6005

Real time interaction with millions of streamlines

Francois Rheault[1], Jean-Christophe Houde[1], and Maxime Descoteaux[1]
[1]*Université de Sherbrooke, Sherbrooke, Quebec, Canada*

Target audience – Researchers developing streamlines interaction tools, as well as all medical researchers, physicians and clinicians who need to visualize massive tractography datasets.

Purpose – This abstract presents a mechanism to significantly improve real-time visualization and interaction with massive streamlines datasets. This method is able to handle 1M streamlines in a $1mm^3$ space while keeping interactive framerates for rendering and interactive streamlines selection. Trying to use a fiber selection mechanism on such a file in current applications is normally limited and handled by 1 of 3 scenarios: 1) the application restricts the number of streamlines to a specified maximum, 2) the application subsamples the streamlines at loading time (eg. Trackvis [1]) or 3) the application simply stops having responsive framerates (eg. Fibernavigator [2]). The proposed mechanism takes advantage of the streamlines' linearity and allows the user to display and manipulate all streamlines at once. The proposed mechanism is implemented in MI-Brain (www.imeka.ca/mi-brain), a new streamlines visualization and interaction tool based on the MITK platform [3].

Methods – The core of the mechanism is the implementation, at load time, of a streamline linearization step, as was presented in [4]. In this technique, points that are almost collinear, up to a specific error threshold, are summarized by only a few points. Using this step, even with a very small error threshold, can save between 75% and 80% of RAM use, without having any visual impact on the streamlines. Our implementation calculates the distance of each point on the streamline to a straight line, and the points close enough to the line are discarded. Using this mechanism implies that interaction methods, such as the selection box for streamlines selection, must be updated to take the sparse nature of the linearized streamlines into account. Most current methods check if the selection box contains points belonging to a certain streamline to know which fibers are included in the object. With linearized data, the distance between successive points can greatly vary (instead of having a fixed 0.2mm step for example), and the selection box can therefore contain fewer points than usual, which would generate missing fibers. To solve this problem, we added a supplementary segments intersection step to the selection procedure, so that the fibers intersecting the box without having any points inside it will still be detected and displayed correctly. For this to be possible, a range around the actual selection box is used to select neighbouring points, from which segments will be computed. The algorithm then checks if those segments intersect the box, using a line-cube intersection test [5]. To reduce computation time, optimizations and some heuristics were put in place to keep the real-time interaction possible. Such heuristics include putting a limit on the maximal linearization length to prevent a too large neighbour search range and therefore unnecessary intersection testing.

Distance threshold	Maximum distance	Total memory used (MB)	Loading/initialisation time(s)
Uncompressed		6255	48,1
0,01	5	1253	40,2
0,01	10	1187	41,6
0,01	25	1185	42,3
0,02	5	1118	39,6
0,02	10	1046	42,0
0,02	25	1040	42,9

Table 1: Memory usage and time before interaction with different compression parameters for a 500k fibers file.

Results – With a very small linearization threshold, the difference between compressed and uncompressed streamlines is barely visible while, as shown in column 3 of Table 1, the memory used by the application rapidly drops under 20% of the original level. This greatly improves the software performance for visualization, as there are less primitives to send and process on the GPU. The compression slightly increases the initial loading time. However, when taking into account the overall time between loading starts and the moment when the streamlines can be interacted with (which also includes transfers to GPU and shader initialization), it is shown that the complete process is shorter when using linearization with a reasonable threshold (see Table 1, col. 4). Using the adapted selection method is important to ensure no fiber is missed. For example, in Figure 1, a box was put in the corpus callosum of linearized streamlines. The selection technique based only on points displayed 790 fibers, while selection based on points and segments displayed 1290 fibers. In that particular case, almost 40% of fibers would be incorrectly missing. The selection time is not impacted for files with less than 500k fibers. For files with over a million fibers, the selection step can slightly reduce the real-time interactivity of the application. However, since no software can currently load such files, this drawback is acceptable. We successfully loaded, visualized and interacted with a file with 3 million fibers with a 0.5mm step size. In that case, the memory used was only 5.25 GB after linearization with a 0.02mm threshold, when it should have been at least 32 GB without linearization.

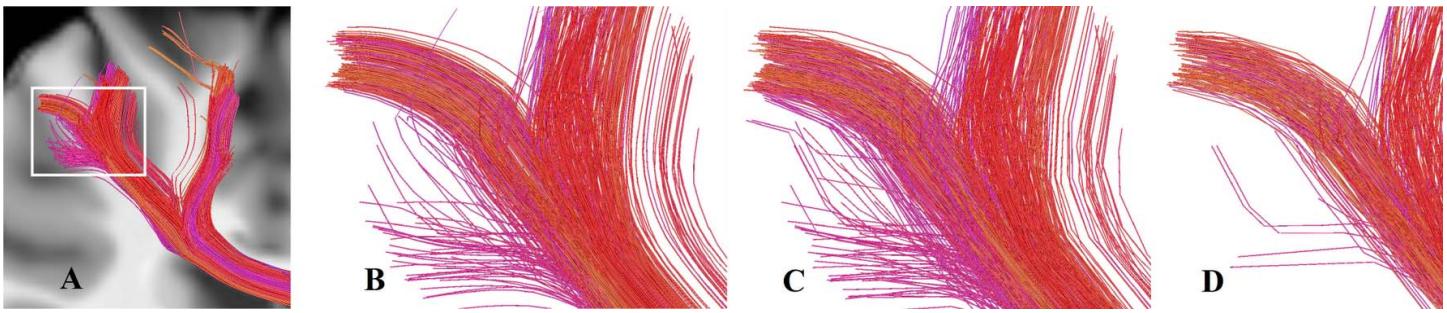

Figure 1: a) subsection of the corpus callosum, white rectangle shows the region that is zoomed; b) uncompressed fibers with classical selection method; c) compressed fibers selected with intersection method; d) compressed fibers selected with classical selection method.

Conclusion – We presented a load-time compression method to greatly reduce the quantity of points in a tractography file; this enables to have real-time visualization and interaction with massive streamlines files (small step size, full brain, millions of fibers) without any perceptible change during visualization. Our method does not restrict what can be loaded by the application and does not affect the original data. We also showed that it is essential to adapt the fibers selection mechanism to guarantee that all fibers crossing the box will be found. In the future, other algorithms will also need to be adapted to correctly handle this kind of compression. Some algorithms could even benefit from the compression since lots of points are removed.

References – [1] www.trackvis.org [2] https://github.com/scilus/fibernavigator [3] www.mitk.org [4] Presseau et al., ISMRM 2014. [5] Pharr and Humprheys, Physically Based Rendering. 2010