Interactive Information Visualization using Graphics Hardware

Martin Florek*
Comenius University Bratislava

Matej Novotný†
Comenius University Bratislava

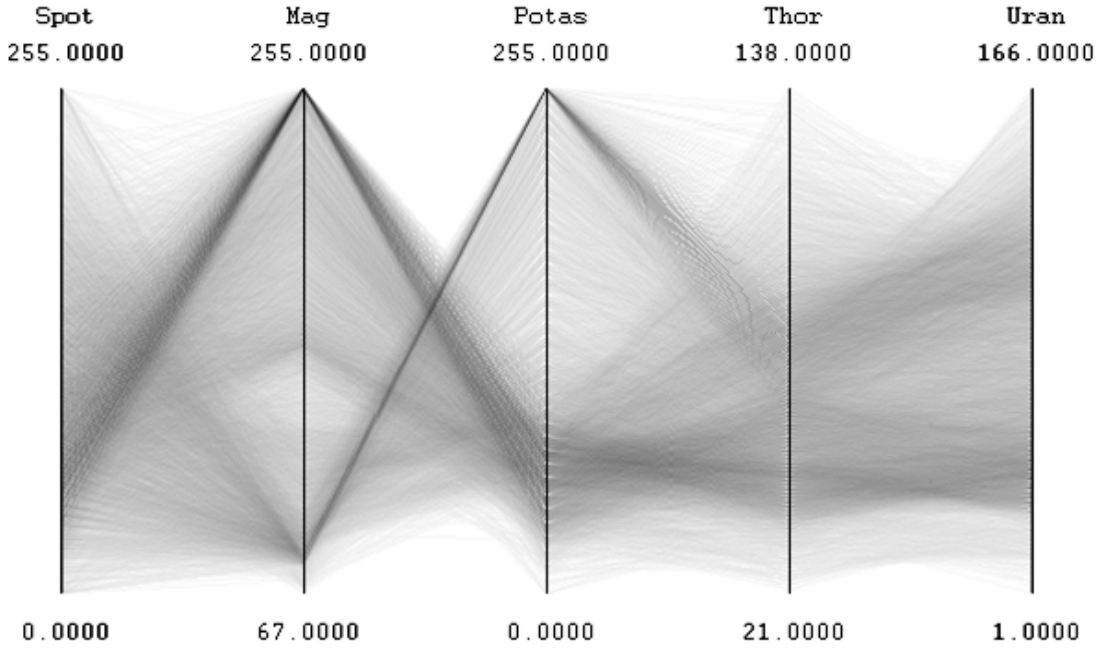

Figure 1: A parallel coordinates display rendered using proper hardware acceleration is capable of visualizing hundreds of thousands of records. Thanks to overcoming the performance penalty that is usually caused by massive overplotting of semi-transparent lines, this display does not lose the essential interactivity.

Abstract

The actual technology behind visualization is meant to provide fast and accurate rendering results and thus to support the exploratory data analysis in an information visualization environment. A crucial element in the exploratory process is overall interactivity, especially when dealing with multidimensional data. However, with the ever growing volume of nowadays data, a fast and interactive display easily becomes static, giving only slow or even none actual interaction feedback.

The technical improvements presented in this paper are oriented on using modern graphics hardware to modify the standard rendering process of a popular information visualization display, the parallel coordinates, in an interaction-oriented way. By intelligently using the advanced features of the now common hardware we achieved great improvement over standard CPU-oriented implementations in terms of both speed and visual quality.

Keywords: GPU, hardware accelration, information visualization, parallel coordinates

*e-mail:mflorek@gmail.com
†e-mail:mnovotny@fmph.uniba.sk

1 Introduction

Information visualization uses graphical representation of data to support and accomplish important tasks like decision making, data exploration or analysis. Compared to scientific visualization, where the data usually contains an underlying spatial geometry, the data in the information visualization domain are often highly multidimensional and usually have no a priori structure or layout.

The motivation for graphical depiction of data is the wide information highway that is provided to humans through the sense of vision. Even some of the most complicated structures and information can (using a proper visualization) be communicated between the man and the machine. Numerous projection methods and graphical metaphors were designed in the field of information visualization [Tufte 1990; Ware 2000; Card et al. 1999]. However, the fully comprehensive mental image of a multidimensional information is only built through the means of user interaction [Kosara et al. 2003].

