Quantitative data visualization with interactive KDE surfaces

Martin Florek*
Comenius University, Department of Applied Informatics
Bratislava, Slovakia

Helwig Hauser†
University of Bergen, Department of Informatics
Bergen, Norway

Abstract

Kernel density estimation (KDE) is an established statistical concept for assessing the distributional characteristics of data that also has proven its usefulness for data visualization. In this work, we present several enhancements to such a KDE-based visualization that aim (a) at an improved specificity of the visualization with respect to the communication of quantitative information about the data and its distribution and (b) at an improved integration of such a KDE-based visualization in an interactive visualization setting, where, for example, linking and brushing is easily possible both from and to such a visualization. With our enhancements to KDE-based visualization, we can extend the utilization of this great statistical concept in the context of interactive visualization.

CR Categories: I.3.3 [Picture/Image Generation]: Display algorithms

Keywords: Information Visualization; Data Visualization; Kernel Density Estimation; Linking and Brushing

1 Introduction

For more than 20 years, visualization gives grand opportunities for getting deep insight into various types of data. Most available techniques, however, provide a mostly qualitative form of visualization (in terms of colour codings, projected geometries, etc.). While this is definitely of use in many cases, often a more quantitative visualization would be useful to achieve specific answers to concrete user questions. For example, with rainbow colours user can easily identify a feature, but it is impossible to deduce its exact location, height or distance to other features. Recently, the task of making visualization more quantitative was repeatedly mentioned as one important future goal for visualization research (for example by Thomas Ertl in the Visual Computing Trends symposium in early 2009 [Ertl 2009]).

In this work, we enhance a frequency based visualization method, the Kernel Density Estimation (KDE). It is a proven statistical method for studying distributional characteristics of data, which returns a continuous function which is an estimate of the data's density distribution. KDE in its basic form (Figures 1 – 4) does not provide enough quantitative information, and it is therefore our main goal, to make the KDE surface visualization more quantitative.

We augmented the KDE surface visualization with quantitative measures, such as modes and saddles, for easier features comparison. We do not only visualize these features, but we also use them for other tasks, such as constrained brushing (Fig. 13) based on their height (density). These features are also used for visualizing special contour lines through them, where in the case of a special contour through a saddle, the user can visually identify possible clusters. Further, we introduce a measure called the primary factor (Sec. 4.2) to the KDE surface visualization, which we use to filter peaks, which, by this measure, are considered insignificant.

Even though the main focus of this paper is on communicating the quantitative information to the user, we also achieved interactive speeds thanks to our GPU implementation. Not only with a single KDE surface window, but with multiple windows with different visualization types and within a linking and brushing environment (Fig. 12).

Accordingly, we introduce the following enhancements to a KDE surface visualization:

- we communicate quantitative information about the surface to the user
- enable interaction with the surface, e.g., brushing at the surface itself
- utilize quantitative measures for constrained brushing
- linked interactive views

2 Related work

Smoothing data is not a new idea, it was studied and used more than 80 years ago by Whittaker and Macaulay [Whittaker 1923; Macaulay 1931], but in the field of visualization, there is little related work which concentrates directly on the use of the KDE. Some work was done by Kidwell et al. [Kidwell et al. 2008], who visualized heat maps with nonparametric density estimates instead of a scatterplot. Wegman [Wegman 1990] adapted the average shifted histogram (ASH [Scott 1985]) to parallel coordinates [Inselberg 1985] to compute a parallel coordinates density plot and Artero et al. [Artero et al. 2004] used KDE to compute a frequency plot of parallel coordinates.

Computing and displaying density is common in bioinformatics, where Eilers and Goehman enhanced scatterplots with smoothed densities [Eilers and Goeman 2004] and Hahne et al. use contour plots and density plots instead of scatterplots to study dense cytometry data [Hahne et al. 2006].

Other work with frequency based approaches include work by Johansson et al. [Johansson et al. 2005] who reveal structure in parallel coordinates view with transfer functions applied to density plots. Another interesting work is by Novotný and Hauser who speed up rendering of parallel coordinates of large data by binning [Novotny and Hauser 2006].

Interesting ideas in continuous density reconstruction in scatterplot views where brought by Bachthaler et al. [Bachthaler and Weiskopf 2008], where they generalized scatterplots to the visualization of spatially continuous input data. The most recent work with KDE

*e-mail: martin.florek@fmph.uniba.sk
†e-mail: Helwig.Hauser@uib.no

SCCG 2010, Budmerice, Slovakia, May 13 – 15, 2010.
© 2010 ACM 978-1-4503-0558-7/10/0005 $10.00

was done by Maciejewski et al. [Maciejewski et al. 2009b; Maciejewski et al. 2009a] where they focus on detecting and analyzing hotspots in spatiotemporal datasets [Maciejewski et al. 2009a] using KDE plots with adaptive kernels and multiple linked views. In their another work [Maciejewski et al. 2009b] they apply nonparametric KDE to group voxels, for better transfer function creation for volumetric rendering.

In the field of quantitative KDE visualizations we only found work by Godtliebsen et al. [Godtliebsen et al. 2002], where they focus on 2D KDE plots and overlay the plots with gradient, curvature and streamlines as colourful arrows, dots and lines respectively. They also categorize points by significant curvatures and assign them colour based on the type of the point. However they did not extend their work to 3D surfaces.

In our work we display the 2D KDE as a terrain metaphor and therefore it is important to deal with problems linked to 3D visualizations, such as occlusion and clutter. Another problem with 3D visualizations of scientific data is problematic navigation. Despite these negatives, many propose the use of landscapes [Brandes and Willhalm 2002; Cockburn and McKenzie 2002]. Landscape metaphor can be easily understood by users, because in every day life, they navigate in 3D environment, thus facilitating pattern recognition and spatial reasoning as suggested by Spence [Spence 2001].

Recent work by Melanie Tory et.al. [Tory et al. 2009; Tory et al. 2007] empirically shows that simple dots displays (e.g. scatterplots) are better for some tasks than the landscape interpretation of the same data. They also proved that visualizing terrain as 3D is much better than visualizing it as 2D gray values.So extending the work of Godtliebsen et al. [Godtliebsen et al. 2002] into 3D terrains is a good direction.

