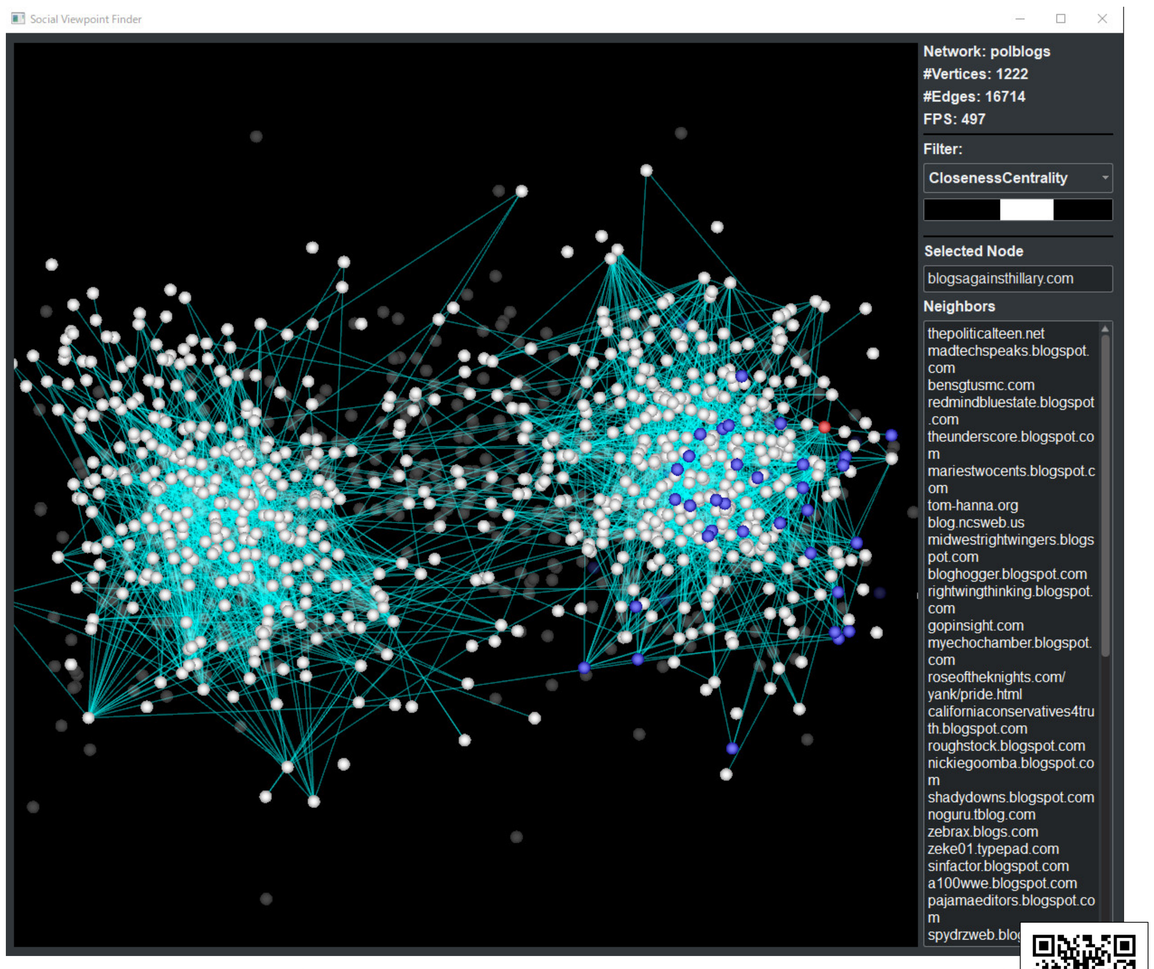
Fluid UI for HIGH-dimensional Analysis of Social Networks

Riku Takano and Ken Wakita, Tokyo Institute of Technology

(https://tknrk.github.io/iui18)

Social Viewpoint Finder (SVF) is a visual analytics tool for social networks and complex networks. SVF lays out the input network data on a HIGH-dimensional (500-3,000 dimensional) Euclidean space, offers a simple, but unique dragging-based UI to trigger computation-intensive, high-dimensional rotation of the presented network, and let the user investigate the clustering structure of the network. This demonstration presents the effectiveness of SVF as well as the employed implementation techniques that enabled the fluid, complex user interaction. To achieve fluidness, SVF heavily relies on modern OpenGL technologies. It achieves massively parallel computing through the use of the compute shader, and graphic pipelines. A large volume of the network data and its layout information is stored in a shader storage buffer. A fragment shader-based, efficient object identification method is devised.