1.1 Interaction

By performing direct manipulation inside the display, through observing the data from different aspects and under different conditions the users immerse themselves in the data. If the display reacts within a fraction of a second (say 100 ms) the user gets the feeling of actually touching the data [Eick and Wills 1995] and can better

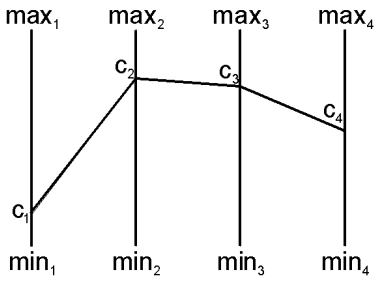

Figure 2: Parallel Coordinates. Point $C(c_1, c_2, c_3, c_4)$ is represented by a polygonal line.

understand the intrinsic structures of the observed space or model.

In an information visualization environment the interaction provides (among others) an access to changing parameters of the visualization and to selecting and emphasizing areas of interest. But if the actual time to refresh the display after such an action takes too long for the user to notice the difference or to perceive the fluent changes the interaction suffers greatly. This happens for large data cases in combination with demanding visualization techniques. Parallel coordinates are an example of a popular and widely acclaimed visualization method that faces severe problems when large data is observed.

1.2 Parallel Coordinates

As originally introduced by [Inselberg and Dimsdale 1990], the parallel coordinates utilize the axis reconfiguration approach to multi-dimensional data visualization. Every *n*-dimensional point is represented by a polyline according to its position in the original space (Figure 2.)

N copies of the real line are placed equidistant and parallel to each other. They are the axes of the parallel coordinate system for R^N. A point C with coordinates $(c_1, c_2, \ldots, c_N)$ is represented by a poly line connecting the positions of c_i on their respective axes [Inselberg and Dimsdale 1990].

This projection provides a 2-dimensional display of the whole data set and is capable of displaying up to tens of different dimensions. An unpleasant drawback is the cluttered display when trying to render a large number of samples. Interaction and even mere understanding of such a display is complicated.

A common graphical feature of the parallel coordinates is semi-transparency of the poly lines. It gives better understanding of an overplotted display by steering the visual importance towards the dense areas where multiple semi-transparent lines overlap. However this improvement has certain limitations due to the dynamic range of the alpha channel [Johansson et al. 2005]. In addition to that the use of transparency and color blending introduces a significant performance issue.

2 Incorporating the GPU

The current graphics hardware with its wide range of advanced processing features provides a promising solution to improve the interactivity of large data visualization. Scientific visualization with numerous volumetric rendering approaches and applications uses the graphics hardware for quite a long time now.

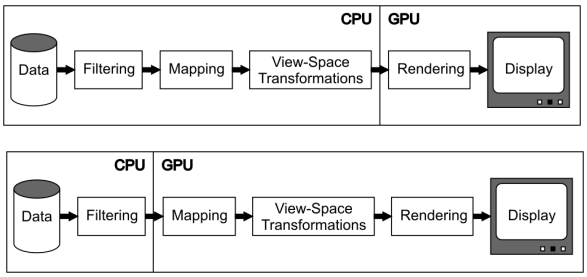

Figure 3: A simple visualization pipeline. In its original form (top) the CPU has to perform much more operations that now (bottom).

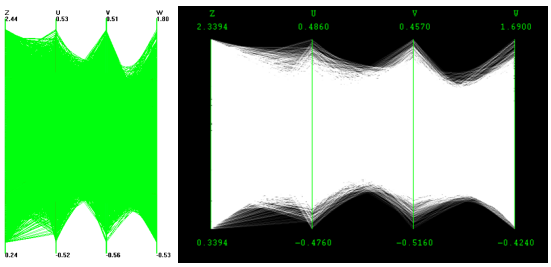

Figure 4: The same data containing 150.000 samples. Rendering the parallel coordinates plot using the popular free visualization tool Xmdv took more than 20 seconds (left). Our implementation (right) of parallel coordinates works interactively with the data.

However in information visualization domain, the actual technological improvements using the GPU receive little attention. In contrast to that, this project focusses on accelerating parallel coordinates using the GPU in order to produce a display that is capable of displaying data sets with tens or hundreds of thousand samples while still providing fast interaction options to the user (Figure 4)

2.1 GPU vs CPU

The parallel coordinates projection has a well defined and simple geometric nature which nicely favors GPU over CPU. With the powerful geometry processing units and parallel pipelines of the GPU many sub-tasks of the visualization process can be transferred to the graphics hardware (Figure 3.)

In addition to that, in the current state of visual exploration, more and more attention is drawn towards interleaving automatic data analysis with human visual processing. Therefore practical application-oriented load balance between the CPU and the GPU is an important issue.

