Interactive Bivariate Mode Tree

Martin Florek
Comenius University, Department of Applied Informatics
Bratislava, Slovakia
florek@sccg.sk

Helwig Hauser
University of Bergen, Department of Informatics
Bergen, Norway
Helwig.Hauser@uib.no

ABSTRACT

Kernel density estimation (KDE) is widely used statistical method to study distribution of the data. The problem with this method is choosing the right bandwidth. In our work we focus on semiautomatic bandwidth selection with the use of visual paradigm of the Mode Tree. We not only enhanced the bivariate mode tree visualization, which was only sketched by the authors, but we also developed a hybrid CPU/GPU implementation to improve the speed of the mode tree construction.
Keywords: Mode Tree, Data Visualization, Kernel Density Estimation, GPU.

1 INTRODUCTION

Nonparametric density estimation was overlooked for years as an interactive visual data exploration tool. It was mainly due to the intense computational complexity, but with todays GPU algorithms even a low end graphics hardware can compute a kernel density estimate in real time. Current hardware allow us to interactively change the parameters and see the results instantaneously, but the search for the best bandwidth is not easily solved even if we can compute and observe tens of different KDEs in real time.

Thus in our work we picked up the mode tree visual paradigm and brought it to interactive life with a GPU algorithm. This powerful tool is not widely used, because one needs to compute tens to hundreds of different KDEs, which is a time consuming task. The Mode Tree is a powerful graphical tool for choosing the right bandwidth for KDE construction, it can reveal distributional characteristics, such as multimodality. Here we present an implementation which is capable of constructing a bivariate mode tree on a main stream hardware in interactive rates and we also enriched the visualization with following enhancements:

- height and depth augmented visualization for easy identification of important features and suppressing the outliers
- mitigated clutter of the 3D plot
- interactive hybrid CPU/GPU implementation

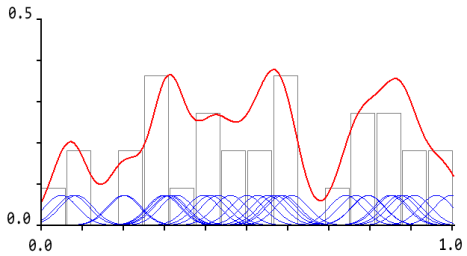

Figure 1: 1D kernel density estimate (in red) for 32 random samples with the bandwidth of 0.04 with the corresponding Gaussian kernels (in blue) and a 16-bin histogram (in grey).

2 RELATED WORK

KDE (Fig. 1) is a well established visual analysis method in many fields, e.g. in bioinformatics [2, 3] and in visualization of heat maps [5]. This idea was also adapted to the parallel coordinates [4] visualization [21, 1].

In nonparametric density estimation, the kernel function choice does not influence the final density reconstruction that much. More important is the choice of the right bandwidth, the smoothing parameter. This choice effects how much of the underlying structure will be exposed to the user. Wrong bandwidth can lead from undersmoothed to oversmoothed reconstruction, as can be seen in Fig. 2.

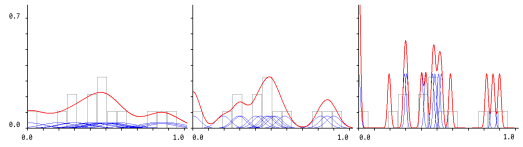

Figure 2: Three KDEs from random data with varying bandwidths from oversmoothed to undersmoothed results.

Scientists were trying to find automatic methods to find the best bandwidth [18], some say that the bandwidth should vary with data sample [10]. In the adaptive bandwidth selection, the work by Maciejew-

ski [7] uses the k-nearest neighbours to estimate the kernel.Until now, there is no completely satisfactory automatic method to find the right bandwidth and many real world data show, that one global bandwidth is insufficient.

