Computers & Graphics 52 (2015) 79–92

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Extracting flow features via supervised streamline segmentation

Yifei Li [a,*,1], Chaoli Wang [c,3], Ching-Kuang Shene [b,2]

[a] Department of Computer Science, Michigan Technological University, 1400 Townsend Drive, Houghton, MI 49931, United States
[b] 305 Rehki CS Hall, Department of Computer Science, Michigan Technological University, 1400 Townsend Drive Houghton, MI 49931, United States
[c] 384 Fitzpatrick Hall, Department of Computer Science & Engineering, University of Notre Dame Notre Dame, IN 46530, United States

ARTICLE INFO

Article history:
Received 22 January 2015
Received in revised form
9 June 2015
Accepted 9 June 2015
Available online 26 June 2015

Keywords:
Flow visualization
Flow feature extraction
Streamline segmentation
Support vector machine

ABSTRACT

Effective flow feature extraction enables users to explore complex flow fields by reducing visual clutter. Existing methods usually use streamline segmentation as a preprocessing step for feature extraction. In our work, features are directly extracted as a result of streamline segmentation. In order to achieve this, we first ask users to specify desired features by manually segmenting a few streamlines from a flow field. Users only need to pick the segmentation points (i.e., positive examples) along a streamline, remaining points will be used as negative examples. Next we compute multiscale features for each positive/negative example and feed them into a binary support vector machine (SVM) trainer. The trained classifier is then used to segment all the streamlines in a flow field. Finally, the segments are clustered based on their shape similarities. Our experiment shows that very good segmentation results can be obtained with only a small number of streamlines to be segmented by users for each data set. We also propose a novel heuristic based on the minimum bounding ellipsoid volume to help determine where to segment a streamline.

1. Introduction

Flow visualization has been a central topic in scientific visualization for more than two decades. A flow field can be visualized using different techniques, including glyph-based [1], texture-based [2], integration-based [3], partition-based [4], illustration-based [5], and surface-based [6] approaches. Among these techniques, integration-based flow visualization is most widely used in practice. For integration-based flow visualization, particles or seeds are placed in a vector field and advected over time. The traces or field lines that the particles follow, i.e., streamlines for steady flow and pathlines for unsteady flow, depict the underlying vector data. In this paper, integration-based technique with random seeding is used for visualization.

However, visual clutter and occlusion is a major issue when hundreds or thousands of streamlines are rendered to depict a flow field. This makes it difficult for users to explore the interesting features. Although clustering and displaying the streamlines based on their similarities may alleviate this issue, a more subtle issue is that very often not all parts of a streamline are equally important: the part of a streamline in the vicinity of a vortex is more important than the part running through a region of laminar flow. Furthermore, different domain experts may have their own criteria on what constitutes an "interesting flow feature". This observation inspired us to segment a streamline based on *user-defined* features. To the best of our knowledge, this problem has not been well studied by the flow visualization community.

To address this problem, we propose a supervised streamline segmentation algorithm which allows the extraction of user-defined flow features. For each data set, users are required to manually segment only a small number of streamlines in order to define what flow features they want to extract from the flow field. The user-picked segmentation points along a streamline will be used to generate *positive* training examples, whereas the remaining ones are used to generate *negative* training examples. Multiscale feature vectors are computed for each positive and negative example, and fed into a binary support vector machine (SVM) trainer. Finally, we use the trained classifier to determine the segmentation points for all the streamlines in the data set. A post-processing step is required for grouping nearby segmentation points detected by the classifier. Fig. 1 shows an example of extracting user-defined partial flow features.

The remainder of this paper is structured as follows. Section 2 reviews approaches that are most related to our work. Section 3 presents the details of our algorithm and is divided into the

* Corresponding author.
E-mail addresses: yifli@mtu.edu (Y. Li), chaoli.wang@nd.edu (C. Wang), shene@mtu.edu (C.-K. Shene).
[1] Tel.: +1 906 487 2209; fax: +1 906 487 2283.
[2] Tel.: +1 906 487 3392; fax: +1 906 487 2283.
[3] Tel.: +1 5746319212; fax: +1 5746319260.

http://dx.doi.org/10.1016/j.cag.2015.06.003
0097-8493/

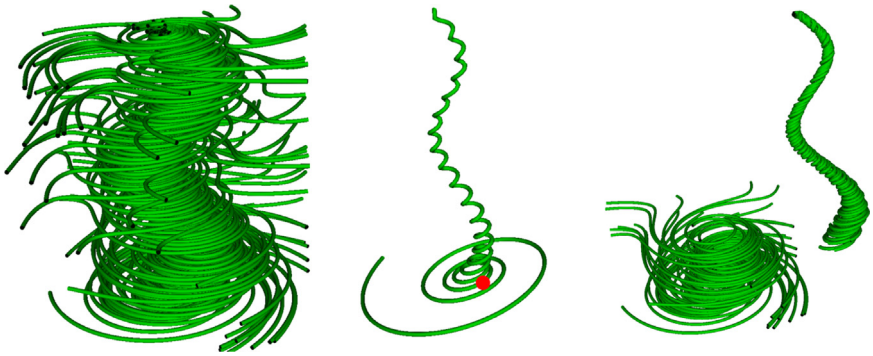

Fig. 1. Given an input pool of streamlines (left), we first segment each streamline using our previously learned classifier for segmentation points (middle, the red point is the segmentation point found by our algorithm). Partial streamline features specified by users will be clustered based on their similarities (right). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