The comparison of using CPU and GPU to do visualization tasks is illustrated in Figure 4. The left picture shows a rendition by the Xmdv visualization tool [Xmd] that took 20 seconds to accomplish. The GPU-powered solution (right) performs about 60 times faster.

2.2 GPU features

Even though the advanced rendering and processing capabilities of nowadays graphics cards are originally oriented on producing realistic images and special effects in real time, much of this functionality can be used to improve information visualization.

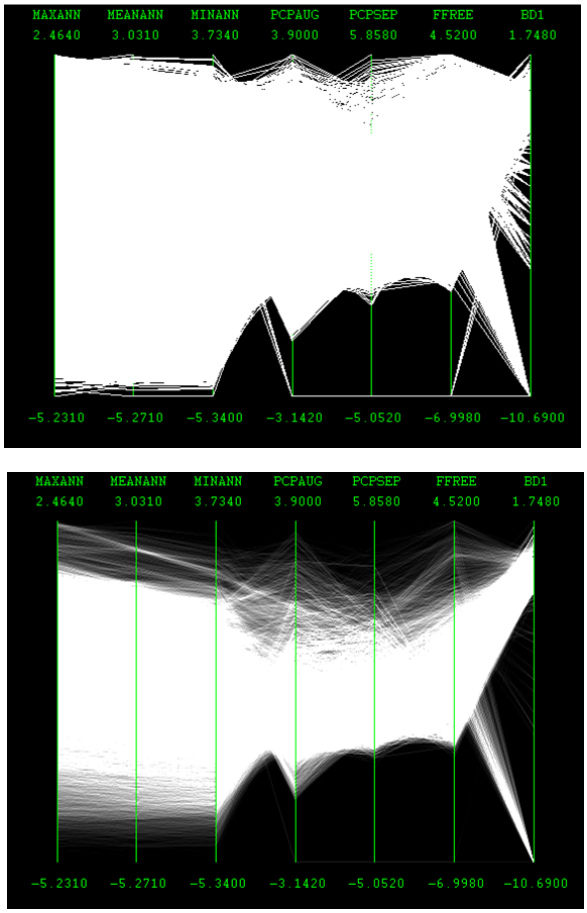

Figure 5: Transparency and alpha blending turn a cluttered and indistinct display (left) into a much clearer shape (right).

Apart from the actual processing features, the memory of the graphics adapter is another beneficial factor. Storing the data for the visualization in the VRAM instead of the RAM improves the performance in two ways. First, the CPU is less loaded since the vertex data do not have to be sent to the GPU by the CPU. Second, the fetching time when data is read to be processed by the GPU is much shorter.

3 Data Operations on GPU

This section describes the implementation and the experimental results of a hardware accelerated parallel coordinates display. The small space in this paper does not allow for much technical details. However the detailed description of the overall contribution as well as issues brought up during the implementation will appear in [Florek 2006].

3.1 Vertex Arrays

The first natural step to move the border between the CPU and the GPU closer to the starting point of the visualization pipeline was to store the actual data on the graphics card and operate on them using vertex programs.

Nowadays graphics cards are common to have up to 512 megabytes of memory, which is technically sufficient to store a decent load of data (e.g. 4 millions of data records in 32 dimensions).

The naive immediate mode for rendering uses the CPU to send individual vertexes to the GPU to render. Instead of that we can group the vertex data to sets of batches that can be sent to the graphics card using only a small portion of the CPU time. Even a simple modification, as this is, produces great improvements to the application producing a convincing resulting performance (Figure 4.)

3.2 Vertex Programs

A necessary extension to the previous improvement is to limit the need for repeated transferring of large amounts of vertex data from CPU to GPU. Therefore we store the raw original data on the graphics card and perform mapping operations by vertex programs.

Using the advanced shading language of the current graphics hardware many axis-oriented operations can be performed on the GPU without having to occupy CPU or RAM. Among others, the mapping between the original data space and the screen space of the parallel coordinates is feasible using vertex programs.

By default this mapping is a linear scaling performed on each axis. Usually after interaction this can change so that the original mapping is modified by panning, flipping or scaling the interval that is mapped onto the axis. Thanks to the state-of-the-art shader capabilities non-linear axis mapping functions (square, square root) are also possible.

4 Alpha Blending and Stencil Test

A common feature of a parallel coordinates display is the use of semi-transparent poly lines to clear up an overplotted display. The resulting frequency-like graphical representation communicates more information than a cluttered and indistinct visualization without the transparency. Dense areas are preattentively emphasized by higher opacity values in contrast to the sparsely populated areas which are less saturated (Figure 5.)

However introducing transparency to the visualization brings the unpleasant drawback of reduced performance. The cause is the large amount of fragments created by numerous lines overlapping in different relative orientations and the additional computational demands put on the graphics cards to process the fragments.