For this reason a visual tool called the Mode Tree was introduced by Minnotte and Scott [11]. This method summarizes information from density estimates computed for many different bandwidths h (Sec 4). The mode tree was further enhanced to display modes' sizes, antimodes and bumps. Later Marchette and Wegman introduced the Filtered Mode Tree which is constructed from filtered kernel estimator [8]. Minnotte et al. also developed a Mode Forest [9], which is a simultaneous view of many mode trees, which are based on original data but with variations such as resampling and jitter. This method can filter out outliers which can strongly affect the simple mode tree.

Recent research on mode trees was done by Klemalä [6] who generalized the idea of mode tree to multivariate settings in form of multiframe mode graphs.

3 KDE

Kernel density estimation (KDE) was introduced by Rosenblatt [14] and Parzen [12]. Given a set of n data samples $(x_1,\ldots,x_n)|x_i \in \mathbf{R}$ the KDE is

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

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. In our work we focus on the Normal kernel and figure 1 shows an example of a KDE with the corresponding Gaussian kernels and histogram.

3.1 2D KDE

Focus of this work is on the bivariate mode tree, so we need an extended 2D kernel to compute 2D density estimate. Equation 1 for two dimensions is

$$\hat{f}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{x}-\mathbf{x_i}) \qquad (2)$$

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 decided to use two different bandwidths (each for one dimension) and not to use the full potential of the matrix $\mathbf{H}$, which would also allow rotation of the kernel. The final separable extension of equation 1 to 2D with two bandwidths looks like this (Eq. 3) and an example 2D plot can be seen in figure 3.

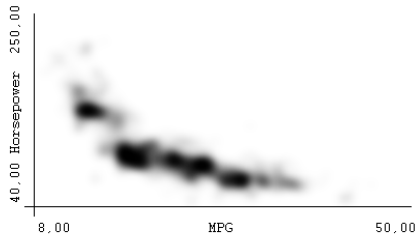

Figure 3: Example of a two dimensional KDE from the cars dataset [20].

$$\hat{f}_h(x,y) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h_x}\right)\cdot K\left(\frac{y-y_i}{h_y}\right) \qquad (3)$$

For details on how to derive the final equation 3 we refer to Scott and Sain [15], where extensions to arbitrary dimensions can be found.

4 MODE TREE

For 1D KDE the mode locations are plotted against the bandwidth at which the density estimate with those modes is calculated [11]. In Fig. 4 the solid vertical lines represent the modes corresponding to those in the density estimates. Important thing in the mode tree construction is to use a logarithmic scale for choosing the different values of h, because at high values of h large changes have lesser effect on the KDE than smaller changes at lower values of h. Plotted locations of all modes result in a set of lines called mode traces.

Silverman [16] proved, that for the normal kernel the number of zeros in all derivatives of $\hat{f}_h$ is decreasing for increasing h. This implies that also the number of modes is decreasing as the bandwidth is increasing.

5 2D MODE TREE

The mode tree is very useful tool in one dimensional case and has a potential to be good with 2D KDE. In the bivariate case it becomes a 3D plot instead of 2D, where on $X-$axis is the first dimension, on $Y-$axis is the second dimension and the $Z-$axis represents increasing bandwidths and the streaks represent modes' locations (Fig. 5).

Minnotte et al. in their work only sketched the idea of bivariate mode trees [11]. They also had to resample the data to have variance equal to 1, because they used only single bandwidth for both dimensions. In our work we do not have a limitation of resampling the data and of single bandwidth.