3 KDE

Kernel density estimation (KDE) is a popular method for data analysis in the field of statistics, but not so common in data visualization. It was introduced by Rosenblatt [Rosenblatt 1956] and Parzen [Parzen 1962]. KDE constructs a nonparametric estimation of data density.

In this section we summarize the mathematical background of KDE in 1D and 2D.

3.1 1D KDE

Given a set of n data samples $(x_1,\ldots,x_n)|x_i \in \mathbf{R}$ the KDE is

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right) \tag{1}$$

where K is a kernel and h is the bandwidth (smoothing parameter). The kernel function K usually has the following properties:

- $K(x) \geq 0\ \forall x$
- $\int_{-\infty}^{\infty} K(x)\mathrm{d}x = 1$
- and K is centered in 0.

This means, that the kernel function should have nonnegative values and the area (volume in 3D) under the function is 1. Some kernels, e.g., sinc kernel [Bissantz and Holzmann 2007], have also some negative values, though.

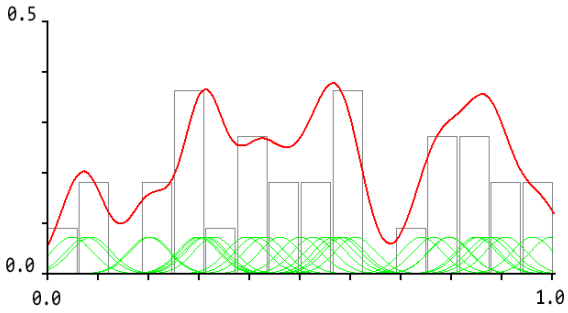

Figure 1: 1D kernel density estimate (in red) for 32 random samples with the bandwidth of 0.04 with the corresponding Gaussian kernels (in green) and a 16-bin histogram (in grey).

The kernel function should be continuous (a box kernel is not very good), such as the Gaussian function. Many kernels have been studied, but the Gaussian kernel is one of the most widely used kernels, so we focus on it in our paper. The Gaussian kernel K is defined as

$$K_h(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2h^2}} \tag{2}$$

Fig. 1 shows an example of a KDE with the corresponding Gaussian kernels and a histogram.

Silverman [Silverman 1986] suggests the use of the Epanechnikov kernel for faster computational time, but our GPU implementation (Sec. 6.1) shows no bottleneck in the computation of the kernel. For a discussion of different kernels we recommend to look at Turlach [Turlach 1993].

3.2 Bandwidth

More important than the choice of the kernel function is the choice of the bandwidth h. Choosing the right bandwidth is very important, because it influences how smooth the KDE will be. A too small bandwidth leads to undersmoothing and a too large bandwidth leads to oversmoothing, as can be seen in Fig. 2.

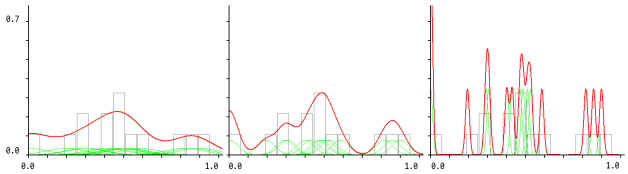

Figure 2: Three KDEs from random data with varying bandwidths from oversmoothed to undersmoothed results.

Statisticians have put a lot of effort into researching automatic methods to find the best bandwidth. Some of these methods are also described by Turlach [Turlach 1993], some even say that the bandwidth should be computed per data sample [Minnotte and Scott 1993]. For example Maciejewski [Maciejewski et al. 2009b] uses the k-nearest neighbours to estimate the kernel, which adapts to the local density.

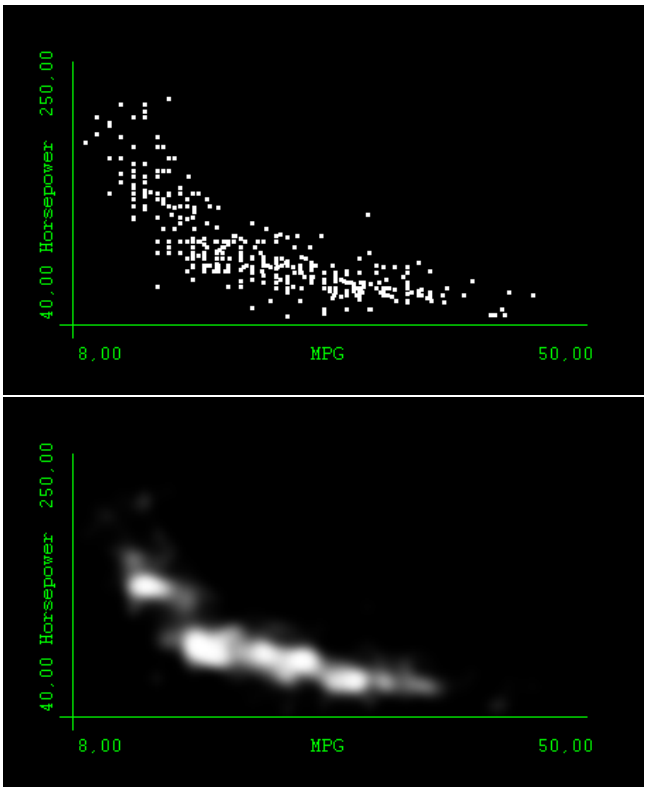

Figure 3: Scatterplot of the cars dataset [Ward et al. 2009] and the 2D KDE from the same data.

3.3 2D KDE

This work's main focus is on two dimensional kernel density estimation plots, so we need an extended 2D kernel. Equation 1 for two dimensions is

$$\hat{f}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{x}-\mathbf{x_i}) \tag{3}$$

where $\mathbf{x} \in \{(x,y)|x,y \in \mathbf{R}\}$ and $\mathbf{H}$ is a symmetric positive-definite bandwidth matrix. In our work we have chosen $\mathbf{H}$ to look like

$$\mathbf{H} = \begin{vmatrix} h & 0 \\ 0 & h \end{vmatrix} \tag{4}$$

so we are limited to only one bandwidth parameter. It is easy to extend it to work with two bandwidths and also allow the rotation of the kernel. For details on how to derive the final equation 3 we refer to Scott and Sain [Scott and Sain 2004], where extensions to arbitrary dimensions can be found. The separable extension of Eq. 1 to 2D with one bandwidth is

$$\hat{f}_h(x,y) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)\cdot K\left(\frac{y-y_i}{h}\right) \tag{5}$$

Fig. 3 shows a comparison between a scatterplot and a KDE plot from the same data.

