PathlinesExplorer
Image-based Exploration of Large-Scale Pathline Fields

Omniah H. Nagoor†
KAUST
Markus Hadwiger ‡
KAUST
Madhusudhanan Srinivasan‡
KAUST

ABSTRACT
PathlinesExplorer is a novel Image-based tool, which designed to visualize large scale pathlines fields on a single computer [8]. PathlinesExplorer integrates Explorable images (EI) technique [4] with Order-Independent Transparency (OIT) method [2] and [9]. What makes this method different is that it allows users to handle large data on a single workstation. Although it is a view-dependent method, PathlinesExplorer combines both exploration and modification of visual aspects without re-accessing the original huge data. Our approach is based on constructing a per-pixel linked list data structure in which each pixel contains a list of pathlines segments. With this view-dependent method, it is possible to filter, color-code, and explore large-scale flow data in real-time. In addition, optimization techniques such as early-ray termination and deferred shading are applied, which further improves the performance and scalability of our approach.

Keywords: Scientific visualization, visualization toolkit, image-based rendering, per-pixel linked list, pathlines fields, explorable images, deferred shading, early-ray termination.

Index Terms:
I.3.6 [COMPUTER GRAPHICS]: Methodology and Techniques—Interaction techniques, Graphics data structures and data types, and Languages; I.3.3 [COMPUTER GRAPHICS]: Picture/Image Generation—Display algorithms, Bitmap and framebuffer operations, Line and curve generation, and Viewing algorithms; H.5.2 [INFORMATION INTERFACES AND PRESENTATION]: User Interfaces—Graphical user interfaces (GUI).

1 INTRODUCTION
While real-time applications are nowadays routinely used in visualizing large numerical simulations and volumes, handling these large-scale datasets requires high-end graphics clusters or supercomputers to process and visualize them. However, not all users have access to powerful clusters. Therefore, it is challenging to come up with a visualization approach that provides insight to large-scale datasets on a single computer. Explorable images (EI) is one of the methods that allows users to handle large data on a single workstation. In this work, we propose a novel image-based exploration method that allows end-user to explore 3D pathlines fields [1] with reasonable computing resources. Figure 1 illustrates a collection of different functionalities presented by our work.

2 METHODOLOGY & WORKFLOW
PathlinesExplorer has mainly two stages, which are the pre-processing (offline stage) and real-time (interactive stage). The first step is to select a camera position that has a global coverage of the 3D pathlines data. Then, the pre-processing stage starts by constructing a per-pixel linked lists and stores it in OpenGL 4.3 buffers (Shader Storage Buffer Objects SSBOs) according to their correct depth order. This is done similarly to the implementation of the OIT technique. The only difference is in the rendering pass. Usually, OIT is implemented at real-time. However, in our approach we follow the same implementation in terms of building the linked list data structure and sorting it. By combining these two processes as a pre-computation to our image-based approach, we provide the user with fast and interactive interface to all the data in the linked list structure. At the real-time stage, the compositing of final color is applied at each pixel. This is achieved by applying the blending equation [6], which is implemented as a front-to-back compositing scheme:

\[ C_{dest} = C_{src} + (1 - A_{dest})C_{src} \]

\[ A_{dest} = A_{src} + (1 - A_{dest})A_{src} \]

Where C represents the color component, which has three-color channels values \( C \in \{R,G,B\} \). The compositing process is different for each pixel since each pixel has a different number of linked list elements. Therefore, each pixel needs to be traversed independently to the other pixels and with respect to the number of depth layers (number of linked list elements) it contains. By applying front-to-back compositing, it is possible to optimize the traversing process with some techniques such as early-ray termination.

Figure 1: A collection of different functionalities presented by our approach for rendering 25000 pathlines. (a) The pathlines are color-coded by Temperature scalar. (b) The pathlines are filtered by time [0,200] and color-coded by Mixture fraction scalar. (c) The pathlines are color-coded by Temperature scalar with some thresholds.

Figure 2: General overview of our implementation diagram.
3 Result & Analysis

In our system evaluation, we attempt to tackle different measuring parameters to evaluate different stages of our approach. This involves evaluating the pre-processing phase with its sub-processes and the real-time phase. In the pre-processing phase, we measure the time required to complete all the sub-processes as well as the total memory consumption that it requires. However, for the real-time phase we measure the rendering speed, which mainly relies on the number of frames per second (FPS).

3.1 Different Viewports

Using different viewpoints imply different computation time and memory size as it is shown in Figure 3. Since each viewpoint has different position and direction to the given dataset, this yields different number of depth layers (the number of linked list elements) that the viewpoints cover. Therefore, each viewpoint requires different computation time and buffer size as well. Figure 4 illustrates rendering the depth layers (the number of linked list elements) of different viewpoints.

An interesting case of the pre-processing results is introduced by view 4, which causes the least computation time as well as memory size. This is due to the direction of lines field, which is parallel to the viewpoint direction. Such view angle leads to a problem in terms of lines sampling - most of the lines are skipped or sampled with a bad sampling rate.

3.2 Early-Ray Termination (ERT)

Applying early-ray termination causes a high reduction in the number of depth layers that need to be processed as it shown in Figure 5. Therefore, it is expected that this method will outperform the overall quality since only a few depth layers need to be rendered. Moreover, it will become independent of the dataset size as it illustrated in Figure 6.

Figure 3: This figure shows the (a) pre-processing computation time as well as (b) the buffer sizes required for rendering 25000 pathlines dataset with different viewpoints.

Figure 4: Illustration of depth images (number of linked list elements) 25000 pathline dataset from the four different viewpoints. The thumbnail images show how the dataset oriented and scaled in the viewport.

Figure 5: Comparing the rendering depth images (number of linked list elements) of 25000 pathlines dataset of the same view without ERT (a) and with it (b).

4 Conclusion & Future Work

The goal of PathlinesExplorer is to provide a novel image-based tool for pathlines visualization. This includes a number of different functionalities, which serve the user queries through image-space in real-time. Filtering, coloring, and exploring are some examples of these functionalities. The reduction from the 3D space to image space enhances the performance in terms of memory consumption as well as portability. Using our proposed method makes it possible to visualize large-scale data on a single workstation without the need to re-access the original data. The way our work is implemented, it utilizes the power of GPU by applying the per-pixel linked list method with shader storage buffer objects (SSBOs).

Further improvement could be achieved by applying screen tiling and memory paging for the buffers. This involves segmenting the depth layers (number of linked list elements) into different segments. Each is mapped to its corresponding screen tile. Then, it loads these segments (caching technique) regarding to the applied filters. Also, another enhancement could be done by using an algorithm that investigates the optimal viewpoints, which can uniquely illustrates the flow field and have minimal average depth layers size (number of linked list elements). Moreover, enhance the quality of the rendered lines by applying some shading techniques as the one presented by Mallo et al. [7] or LineAO method by Eichelbaum et al. [3]. Also, combine our work with other optimization methods for lines selection, namely, Opacity Optimization method provided by Günther et al. [5].

References