One of the biggest drawbacks of mode trees is their computational complexity, which is even bigger problem in the bivariate case. Recent main stream GPUs

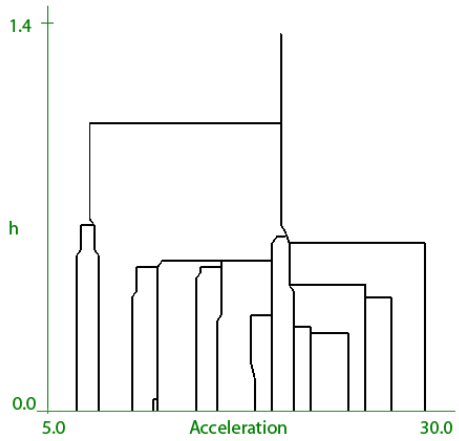

Figure 4: 1D Mode Tree. Solid vertical lines represent the modes corresponding to those in the density estimates, where the bandwidth is on the Y-axis. Horizontal lines are links from extinct modes to the modes, which consumed them. Plotted from 100 KDEs.

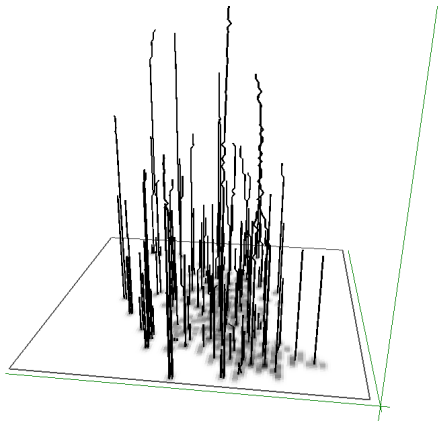

Figure 5: The bivariate Mode Tree as defined by Minnotte and Scott [11]. The streaks represent modes' locations on 2D KDE for increasing bandwidths (the $Z-$coordinate). At the bottom of the mode tree there is the 2D KDE of the lower bandwidth limit.

can compute the 2D KDE in interactive times and we modified the current GPU algorithm to be even more efficient for the mode tree computation (Sec. 7.1).

6 ENHANCED BIVARIATE MODE TREE

In this work we focus on enhancing the bivariate mode tree, not only to mitigate the problems of three dimensional plots (occlusion, navigation, etc.) but also make the visualization more informative and clear to the user. We also developed a GPU implementation to compute the 2D mode tree and for smaller data sets we achieved instantaneous construction times (Sec. 7.1).

One very important information, which is not communicated to the user in the original mode tree, is the height of the modes. The mode tree tells us only where the modes are and nothing more. To address this issue we used special color coding with monotonically increasing luminance of the color (Fig. 6), as developed by Wyszecki and Stiles [22].

Figure 6: Color coding with increasing luminance used to encode mode's height.

The color and the brightness represent the relative height of the modes, where the black color represents the tallest mode and streaks with lighter colors are the smaller modes, possible outliers. This color coding makes the significant modes stand out, while the small modes, outliers, are more hidden from the viewer.

Another problem of all 3D plots, not only bivariate mode tree plot, is occlusion, so the features closer to the camera occlude features farther away. This can lead to hiding important features behind unimportant ones, in some camera positions. To mitigate the problems with occlusion, we used semi transparent mode streaks, where the transparency values are based on the relative height of the mode to the current tallest mode. This means that small modes are represented not only by lighter streaks (which could occlude dark streaks), but the streaks are also semi transparent (Fig. 7). The smaller the mode, the more transparent it is, thus ensuring that significant modes represented by dark streaks are always clearly visible and only subtly overlayed by the outliers. If the user wants to study outlying modes, then the color coding and semi transparency can be inverted.

The last problem of the 3D plots we addressed in this work is difficult spatial location deduction. When looking at a simple bivariate mode tree (Fig. 5) it is hard to deduce the correct locations. Even an approximate location is very hard to guess. We adapted a simple idea from [13], where they made contour thickness of a point in space dependent on distance to the camera. So we made the mode streaks closer to the camera more thicker and the far away streaks are thinner, which improves depth perception (Fig. 7).

7 IMPLEMENTATION

Our implementation is a combination of a CPU and GPU algorithm. The KDEs are computed on the GPU (Sec. 7.1) and the identification of modes and mode traces construction is done on the CPU. Finding modes on a 2D grid is a simple task, but constructing the mode streaks is a more challenging one.