3.4 KDE surfaces

A KDE surface is a three dimensional visualization of a 2D kernel density estimate, where density values are mapped to the z-coordinate, so the 2D KDE is rendered as a height map (Fig. 4). It is a terrain metaphor from geo-spatial data visualizations, which is empirically proven to be better for certain tasks than simple 2D visualization of gray values [Tory et al. 2009]. Further important reason for displaying the 2D KDE as a landscape is that according to Ware [Ware 2004] only four lightness levels and only up to ten colours can easily be distinguished.

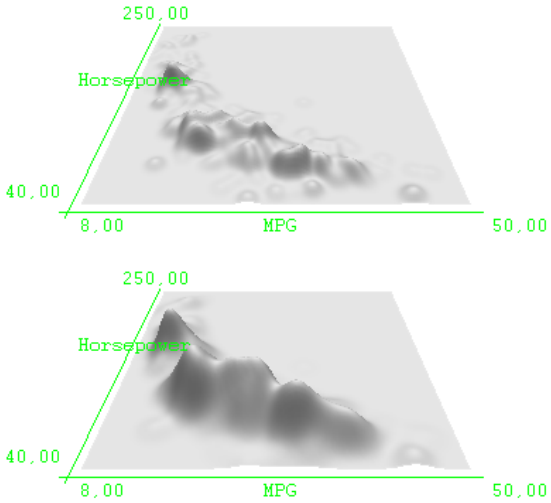

Figure 4: Two examples of KDE surfaces from the same data as in Fig. 3. (top): bandwidth = 0.8; (bottom): bandwidth = 1.2

4 Quantitative measures

One contribution of our work is to augment the KDE surface visualization with quantitative measures. These measures mitigate some problems with 3D environments, such as comparison of distinct features, wich can not be easily done in an simple unaugmented environments due to perspective projection.

4.1 Minimum, maximum and saddles

Local minima and maxima (modes) are identified by the robust mean-shift algorithm [Comaniciu and Meer 2002], which locates features based on the gradient of the density reconstruction function. It is an iterative hill climbing algorithm, where features are located at the zero gradient points $\nabla \hat{f}_h(x) = 0$

Then we visualize local extremes as little colour coded flags, with a number representing the order based on their heights (from the tallest to the smallest) and with the corresponding height (density) value (Fig. 5).

Saddle points are another important quantitative measure, because they could be a separator between clusters. After identifying the saddle points with the mean-shift algorithm [Comaniciu et al. 2002] we visualize them as magenta squares (Fig. 5).

To colour code features we use a colour map with monotonically increasing luminance (Fig. 6), as developed by Wyszecki and Stiles [Wyszecki and Stiles 1982]. The classic hue (rainbow) colour coding is mostly a bad choice [Borland and Taylor II 2007] for representing quantitative data. Our chosen increasing luminance

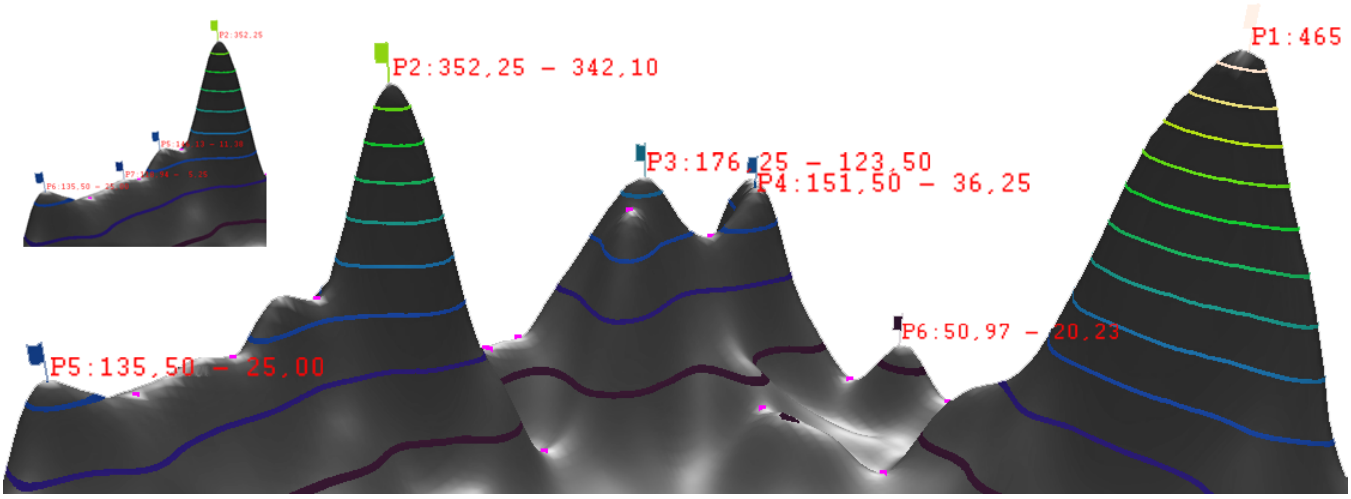

Figure 5: Quantitative measures on a KDE surface. Peaks have colour coded flags on top of them, with a number representing the order from the tallest to the smallest peak, the height and the primary factor. Saddle points are displayed as magenta squares. There are also colour coded contour lines for easier features comparison (with the help of a colour and increasing luminance towards taller areas). Very tall peaks have contours under them closer together because of the nonlinear height remapping. Some peaks are not flagged because they are filtered out by their primary factor measure (4% of the highest peak). The small image at the top left shows part of the bigger image without peaks filtering, with some bumps marked as peaks. This is the *out5d* data set [Ward et al. 2009] with 16384 records and 5 dimensions showing the dimensions *magnetics* vs. *uranium*.

