single parameter distribution. Therefore, for our purpose it is sufficient to find a color scale with a monotonically increasing (decreasing) brightness while using the full color range.

In designing the system, we experimented with different colormaps. We found that the coloration has a high impact on the intuitivity of the system. The user, for example, may implicitly connect good answers with light colors and bad answers with dark colors, or may be accus-

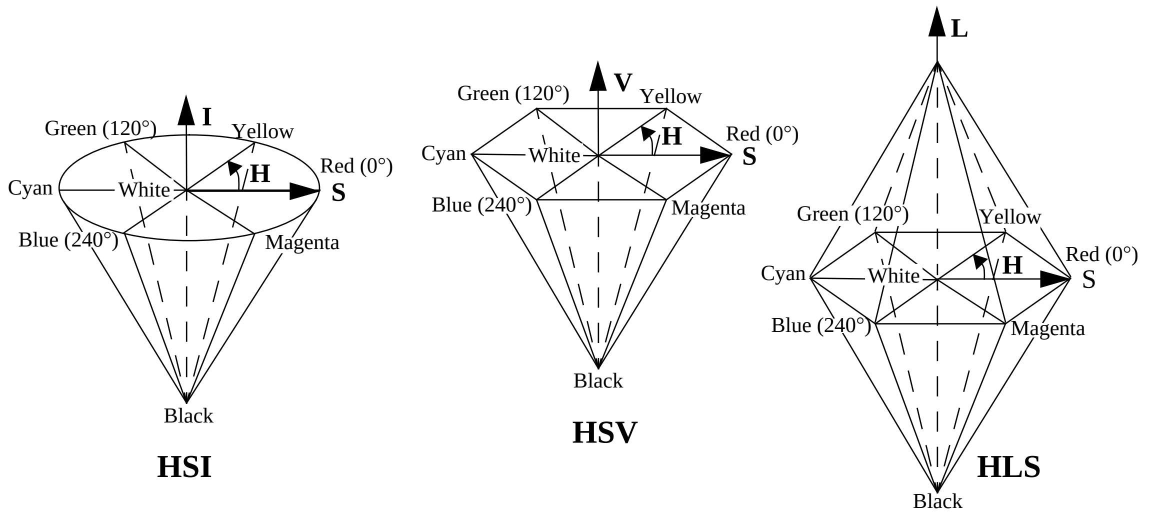

Figure 2 The HSI Color Model compared to the HSV and HLS models.

tomed to green colors for good answers and red colors for bad answers (like the colors used for traffic lights). We tried many variations of the colormap to enhance the usefulness of our system and found experimentally that a colormap with constant saturation, a decreasing value (intensity, lightness), and a hue (color) ranging from yellow over green, blue, and red to almost black are good choices to denote the distance from the correct answers.

For generating color scales, we used a linear interpolation between a minimum and a maximum value for hue, saturation, and value (intensity, lightness) within the various color models. We first used the standard HSV and HLS color models (Foley et al. 1990). The HSV color model is best described by a single-hexcone and the HLS model by a double-hexcone (see Figure 2). Linear interpolations within these color models, however, do not provide color scales with a monotonically decreasing brightness. Figure 5 shows HSV and HLS color scales which are produced using a linear interpolation algorithm similar to the one presented in the bottom part of Figure 4. The color scales are generated using a constant saturation, a decreasing value (intensity, lightness), and a hue varying over the complete range. If the generated HSV and HLS color scales are used in conjunction with our visualization techniques, brighter colors in the visualizations do not necessarily denote lower distances. If HSV and HLS color scales are mapped to grey-scale, their non-monotonicity becomes obvious (see Figure 5)[*].

Since linear interpolation within the HSV and HLS color models does not produce color scales with monotonically decreasing brightness, we developed our own color model which we call the HSI model (H: Hue, S: Saturation, I: Intensity). The HSI color model is a variation of the HSV model. In contrast to color scales generated according to the HSV model, linear interpolation within the HSI model provides color scales whose lightness ranges continuously from light to dark (see Figure 5). This is achieved by using a circular cone instead of the hexcone used in the HSV model (see Figure 2), which means that colors with constant intensity and saturation form a circle. The hue is defined as the angle α between red and the chosen color. When moving on a circle of the HSI color cone, the red, green, and blue portion of the color

[*] The standard mapping from color to grey-scale used by the X-windows system is a linear combination of the RGB-values [Grey(r,g,b) = 0.34*r + 0.5*g + 0.16*b] which tries to reflect the perceived brightness. Note that the X-windows mapping to grey does not correspond to the grey portion in either of the color models, which is defined as the corresponding portion of the grey axis (center of the color cones).