Even with simple blending the performance is reduced by up to 70 percent in an average parallel coordinates plot. This could easily make a visualization incapable of interactive response. We used stencil test performed on the GPU to eliminate a significant number of fragments in order to compensate for the performance loss.

The resulting visualization using the stencil test operates 2.5 times faster than without the test reducing the 70 percent performance penalty to less than 25 with the alpha blending on.

5 Frame Buffer Objects and Textures

Another performance boost is achieved by reusing rendering output in cases when no data-oriented operation is performed. Such cases involve resizing the display, local axis operations or composing several segments of data in the display. To achieve this, all rendering is routed to frame buffer objects and subsequently stored in a texture.

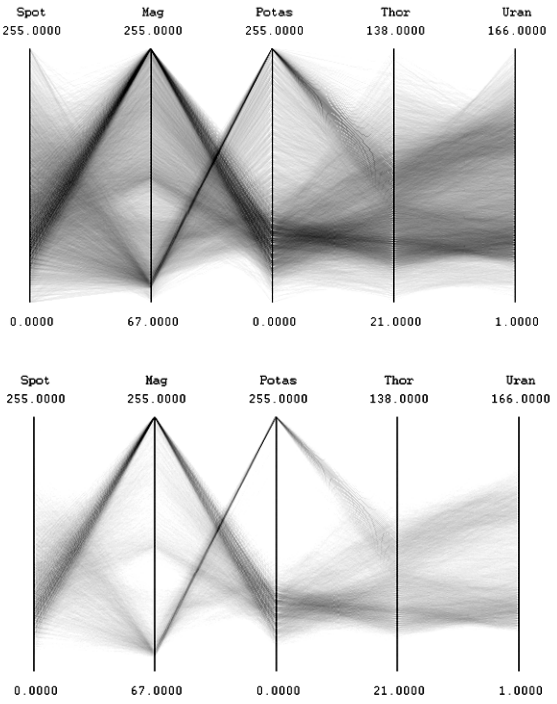

Figure 6: By changing only the transfer function applied on the texture, various modes of density can be observed without having to re-render the data.

Afterwards the texture can be processed in several ways. If the rendition is considered as an density-based representation of the original data, with dense areas opaque and sparse areas transparent, transfer functions can be applied to observe the relative densities of the data without having to actually re-render them (Figure 6.)

This approach was originally introduced by Johansson et al [Johansson et al. 2005]. The growing range of mathematical functions supported by the shader models enables us to provide the user with the option to filter the texture in virtually any arbitrary way.

Moreover, by storing different data segments visualizations in different textures, even the data-oriented actions (selection, deletion etc.) are improved since only the relevant portion of data has to be re-rendered. The remaining textures don't change are only composed one over another to form the final image.

6 Conclusions and Future Work

The presented technical improvement proves that current graphics hardware can be used also for such a specific rendering task as the parallel coordinates display. By exploiting features of the shader models, large GPU-bound memory or parallel processing pipelines we achieved a significant performance boost. The only bottleneck that the technical modifications can not improve is the performance of the actual line rasterization on the GPU.

As the wide variety of GPU functions and features grows with every next generation of the graphics cards, there is much space for extending the concept of hardware-accelerated information visualization. More and more interaction tasks can be performed in parallel on the GPU, leaving the CPU free for non-geometric or more data-oriented operations.

References

CARD, S. K., MACKINLAY, J. D., AND SHNEIDERMAN, B., Eds. 1999. *Readings in information visualization: Using vision to think*. Morgan Kaufmann Publishers, San Francisco.

EICK, S., AND WILLS, G., 1995. High interaction graphics.

FLOREK, M. 2006. *Using Modern Graphics Hardware for Interactive Information Visualization of Large Data*. Master's thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava.

INSELBERG, A., AND DIMSDALE, B. 1990. Parallel coordinates: a tool for visualizing multidimensional geometry. In *IEEE Visualization '90 Proceedings*, IEEE Computer Society, 361–378.

JOHANSSON, J., LJUNG, P., JERN, M., AND COOPER, M. 2005. Revealing structure within clustered parallel coordinates displays. In *INFOVIS*, 17.

KOSARA, R., HAUSER, H., AND GRESH, D. 2003. An interaction view on information visualization. In *EUROGRAPHICS 2003*.

TUFTE, E. R. 1990. *Envisioning Information*. Graphics Press.

WARE, C. 2000. *Information visualization: perception for design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Xmdvtool homepage: http://davis.wpi.edu/xmdv/.