Figure 6: Colour map with monotonically increasing luminance developed by [Wyszecki and Stiles 1982], used for colour coding heights of the features like contour lines, special contour lines and modes' flags.

colour coding, where luminance is increasing towards the more dense areas even more facilitates the comparison of features in a KDE surface displayed with perspective projection, thus mitigating one of the 3D visualizations problems. The increasing luminance is also very good for grayscale reproduction of images, where the rainbow colours fail.

4.2 Primary factor / prominence

The primary factor, also known as prominence, is a measure from topography, which is used to categorize hills and mountains. It is a measure of independence of peaks. One of its definitions is: If a peak has a primary factor of m units, then one must descend at least m units to get to a higher peak (Fig. 7). This means that the primary factor is the difference between the peak's height and the lowest contour line which encircles the peak and no other taller peak. Every peak has a true primary factor except the highest peak (and isolated peaks on islands), where the primary factor equals to their height. This measure classifies peaks as significant ones and insignificant ones.

Computing the primary factor on a 2D grid is a challenging task. For this, we developed an algorithm, which uses the Dijkstra's shortest path algorithm [Dijkstra 1959], which in its basic form can fail because when there exist more than one shortest path, it is unpredictable which one is found as the first.

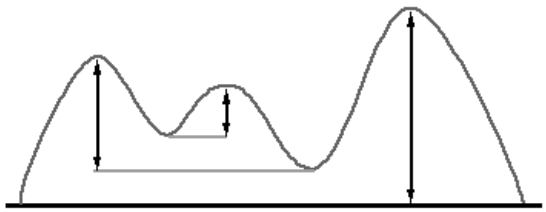

Figure 7: The size of the vertical arrows is the primary factor of the corresponding peak. The primary factor of the tallest peak equals to its height.

We compute the shortest path from every peak to all other points on the KDE surface (this is done in $O(N \cdot \log N)$ time, because the number of the edges in a 2D grid is linearly dependent on the number of vertices, N being the number of grid points). For this we use an eight connected graph, so from every point (except the ones at the edges), there is an edge to all of its eight neighbours. If there is more than one shortest path, then it is unpredictable which path would be found. Therefore we developed a constrained Dijkstra's algorithm, where we constrained the height below which the path can not go. The constraining heights are chosen based on the saddle points, which are lower than the peak. After finding these paths we look for the paths, to all other taller peaks, which has the highest lowest point. This highest lowest point is the point, where we must descend from the peak, to get to a higher peak. Then the difference between the peak's height and this point is the primary factor of this peak.

In our visualization we display the primary factor simply as a number right after the peak's height (Fig. 5). We also allow filtering of the peaks based on their primary factor, so the less independent peaks are not labeled as peaks. This measure also filters out small

bumps, which can be caused by sampling artefacts.

4.3 Data remapping

Another contribution is that we succeeded in fitting a KDE surface into a window, without compromising smaller details in sparse areas. This is necessary when data is grouped in clusters, thus probably forming tall peaks, very dense areas.

To achieve this, we compared three different remapping techniques. We are remapping the density value – z-coordinate of the KDE surface. First, linear downscaling can be used, but when the highest peak is much taller than other data, all the smaller details become flat. The only thing that makes the structure of the sparse areas a bit visible is the lighting, which is computed from the original, not remapped, data (Fig. 8 (b)). Enhancing the smaller details, to give them some height, was achieved by appling an exponential transfer function to the normalized data with the exponent of $1/2$. This nonlinear remapping is giving us the best results, but we also experimented with a third method, density values equalization. It is the same principle as histogram equalization, in fact we are equalizing the density values histogram. This method was giving opposite results to the linear normalization, thus it was equaling taller areas and emphasizing the structure in sparse areas, as can be seen in Fig. 8 (d), where we compare these three methods. This equalization is causing heavy occlusion in the sparse areas, so it is harder to analyze data.

The decision, which remapping technique to use, should be affected by the features we want to study. If we only want to see the whole data in the window, without losing the proportions of the heights and sacrificing details in sparse areas, then the linear downscaling is sufficient. If we can not lose the details in the sparse areas, then an exponential transfer function or an S-shaped transfer function looks promising. And if we want to emphasize the sparse areas, then the equalization method is the best (Fig. 8).

4.4 Contour lines

After nonlinear remapping of the data it is more difficult to visually compare heights (densities) in place of features (modes, saddles, etc.), mainly due to the perspective projection of the camera and partly because of the nonlinear remapping. For more precise visual comparison we render approximate colour coded contour lines on the KDE surface. With these contour lines it is easier to compare the features, thanks to the changing colour and increasing luminance, it is more obvious on the first sight which parts of the data are denser as can be seen in Fig. 5.

These contour lines are equally spaced according to the height value, so if we use nonlinear remapping, the very tall peaks are obvious, even when they are farther away from the camera. It is because the contour lines under these peaks are visually closer together. Contours being closer together towards higher peaks is also facilitating the comparison of the scales between peaks.

4.4.1 Special contours

Our application also allows the rendering of special contours through the features (modes, saddles, etc.). These special contours are distinguished from the regular ones with a checker pattern as can be seen in Figure 9. They enable users to clearly identify the areas of the surface with higher/lower densities than a specific feature. In the case of a special contour through a saddle, users easily spot the border of possible clusters. To distinguish special contours through the modes from special contours through the saddles we use a different secondary colour for the checker pattern. The primary colour of the checker pattern indicates the height (density) of the contour and the secondary colour specifies the type of the special contour (Fig. 9). The peak special contour has white secondary colour and saddle special contour has magenta secondary colour, the same as in marking the saddles on the surface.