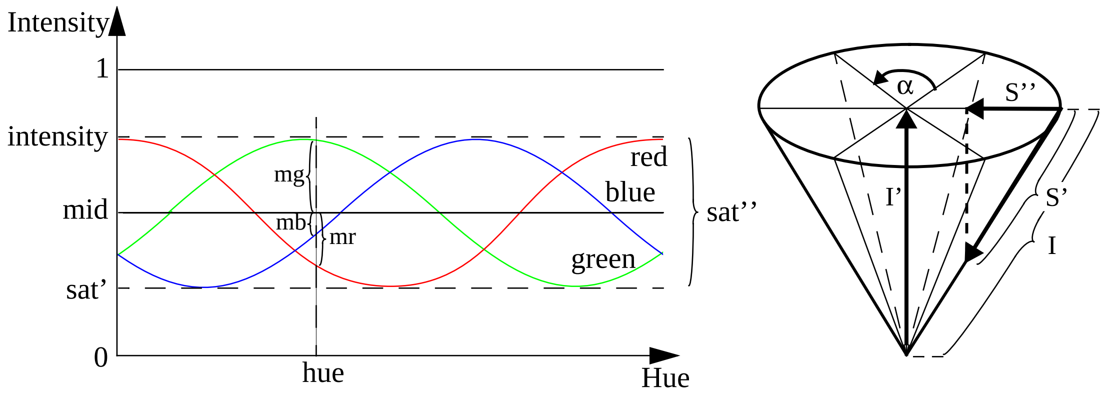

Figure 3 The HSI color model.

follow cosine-curves with phase-shifts of $\frac{2}{3}\pi$ (see Figure 3). If intensity and saturation are both equal to 1, the cosine-curves for each of the RGB-color-portions run between 0 and 1. Reducing the intensity means reducing the maximum of the cosine-curve. Reducing the saturation means lifting the minimum of the cosine-curve (sat'). In terms of the HSI color cone, the intensity is defined as the euclidean distance to the origin and the saturation as percentage of the intensity $\left(\frac{S'}{I}\right)$. In the HSV model, intensity and saturation are determined by using the maximum and minimum of (red, green, blue). In Figure 3, I and S' are marked in the HSI color cone. Since I is proportional to I' and S' proportional to S'', the HSI color cone can also be described with I' and S'' as defining axes as done in Figure 2.

The exact mathematical definition of the HSI parameters (hue, saturation, intensity) in terms of the RGB color components is rather complicated. Therefore, in the following we briefly describe the basic ideas. The intensity can be determined directly from the (red, green, blue)-components. Since the cosine-curves for red, green, and blue have phase-shifts of $\frac{2}{3}\pi$, the square sum of (mr, mg, mb) is constant and proportional to (intensity - mid)^2. The proportionality constant $\left(\frac{3}{2}\right)$ may be determined by using the special case: saturation = 1.

$$(mr^2 + mg^2 + mb^2) = const \quad \Rightarrow \quad (mr^2 + mg^2 + mb^2) = \frac{3}{2} \times (intensity - mid)^2$$

$$\Rightarrow \quad \boldsymbol{intensity} = mid + \sqrt{\frac{2}{3} \times (mr^2 + mg^2 + mb^2)}$$

To determine the saturation, we have to consider the lower limit of the cosine-curves (c.f. sat' in Figure 3). Sat' only yields values between 0 and intensity. To allow values between 0 and 1, sat' has to be normalized $\left(\frac{sat'}{intensity}\right)$. Increasing this value causes a shrinking of the amplitude of the cosine-curves which means that the equal portion of red, green, and blue and therefore the white-portion of the color increases. This is the inverse of the normal use of the term saturation. The saturation is therefore defined as

$$\boldsymbol{saturation} = 1 - \frac{sat'}{intensity} = \frac{sat''}{intensity} = \frac{2 \times (intensity - mid)}{intensity}.$$