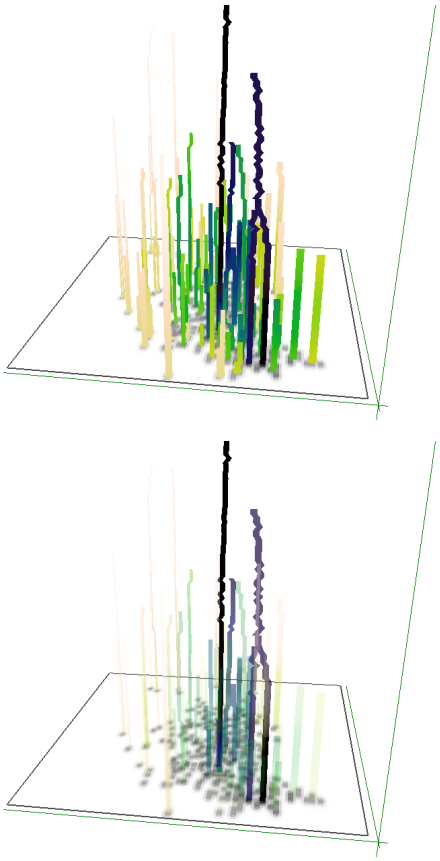

Figure 7: Enhanced bivariate mode tree: (top) Color coding enabled, which reveals relative height of the modes to the tallest mode, but with some small modes occluding taller ones. (bottom) With transparency enabled, mitigating the clutter caused by plotting the mode streaks in 3D. Only the important peaks are clearly displayed and also their starting positions are always visible. Streaks closer to the camera are more thick and the far away streaks are thinner, for better depth perception.

First we start by computing many KDE's with different bandwidths, from zero to the variance of the data. The default value for the number of density estimates was set to 64 and the bandwidth values are equally spaced on a logarithmic scale. For the reason we refer to the section 4. After this we identify the modes and then run a matching algorithm to connect the modes' locations for different bandwidths to construct a mode streak. For this we developed a simple multipass heuristic, where in the first pass we connect modes which are directly above each other and remove these modes from the queue of unprocessed modes. Then we run multiple passes where in each pass we try to find a corresponding mode streak for each unprocessed mode, within an increasing radius.

7.1 GPU implementation

Computing hundreds of KDEs is computationally the most intensive task and therefore is entirely done on a GPU. For this we developed a GPU algorithm, where we precompute four Gaussian kernels into an RGBA floating point texture, each for a different bandwidth and to a different texture channel. Then we use a splatting technique, which results in four KDEs computed with different bandwidths, in the same time as one. The thing that slows down the construction the most, is not the KDEs computation, but the construction of mode streaks, which is done on a CPU (Sec. 7).

Table 1 shows performance values for different data sets and different GPUs, from low-end laptop Intel 4 mobile GPU to a powerful AMD HD5700. The times vary for the same GPU, because the construction time is heavily dependent on the point matching algorithm, which depends on the number of modes. We can compute mode tree in real time even on low end hardware.

data / GPU	Intel 4 mobile	AMD HD5700
cars (400)	200 - 300 ms	60 - 90 ms
out5d (16k)	2800 - 3900 ms	600 - 800 ms

Table 1: Time of 2D mode tree construction, with 128×128 KDE resolution and 100 different bandwidth values. The cars data set has 400 records and out5d has 16000 records.

8 CONCLUSION AND FUTURE WORK

This work brought attention to an old idea of the Bivariate Mode Tree. We achieved not only an interactive hybrid GPU/CPU implementation, but we also enhanced the mode tree visualization with information about the height of the modes for easy identification of important modes. With making the unimportant modes transparent, we removed some clutter, which is one of the biggest problems of 3D plots. We also made modifications for better depth perception of mode streaks for easier spatial localization.