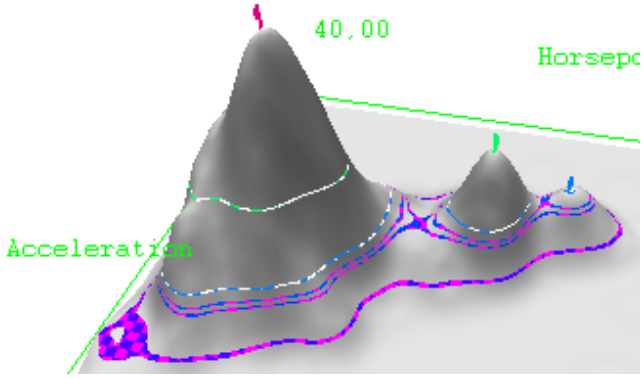

Figure 9: KDE surface from the cars data set (acceleration vs. horsepower, bw= 0.8) with special contours through features enabling visual recognition of possible clusters. Special contours are distinguished from the regular contours (Fig. 5) with checker pattern, where primary colour is signaling the height (density) and the secondary colour is for recognizing special contours through peaks (white) and through saddles (magenta).

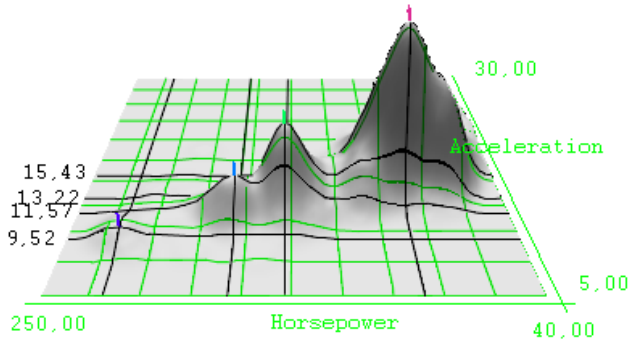

Figure 10: Orthogonal grid over a KDE surface and guide lines through features to facilitate reading of the features' locations in the 3D environment with auxiliary help from the axes and interval values near them. With this enhancement, it is easy for the user to know the order of the peaks from the nearest to the farthest.

4.5 Grid and guides

One of the problems when moving from a 2D visualization to a three dimensional one is that it is harder to read a location of a feature, mainly due to the perspective projection of the camera. We addressed this problem by overlaying an orthogonal grid over the surface and for a more precise localization we draw special guide lines through the features them self (Fig. 10). This guides together with drawn axes, with interval values, near the surface edges facilitates the reading of the features' locations.

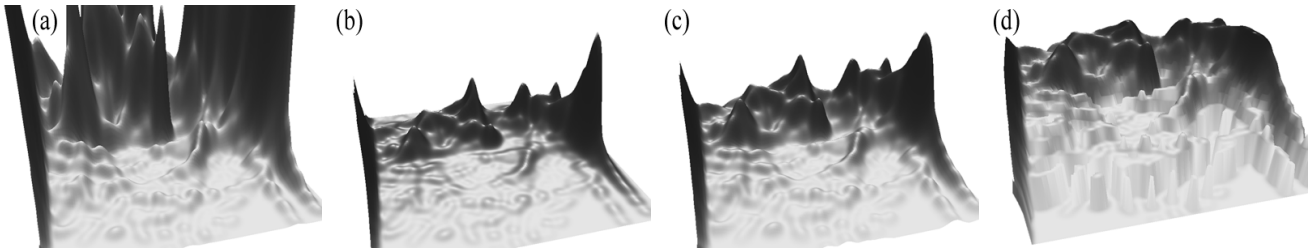

Figure 8: Comparison of the three different remapping methods. Lighting is always computed from the original not remapped KDE surface. (a): no remapping – the KDE surface is much bigger than the window; (b): linear remapping – the whole KDE surface is in the window, but smaller details are flat; (c): exponential remapping – the whole surface is in the window and also the small details remained visible; (d): equalized data as in histogram equalization – the surface fits into the window, details in more dense areas are almost lost but the details in sparse areas are nicely emphasized; This is the *out*5*d* data set [Ward et al. 2009].

5 Linking and brushing

Linking and brushing [Keim 2002] are interaction techniques, which combine selection (brushing) of a subset of the data and linking the results of the brush to other visualizations of the same data.

Here we show a new way of interaction with the data. We give the users the ability to make a selection directly from the KDE surface (Fig. 11 and 12). Our selection implementation is like painting the surface with a brush. First, we project the window 2D coordinates (from under the cursor, where we have clicked) to the 3D world coordinates, so we get the 3D coordinates on the surface. From these coordinates, from the dimensions' intervals and the KDE's surface size and resolution, we get the point, where we clicked, in attributes' space. After this, the selection is like from a scatterplot, but with the difference, that we do not use a rectangular brush, but a circular one. The diameter of the brush is user adjustable. The brush is circular in screen space, but it is elliptical in attributes' space. During the selection the brush is displayed on the surface as a yellow circle (Figures 11 and 12).

The selection is interactive, so during the selection the effect of the brush (addition, subtraction etc.) is instantly visible on the surface and it is also updated in all other views in real time (Fig. 12). It works also in the opposite way, thus when we select from an arbitrary scatterplot or from a parallel coordinates view, the effect is instantly visible on the KDE surface. We also implemented constrained selection, where a user can select a special contour, to which the selection will be constrained, thus only the data in an area with a greater or a lower density will be selected (Fig. 13). To achieve this effect we lookup the density value from the corresponding KDE plot texture (Sec. 6.1) and compare it to the threshold based on the special contour.

For the brush effect to be visible on the surface we visualize the selected data records as red dots on the surface (Fig. 11). For implementation details we refer to the Section 6.1.