Social Viewpoint Finder (Analysis of the citation network among US political blogs)
This video and more are available on Vimeo. (https://vimeo.com/album/4931455)

Interactions of SVF

Filter

We support several important centrality indices. The user choose a filter based on one of them and the threshold values. Filtered content is painted with faint color.

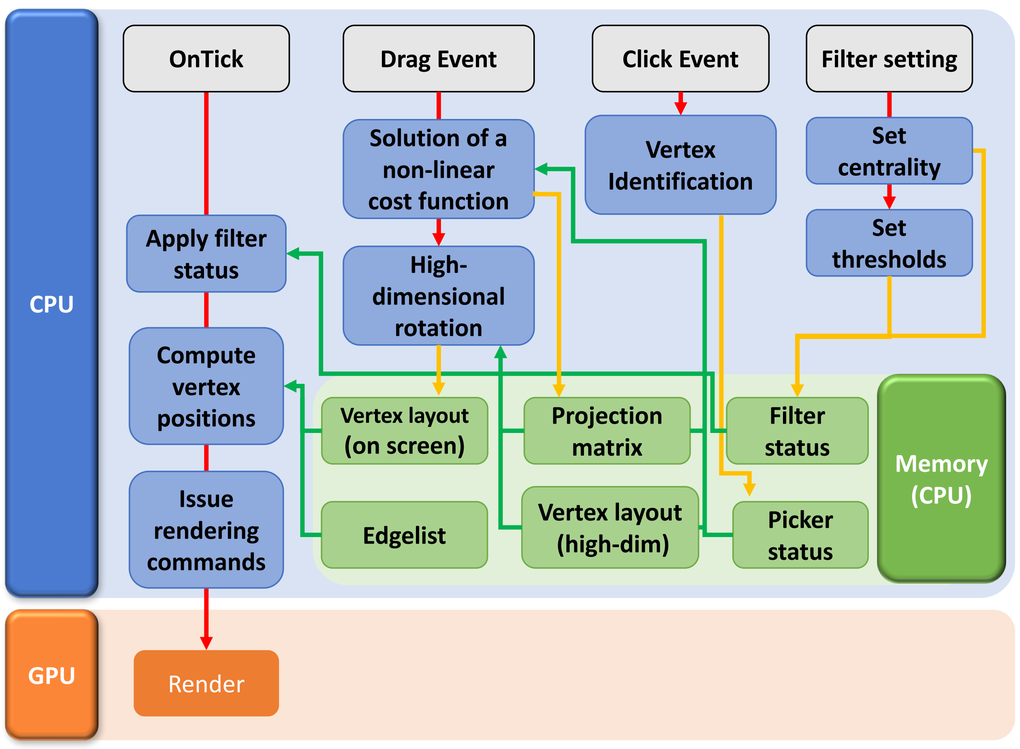
Vertex Identification

Vertex identification conceptually attempts to find the vertex who has the same coordinate as the mouse position. This task is essential to implement UI which provide high-dimensional rotation.