The hue is determined by using the scalar product between the vector from the grey axis to red and the vector from the grey axis to the color-point. The point on the grey axis corresponding to a color point (red, green, blue) is calculated as $\frac{red + green + blue}{3}$. The vector from the grey axis $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ to red (1, 0, 0) results in $\left(\frac{2}{3}, -\frac{1}{3}, -\frac{1}{3}\right)$, which is equivalent to (2, -1, -1) when used in the

```
. . .
/*****************************  Constructor   **********************************/
HSI :: HSI( double h, double s, double i ): hue( h<6 ? h : h-6 ), saturation(s), intensity(i)  { }

/**************************  Conversion from RGB to HSI  ***********************/
HSI :: HSI( const RGB& c ){
    double red = c.red, green = c.green, blue = c.blue;
    double mid = (red + green + blue) / 3.0;
    double mr  = red - mid; double mg  = green - mid; double mb  = blue - mid;
    double cos_hue = (2.0 * mr - mg - mb) / sqrt((mr*mr + mg*mg + mb*mb) * 6.0);
    intensity = mid + sqrt (2.0 * (mr*mr + mg*mg + mb*mb) / 3.0);
    saturation = 2.0 * (intensity - mid) / intensity;
    hue = acos( cos_hue ) * 3.0 / M_PI;
    if (blue > green) hue = 6.0 - hue;
}

/**************************  Conversion from HSI to RGB   ***********************/
HSI :: operator RGB(){
    if( saturation == 0.0 )  return RGB( intensity, intensity, intensity );
    double red   = value( hue + 0.0 );          //   0 Grad =   0 pi
    double green = value( hue + 4.0 );          // 240 Grad = 4/3 pi
    double blue  = value( hue + 2.0 );          // 120 Grad = 2/3 pi
    return RGB( red, green, blue );
}

double HSI :: value( double hue_phase ){
    double pure = 0.5 * (1 + cos( hue_phase * M_PI / 3.0 ) );
    return( intensity * (1.0 - saturation * (1.0 - pure)) );
}

/*************************  Generation of HSIColorScales  ***********************/
HSIColorScale :: HSIColorScale( const HSI& from, const HSI& to )
              : foot( from ), temp(0,0,0)
              , hue_range( to.hue - from.hue )
              , sat_range( to.saturation - from.saturation )
              , int_range( to.intensity - from.intensity )
    { if( hue_range < 0.0 ) hue_range += 6.0; }

Color& HSIColorScale :: operator[]( double f ){
    temp =    HSI( hue_range * f  +  foot.hue
                 , sat_range * f  +  foot.saturation
                 , int_range * f  +  foot.intensity );
    return temp;
}

void main () {
    for( int i = 1; i < number; i++ ) {
        RGB rgb = HSIColorScale(MinHsi, MaxHsi) [ double(i) / (number-1) ];    . . .
    }
}
```

Figure 4   Generation and conversion algorithms for HSI color scales.

scalar product. The vector from the grey axis (mid, mid, mid) to the color-point (red, green, blue) results in (mr, mg, mb). The angle α is calculated as

$$\cos\alpha = \frac{(2,-1,-1)\times(mr, mg, mb)}{|(2,-1,-1)|\times|(mr, mg, mb)|}$$

$$\Rightarrow \quad \boldsymbol{hue} = \operatorname{acos}\left(\frac{(2\times mr - mg - mb)}{\sqrt{6}\times\sqrt{mr^2+mg^2+mb^2}}\right).$$

The algorithms for generating HSI color scales and for converting HSI to RGB and vice versa are presented in Figure 4. The parameters for generating the color scales presented in Figure 5 — including the HSI color scale used for the visualizations presented in Figures 6 to 8 — are:

	hue	*saturation*	*intensity*
MinHsi	1.5 (= LightGreen)	1.0	0.4
MaxHsi	1.0 (= Yellow)	1.0	1.0

Since the usefulness of colormaps varies depending on the user and the application, we allow the users to define their own colormaps and use them instead of our standard colormap.

5 SUMMARY AND CONCLUSIONS

Visualizing very large amounts of arbitrary multidimensional data is one of the big challenges that researchers in the graphics/visualization area are currently facing. The task is to efficiently find interesting data sets, i.e. hot spots, clusters of similar data or correlations between different parameters. In this paper, we presented our approach for visualizing large databases. Our techniques provide visual representations of the largest amount of data that can be displayed at one point of time on current display technology. Besides introducing our visualization techniques, we compared the *data space* which is the information contained in the database to the *visualization space* of our visualization techniques, and we defined the HSI color model which produces color scales that are adequate for our distance-to-color mapping.