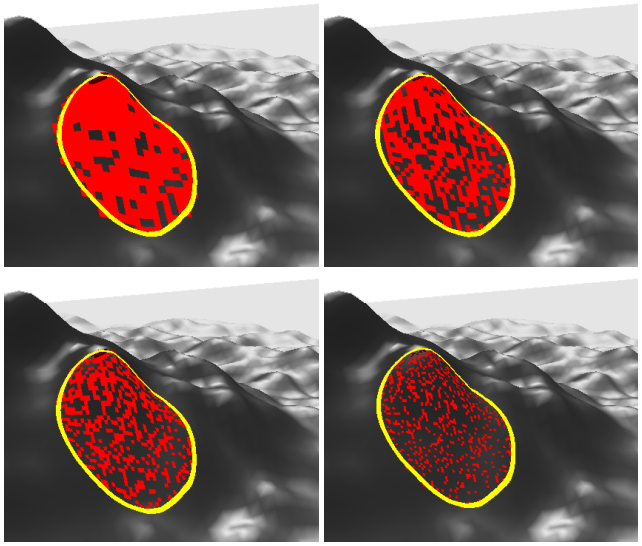

Figure 11: Selection from a KDE surface using a circular brush (yellow), data within the brush get selected (red dots). Effect of a different resolution selection texture. Normal resolution (256^2) selection texture (top left) shows almost no structure in the selected dense data. As we increase the resolution of the selection texture, from top left to the bottom right ($256^2, 512^2, 1024^2$ and 4096^2), structure of the underlying data is revealed. This is the 149769 records large *UVW* data set [Ward et al. 2009].

6 Implementation

In this section we discuss the implementation details of our method, where some of them are implemented on the CPU and some, the computationally more demanding ones, are implemented on the GPU.

All of the quantitative measures are computed on the CPU. The reasons why, are that the grid, on which they are computed, is too small to gain benefits from a GPU implementation. Not computing the primary factor on the GPU is because the single source shortest path in a graph is hard to parallelize and the parallel algorithms does not scale well with the increasing number of processors. Instead of computing one shortest path with parallel algorithm we have chosen to compute multiple shortest paths in parallel. These measures are straightforward to implement on the CPU and their deeper explanation is in the Section 4.

6.1 GPU implementation

Computing and rendering a KDE in real time is demanding task on the computational power. Our GPU implementation achieved interactive frame rates of more than ten frames per second with hundred thousand data records on a laptop with a Mobile Intel® 4 series graphics card.

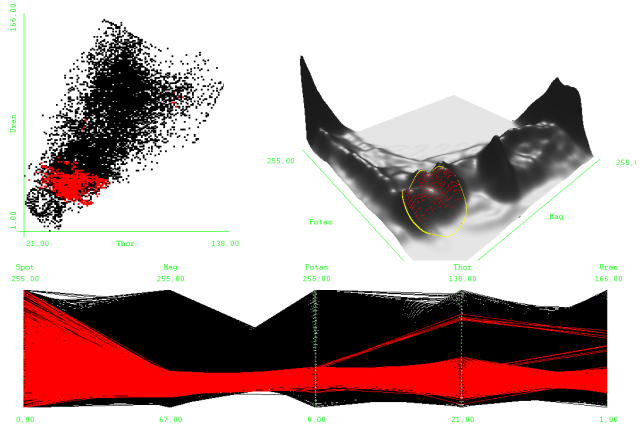

Figure 12: Linking and brushing from a KDE surface. The effect of a selection from the surface (top right – yellow circular brush) is instantly updated in all other visualization views. Data used is *out5d* with 16384 records and 5 dimensions. The scatterplot at top left shows different two dimensions than the KDE surface at top right.

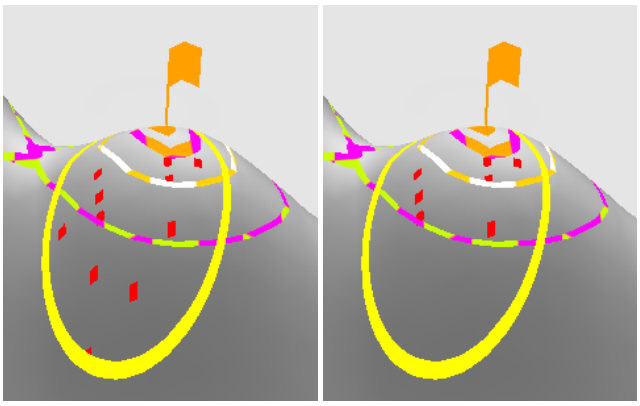

Figure 13: Normal brushing (left) vs. constrained brushing (right) with a constraint on the green saddle special contour – only data above get selected.

6.2 2D KDE

Computing two dimensional KDE is a "GPU friendly" task and its implementation is straightforward, because it is the summation of 2D kernels (in our case 2D Gaussians) and for that we use additive blending. One thing we omit from the basic equation 1 is the normalization by n – number of data records. That is because we later normalize the data in a different way and also nonlinearly remap the data.

Our approach is based on precomputing one kernel into a texture using OpenGL's extension *frame buffer object* (FBO), and then render quads instead of points with the appropriate size (which is based on the bandwidth and the window resolution) with this kernel texture. To render quads instead of points we use OpenGL's extension *point sprites*, which eliminates the overhead of a regular quad – sending one vertex into the graphics card instead of four.

To achieve the summation effect of the kernels, blending must be enabled with additive blending function. Simple blending allows only 256 distinct levels (8-bits of information), which can be enough for smaller or evenly distributed data, but on a real world data it is not enough. To relax this limitation we use OpenGL's FBO extension together with a high precision, 16, 32 or 64 bit, floating point texture. For performance reasons we use 16 bit textures. This high precision texture allows thousands of distinct values. After precomputing the KDE plot into a texture, we read it back to the CPU memory and render it as a height map, lit by one, directional, light from above with Phong's lighting model [Phong 1975]. Landscape lighting and shading is important for better perception of the shape and height of it [Ware 2004]. Data remapping is done in a vertex shader where we pass the maximum density, the maximum height of the surface we want and the exponent for nonlinear remapping. The final 2D KDE plot with exponentially scaled densities can be seen in Fig. 3 – down, and KDE surfaces from the same data is in Fig. 4.

This KDE surface computation is needed again only if we change a parameter like the bandwidth or the surface's resolution or if we change the attributes from which the KDE is computed. If we only want to rotate or zoom to the surface, we can use the computed KDE from the last time, thus saving time and achieving fast interaction.