In the future we want to further enhance the plot, for example with better outliers visualization (stippled lines) and with links from extinct modes to the modes which consumed them. Further we would like to improve the performance of our GPU algorithm with the use of multiple render targets, which will allow us to render $16-32$ KDEs at once with only a small time increase compared to the current implementation. And the last thing we will do is a better initial camera placement, which is very important in 3D plots, but is not a trivial task. This could be achieved by geometric best view algorithms [19], which we will have to adapt to the lines geometry.

9 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 authors of the XmdvTool [20], to the StatLib [17] web page and especially to prof. Matthew Ward for providing the data sets.

REFERENCES

[1] Almir Olivette Artero, Maria Cristina Ferreira de Oliveira, and Haim Levkowitz. Uncovering clusters in crowded parallel coordinates visualizations. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization*, pages 81–88, Washington, DC, USA, 2004. IEEE Computer Society.

[2] Paul H. C. Eilers and Jelle J. Goeman. Enhancing scatterplots with smoothed densities. *Bioinformatics*, 20(5):623–628, 2004.

[3] Florian Hahne, Dorit Arlt, Mamatha Sauermann, Meher Majety, Annemarie Poustka, Stefan Wiemann, and Wolfgang Huber. Statistical methods and software for the analysis of high throughput reverse genetic assays using flow cytometry readouts. *Genome Biology*, 7:R77+, August 2006.

[4] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(4):69–91, Dec 1985.

[5] Paul Kidwell, Guy Lebanon, and William S. Cleveland. Visualizing incomplete and partially ranked data. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1356–1363, 2008.

[6] Jussi Klemalä. Mode trees for multivariate data. *Journal of Computational and Graphical Statistics*, 17(4):860–869, December 2008.

[7] Ross Maciejewski, Insoo Woo, Wei Chen, and David Ebert. Structuring feature space: A nonparametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1473–1480, 2009.

[8] David J. Marchette and Edward J. Wegman. The filtered mode tree. *Journal of Computational and Graphical Statistics*, 6(2):143–159, June 1997.

[9] Michael C. Minnote, David J. Marchette, and Edward J. Wegman. The bumpy road to the mode forest. *Journal of Computational and Graphical Statistics*, 7(2):239–251, June 1998.

[10] Michael C. Minnotte and David W. Scott. The mode tree: A tool for visualization of nonparametric density features. *Journal of Computational and Graphical Statistics*, 2:51–68, 1993.

[11] Michael C. Minnotte and David W. Scott. The mode tree: A tool for visualization of nonparametric density features. *Journal of Computational and Graphical Statistics*, 2:51–68, 1993.

[12] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[13] Harald Piringer, Robert Kosara, and Helwig Hauser. Interactive focus+context visualization with linked 2d/3d scatterplots. In *In proceedings of the International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2004*, pages 49–60. IEEE Computer Society, 2004.

[14] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27:832–837, 1956.

[15] D. W. Scott and S. R. Sain. *"Multi-Dimensional Density Estimation"*, pages 229–263. Elsevier, 2004.

[16] B. W. Silverman. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society. Series B (Methodological)*, 43(1):97–99, 1981.

[17] Statlib. Statlib. *http://lib.stat.cmu.edu/*.

[18] Berwin A. Turlach. Bandwidth selection in kernel density estimation: A review. In *CORE and Institut de Statistique*, pages 23–493, 1993.

[19] Pere-Pau Vázquez and Mateu Sbert. Fast adaptive selection of best views. In *ICCSA'03: Proceedings of the 2003 international conference on Computational science and its applications*, pages 295–305, Berlin, Heidelberg, 2003. Springer-Verlag.

[20] Matthew O Ward, Elke A Rundensteiner, Qingguang Cui, Zaixian Xie, Di Yang, Charudatta Wad, and Do Quyen Nguyen. XmdvTool, 2009.

[21] Edward J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

[22] Günther Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley-Interscience, 2 edition, September 1982.