Interesting topics for future research are an examination of the type of information (type of clusters, type of correlations, etc.) that is perceivable with our visualization techniques, an examination of the impact of different weighting and distance functions, and a comparison of the different visualization techniques for multidimensional data that have been proposed so far. One important step towards an in-depth examination of current visualization techniques for multidimensional data will be an integrated test data generation and evaluation tool which is currently being implemented at our institute. The tool will allow the generation of artificial data sets with given characteristics. The test data sets may be described by the distribution functions for each of the attributes, the correlations or functional dependencies between the attributes, and the clusters which again may have arbitrary characteristics. The tool may be used to evaluate one single visualization technique to find its strength and weaknesses, but it will also be helpful to compare different visualization techniques to find out which techniques are most suitable for which types of data. The goal of our future research is to improve the perception of our visualization techniques and to push their limits to be able to visualize even larger amounts of data with an even higher dimensionality.

6 REFERENCES

Beddow J., (1990) Shape Coding of Multidimensional Data on a Microcomputer Display. *Visualization '90, San Francisco, CA*, 238-246.

Bergeron R. D., Meeker L. D. and Sparr T. M., (1992) A Visualization-Based Model for a Scientific Database System, in *Focus on Scientific Visualization* (eds. Hagen H., Müller M., Nielson G.), Springer, 103-121.

Buja A., McDonald J. A., Michalak J. and Stuetzle W., (1991) Interactive Data Visualization Using Focusing and Linking. *Visualization '91, San Diego, CA*, 156-163.

Feiner S. and Beshers C., (1990) Visualizing n-Dimensional Virtual Worlds with n-Vision. *Computer Graphics*, **24 (2)**, 37-38.

Foley J. D., van Dam A., Feiner S. K. and Hughes J. F., (1990) Computer Graphics: Principles and Practice. *Addison-Wesley, Reading*, 1990.

Frawley W. J., Piatetsky-Shapiro G. and Matheus C. J., (1991) Knowledge Discovery in Databases: An Overview, in *Knowledge Discovery in Databases*, AAAI Press, Menlo Park, CA.

Herman G. T. and Levkowitz H., (1992) Color Scales for Image Data. *Computer Graphics and Applications*, **12 (1)**, 72-80.

Hendee W. R., Wells P. N. T., (1993) 'The Perception of Visual Information', *Springer.*

Inselberg A. and Dimsdale B., (1990) Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry. *Visualization '90, San Francisco, CA*, 361-370.

Keim D. A., (1994) Visual Support for Query Specification and Data Mining. Ph.D. Thesis, University of Munich, Shaker Publishing Company, Aachen, 1995.

Keim D. A. and Kriegel H.-P., (1995) Possibilities and Limits in Visualizing Large Amounts of Multidimensional Data, in *Perceptual Issues in Visualization* (eds. Levkowitz H., Grienstein G.), Springer.

Keim D. A., Kriegel H.-P. and Seidl T., (1994) Supporting Data Mining of Large Databases by Visual Feedback Queries. *Proceedings Int. Conference on Data Engineering, Houston, TX*, 302-313.

Pickett R. M. and Grinstein G. G., (1988) Iconographic Displays for Visualizing Multidimensional Data. *Proceedings Conference on Systems, Man and Cybernetics, Beijing and Shenyang, China*.

Shneiderman B., (1994) Dynamic Queries for Visual Information Seeking, *IEEE Software*, **11**, 70-77.

Treinish L. A., Butler D. M., Senay H., Grinstein G. G. and Bryson S. T., (1992) Grand Challenge Problems in Visualization Software. *Visualization '92, Boston, Mass.*, 366-371.

Turo D. and Johnson B., (1991) Improving the Visualization Hierarchies with Treemaps: Design Issues and Experimentation. *Visualization '92, Boston, Mass.*, 124-131.

7 BIOGRAPHY

Daniel A. Keim is a teaching and research assistant in the Institute for Computer Science at the University of Munich, Germany. His research interests include visualization of statistical data, visual support for querying databases, database support for visualization systems, interfaces to database systems, and interoperability of heterogeneous databases. Keim received his MS from the University of Dortmund in 1990 and his PhD from the University of Munich in 1994.

Hans-Peter Kriegel is a professor for database and information systems in the Institute for Computer Science at the University of Munich, Germany. His research interests are spatial database systems, particularly query processing, performance issues, and parallel operations. Data exploration and data mining in very large spatial databases led him to the area of visualization. Kriegel received his MS and PhD in 1973 and 1976, respectively, from the University of Karlsruhe, Germany.