6.3 Reductions

The maximum density, needed for normalization of the height, is also found on the GPU, with a method called reductions [Göddeke 2006]. The method is very simple, we just render the original texture on a half sized quad, thus choosing from four texels to be rendered to one final pixel. In a pixel shader we lookup these corresponding four texels and output the one with the highest value. We iterate this process until we have output of size 1×1, which is the maximum density of our KDE plot. This method is several times faster than computing it on a CPU, because first we have to download the texture to the CPU memory and then we must traverse through the whole KDE ($O(N^2)$ time, N being the resolution of the KDE) to find the maximum. This task is easily parallelized to achieve better performance on a multi-core/multi-processor systems, but common GPUs have several times more parallel units and despite the fact that the reductions algorithm takes $O(\log(N))$ passes, so the whole algorithm takes $O(N^2 \cdot \log(N))$ time, the GPU algorithm is faster. There is also no bigger memory block copying (from GPU to CPU), only the last single value with the maximum density.

6.4 Contour lines

The contour lines are also displayed with a pixel shader. We send the information about the maximum height of the KDE surface and the number of contour levels we want and then we compute the height distance of a fragment to any contour level. If this distance is within a threshold (empirically chosen), we lookup the appropriate colour from a texture with our colour coding (Fig. 6) based on the fragment's height (density) (Fig. 5).

The special contours through features are like normal contours, but we combine the height colour with a checker pattern and choose the secondary colour of the pattern based on the special contour type (peak – white or saddle – magenta).

6.5 Selection on a KDE surface

When making a selection it is important to see the actual results of it. To address this issue, we visualize the selected records as

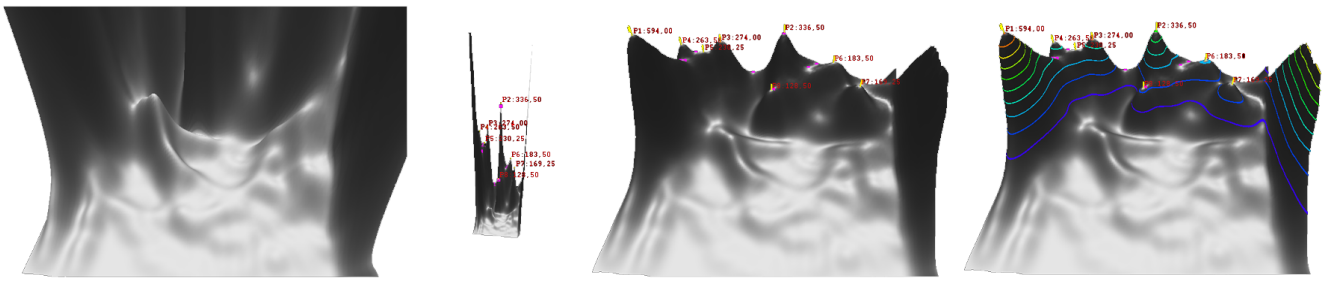

Figure 14: Illustration of the KDE surface enhancements and how they facilitate visual exploration, from the worst (left) to the best (right) visualization: (1): simple KDE surface with no modifications, zoomed-in to show small detail, but we do not see tall peaks; (2): same surface but zoomed out to show tall peaks, small details invisible, peaks have flags; (3): exponentially remapped data to fit into the window, small details are preserved, peaks are marked with flags and heights, saddle points are displayed; (4): same as before, but with colour coded contours for better orientation in heights, so direct comparison of (not only) distinct peaks is easy and intuitive. Together with quantitative measures this visualization is more user friendly than the simple KDE surface seen at the most left.

red dots on the surface. We achieved this effect by rendering the corresponding scatterplot of the selected data into a texture using OpenGL's FBO extension. Then we map this selection texture onto the surface and combine it with other features we render on the surface in a pixel shader. The user can choose between different resolutions of the selection texture, to visualize finer details of the underlying data. Higher resolution selection texture is needed for larger and more dense data (Fig. 11).

7 Conclusion and future work

Our main contribution is making the KDE surface visualization more quantitative with measures like, local maxima (modes), minima, saddle points and the primary factor. We augmented these measures into the KDE surface (Fig. 5) and with the help of special guides and colour coding with monotonically increasing luminance, user can easily deduce spatial locations of these features and compare features with each other. We enable the user to display special contours through the features (Fig. 9), which can visually identify possible separators of clusters. The user can also filter peaks based on their primary factor measure, which classifies modes as significant ones and insignificant ones. This measure can, e.g., filter out (not mark as a peak) small bumps and possible sampling artefacts.

Further we introduced linking and brushing (Fig. 12) on the KDE surface, thus allowing the user to select the data directly from the surface and the brushing effect is instantly updated in all other visualization windows of the same data. We also enable interaction based on the quantitative measures such as constrained brushing on the surface (Fig. 13).

Our GPU implementation proves that it is the highest time to widely start using this powerful method in the field of visual data analysis. We enhanced the KDE plot visualization in several ways. We were able to remap the data so that it fits into a window and that no small details are lost. For this we compared three methods, from which everyone is suitable for a different task (Sec. 4.3, Fig. 8). Advantage of data remapping together with quantitative information can be seen in figure 14.

For future research there are several directions. For example further study the possibilities of the primary factor for KDEs. It could be interestingness measure for peaks ranking. Other direction can be optimizing the transfer function for different tasks, or try *S*-shaped transfer functions or even arbitrary transfer functions. Also the 2D kernel generation can be enhanced by incorporating second bandwidth and even a rotation of the kernel. And at last, new quantitative measures should be visualized on the KDE surface, like valleys and ridges or even the gradient and curvature of the surface. We will also extend the special contour lines rendering to have a better pattern, because as you can see in Figure 9, there are sometimes problems with longer parts of the same colour (the white-green special contour has a long white part). But mainly, we want more quantitative visualizations.

8 Acknowledgement

This work has in part been funded by Slovak Ministry of Education VEGA No. 1/0763/09. Parts of this work have been done in the context of the VisMaster project, which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission under FET-Open grant number 225429. We would like to thank Ove Daae Lampe for technical help and we thank to the authors of the XmdvTool [Ward et al. 2009], to the StatLib [Statlib] web page and especially to prof. Matthew Ward for providing the data sets.

References

ARTERO, A. O., DE OLIVEIRA, M. C. F., AND LEVKOWITZ, H. 2004. Uncovering clusters in crowded parallel coordinates visualizations. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization*, IEEE Computer Society, Washington, DC, USA, 81–88.