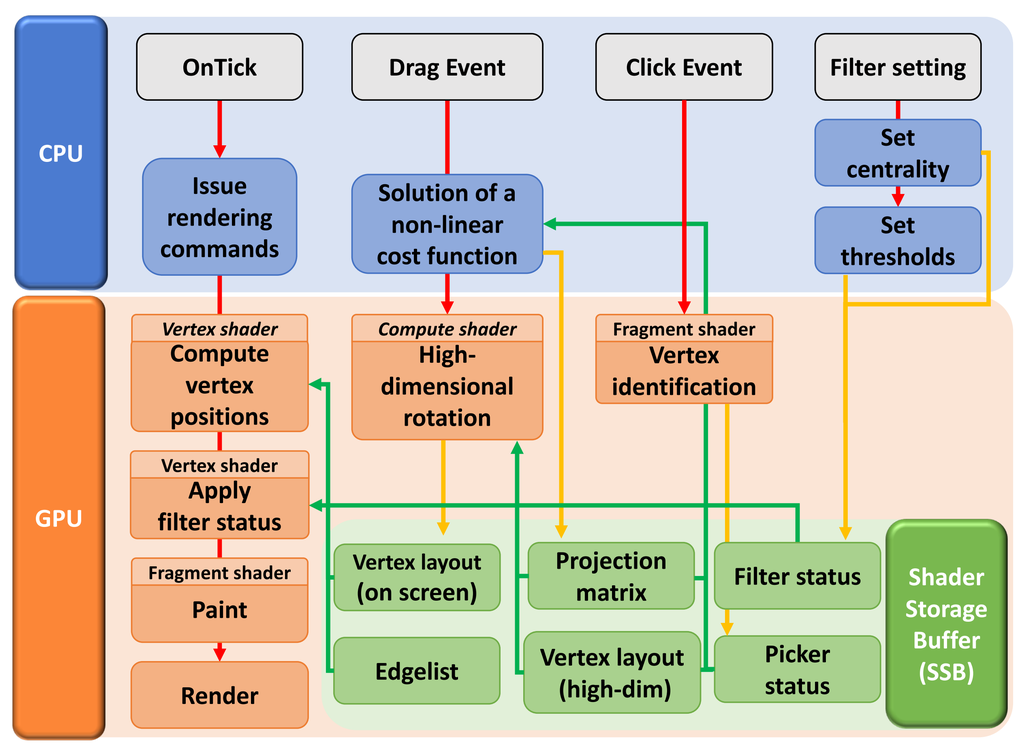
High-dimensional Rotation

High-dimensional rotation of the presented network let the user investigate clustering structure. This computation is heavy-weight task for the computer, but the UI is light-weight for the users.

CPU-based Implementation



GPU-based Implementation (Proposal)



CPU-based Implementation		GPU-based Implementation (Proposal)
All data are allocated to CPU memory. When a rendering command is issued, some data on CPU need to be transferred to the GPU. The latency of data transfer may harm the fluidness of SVF.	**Data Layout**	Most of data are allocated to shader storage buffer, which allows the shaders (and CPU-side application) to share data structures. This data allocation makes the processing of computational tasks of SVF simple and efficient.
This heavy-weight computational task becomes a bottleneck in comfortable visual analytics with SVF. It can be accelerated by using CPU Multi-core and SIMD, but it can not demonstrate sufficient performance to realize comfortable, and fluid UI.	**High-dimensional Rotation**	This is a computation-intensive task but is highly parallelizable one as it is essentially matrix multiplication of a huge matrix that holds the high-dimensional coordinates of the vertices and smaller projection matrix. This computation is performed by the compute shader.
This computation is performed by comparing clicked coordinate with the coordinates of all vertices. To implement this task on CPU-side, we should rely on API for picking up objects provided in a GUI.	**Vertex Identification**	We implemented vertex identification by comparing clicked pixel with the pixels of all vertices. This way enables us to intuitively implement parallelized vertex identification in the fragment shader.
A filter is applied at each rendering frame. An application of filtering per a vertex (or an edge) is light-weight computation, but the amount of vertices and edges is so large, and the application of filter is performed frequently: this task may harm the fluidness of SVF.	**Filter**	A filter is applied at each rendering frame. This computational task is parallelizable one. This computation is performed in the graphics pipeline, because the computation is related to the rendering process.

Benchmark

Data (#vertices, #edges)	CPU-based Drag [millisec]	CPU-based Render [millisec]	CPU-based SVF [fps]	Proposal Drag [millisec]	Proposal Render [millisec]	Proposal SVF [fps]
USPol (1,222, 16,714)	1	57	57	3.01	0.01	331.1
Math (3,608, 48,315)	5	30	26	3.35	0.06	293.3
Internet (22,463, 48,436)	193	15	3.8	3.37	1.0	228.8
Enron (33,696, 180,811)	379	7.5	1.9	3.35	2.8	162.6

For each dataset, the following are given: execution time per one drag event (Drag), execution time per one rendering on graphic pipelines (Render), average frames per second (FPS).
("USPol": Citation network among US political blogs, "Math": References network in Mathematics related Wikipedia pages, "Internet": BGP network., "Enron": Corporate-wide email message exchange network)

Summary

A modern OpenGL-based implementation succeeded in making complex interactions of SVF fluid and providing users with comfortable and efficient network analysis tool. We realized the combination of a heavy-weight computational task on the computer side and light-weight operating task on the human side: in one aspect, we implemented ideal visual analytics tool.

Future Works

It is our future work to sophisticate the UI of SVF in order to help the user to analyze complex networks more comfortably and efficiently.

Reference: [1] (Wakita, Takami, Hosobe 2015) Interactive high-dimensional visualization of social graphs, PacificVis 2015, DOI:10.1109/PACIFICVIS.2015.7156391.
[2] (Hosobe 2014) A high-dimensional approach to interactive graph visualization, SAC 2014, DOI: 10.1145/967900.968155.