BACHTHALER, S., AND WEISKOPF, D. 2008. Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics 14*, 6, 1428–1435.

BISSANTZ, N., AND HOLZMANN, H. 2007. Estimation of a quadratic regression functional using the sinc kernel. *Journal of Statistical Planning and Inference 137*, 3, 712–719. Special Issue on Nonparametric Statistics and Related Topics: In honor of M.L. Puri, Special Issue on Nonparametric Statistics and Related Topics: In honor of M.L. Puri.

BORLAND, D., AND TAYLOR II, R. M. 2007. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications 27*, 2, 14–17.

BRANDES, U., AND WILLHALM, T. 2002. Visualization of bibliographic networks with a reshaped landscape metaphor. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 159–164.

COCKBURN, A., AND MCKENZIE, B. 2002. Evaluating the effectiveness of spatial memory in 2d and 3d physical and virtual environments. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, New York, NY, USA, 203–210.

COMANICIU, D., AND MEER, P. 2002. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24*, 5 (August), 603–619.

COMANICIU, D., RAMESH, V., AND BUE, A. D. 2002. Multivariate saddle point detection for statistical clustering. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, Springer-Verlag, London, UK, 561–576.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 269–271.

EILERS, P. H. C., AND GOEMAN, J. J. 2004. Enhancing scatterplots with smoothed densities. *Bioinformatics 20*, 5, 623–628.

ERTL, T. 2009. The future of scientific visualization. In *VRVis-Symposium – Visual Computing Trends 2009*, vol. http://www.vrvis.at/presse/pressemitteilungen-pdf/vct-2009-talk-ertl.

GÖDDEKE, D. 2006. http://www.mathematik.uni-dortmund.de/ĝoeddeke/gpgpu/tutorial2.html.

GODTLIEBSEN, F., MARRON, J. S., AND CHAUDHURI, P. 2002. Significance in scale space for bivariate density estimation. *Journal of Computational and Graphical Statistics 11*, 1, 1–21.

HAHNE, F., ARLT, D., SAUERMANN, M., MAJETY, M., POUSTKA, A., WIEMANN, S., AND HUBER, W. 2006. Statistical methods and software for the analysis of high throughput reverse genetic assays using flow cytometry readouts. *Genome Biology 7* (August), R77+.

INSELBERG, A. 1985. The plane with parallel coordinates. *The Visual Computer 1*, 4 (Dec), 69–91.

JOHANSSON, J., LJUNG, P., JERN, M., AND COOPER, M. 2005. Revealing structure within clustered parallel coordinates displays. In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, IEEE Computer Society, Washington, DC, USA, 125–132.

KEIM, D. A. 2002. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics 8*, 1, 1–8.

KIDWELL, P., LEBANON, G., AND CLEVELAND, W. S. 2008. Visualizing incomplete and partially ranked data. *IEEE Trans. Vis. Comput. Graph. 14*, 6, 1356–1363.

MACAULAY, F. R. 1931. The Whittaker-Henderson method of graduation. In *The Smoothing of Time Series*, NBER Chapters. National Bureau of Economic Research, Inc, December, 89–99.

MACIEJEWSKI, R., RUDOLPH, S., HAFEN, R., ABUSALAH, A. M., YAKOUT, M., OUZZANI, M., CLEVELAND, W. S., GRANNIS, S. J., AND EBERT, D. S. 2009. A visual analytics approach to understanding spatiotemporal hotspots. *IEEE Transactions on Visualization and Computer Graphics 99*, 2.

MACIEJEWSKI, R., WOO, I., CHEN, W., AND EBERT, D. 2009. Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics 15*, 6, 1473–1480.

MINNOTTE, M. C., AND SCOTT, D. W. 1993. The mode tree: A tool for visualization of nonparametric density features. *Journal of Computational and Graphical Statistics 2*, 51–68.

NOVOTNY, M., AND HAUSER, H. 2006. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics 12*, 5, 893–900.

PARZEN, E. 1962. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics 33*, 3, 1065–1076.

PHONG, B. T. 1975. Illumination for computer generated pictures. *Commun. ACM 18*, 6 (June), 311–317.

ROSENBLATT, M. 1956. Remarks on some non-parametric estimates of a density function. *The Annals of Mathematical Statistics 27*, 832–837.

SCOTT, D. W., AND SAIN, S. R. 2004. *"Multi-Dimensional Density Estimation"*. Elsevier, 229–263.

SCOTT, D. W. 1985. Average shifted histograms: effective nonparametric estimators in several dimensions. *Ann. Statist*, 13, 1024–1040.

SILVERMAN, B. W. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, April.

SPENCE, R. 2001. *Information visualization*. ACM Press books. Addison-Wesley, Harlow, England.

STATLIB. Statlib. *http://lib.stat.cmu.edu/*.

TORY, M., SPRAGUE, D., WU, F., SO, W. Y., AND MUNZNER, T. 2007. Spatialization design: Comparing points and landscapes. *IEEE Transactions on Visualization and Computer Graphics 13*, 6, 1262–1269.

TORY, M., SWINDELLS, C., AND DREEZER, R. 2009. Comparing dot and landscape spatializations for visual memory differences. *IEEE Transactions on Visualization and Computer Graphics 15*, 1033–1040.

TURLACH, B. A. 1993. Bandwidth selection in kernel density estimation: A review. In *CORE and Institut de Statistique*, 23–493.

WARD, M. O., RUNDENSTEINER, E. A., CUI, Q., XIE, Z., YANG, D., WAD, C., AND NGUYEN, D. Q., 2009. XmdvTool.

WARE, C. 2004. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

WEGMAN, E. J. 1990. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association 85*, 411, 664–675.

WHITTAKER, E. T. 1923. On a new method of graduation. *Proc. Edinburgh Math. Soc*, 41, 63–75.

WYSZECKI, G., AND STILES, W. S. 1982. *Color Science: Concepts and Methods, Quantitative Data and Formulae*, 2 ed. Wiley-Interscience, September.