following subsections: Section 3.1 introduces basic concepts of supervised learning and SVM, Section 3.2 discusses the features we use to train a classifier, Section 3.3 explains how the positive and negative examples are generated, Section 3.4 explains how the training is conducted, and Section 3.5 introduces our segmentation algorithm and necessary post-processing steps. Section 4 demonstrates the utility of our algorithm by clustering streamline segments using different 3D flow fields. Section 5 compares our method with a few other state-of-the-art streamline segmentation/feature extraction methods. Finally, Section 6 points out the directions of our future work.

2. Related work

Flow feature extraction provides an effective way to reduce visual clutter. For 2D flows, Schlemmer et al. [7] and Bujack et al. [8] both leveraged moment invariants to detect 2D flow features. Wei et al. [9] relied on user-sketched 2D curves to retrieve similar occurrences from a 3D flow field. The retrieval might be ambiguous because the 3D streamlines first need to be projected to 2D curves for similarity comparison. Tao et al. [10] converted each streamline into a string such that all the streamlines can be recorded in a suffix tree. The string patterns detected in the suffix tree correspond to certain flow features. Users can specify a query string to search for interesting flow features. Finally, Wang et al. [11] proposed an example-based flow pattern search approach for the detection of similar flow feature patterns given a query pattern, where flow patterns are given by a subset of segments from the set of all streamline segments.

Streamline clustering and selection provides another way to reduce visual clutter. Common clustering algorithms such as nearest neighbor, fuzzy clustering and hierarchical clustering have been used in the works of [12–14]. Different streamline similarity measures have been proposed for streamline clustering. Examples include the average of point-by-point distance [13], the mean of closest point distances [15] and the thresholded average distance [12]. These similarity measures are all based on the Euclidean distance, and hence are not affine invariant. To overcome this shortcoming, similarity measures based on feature distribution were adopted in the works of [16–18]. For a detailed survey of streamline clustering methods and similarity metrics, we refer readers to [19]. Günther et al. [20] rendered streamlines using different opacity values which are computed to optimize the balance between information presentation and occlusion avoidance. They also gave a comparison among the different state-of-the-art streamline selection algorithms.

Streamline segmentation has been used to facilitate streamline similarity comparison [17] and flow pattern extraction [10,11]. Lu et al. [17] recursively segmented a streamline into two most dissimilar segments until either the dissimilarity is below a certain threshold or the current segment is too short. Tao et al. [10] segmented a streamline such that the accumulated curvature of each segment does not exceed a certain threshold. An obvious drawback of their approach is that they failed to separate straight segments. Wang et al. [11] partitioned a streamline into so-called minimal segments first, and the final segmentation is obtained after merging the minimal segments based on two thresholds: total curvature and average binormal direction. However, their approach cannot segment turbulent streamlines very well. All of these segmentation algorithms need some manually tuned thresholds to determine whether to segment or not at a given point.

The problem of curve segmentation has also been studied in the computer vision community. However, they usually focused on segmenting a curve into a combination of representations such as lines, circular, elliptical and superelliptical arcs, and polynomials [21]. We cannot apply their methods to streamline segmentation because we are usually interested in more complicated features such as spirals rather than just, for example, circular curves.

The *minima rule* [22] from cognitive science is widely used by mesh segmentation algorithms [23]. The rule states that human beings tend to divide a *surface* into parts at loci of negative minima of each principal curvature along its associated family of lines of curvature. However, after extensive research, we have not found any rules from cognitive science which can help us segment a 3D curve (e.g., streamlines).

3. Supervised streamline segmentation

In order to obtain a streamline segmentation based on user-defined features, we leverage supervised learning to train a classifier to determine whether we should separate a streamline around a given point. The motivation for using machine learning is that we want to take multiple features into consideration when determining whether to segment and generate a complex decision function based on those features.

Therefore, we propose a user-guided streamline segmentation framework, which is illustrated in Fig. 2. After streamlines are traced, we cluster (Section 3.3.1) and simplify (Section 3.3.2) all the streamlines. From cluster representatives, users choose which streamlines to segment manually. A binary SVM classifier (Section 3.1) is trained (Section 3.4) to classify segmentation points of a streamline. A post-processing step (Section 3.5) is carried out to generate final segmentation. The training process (the dashed line in Fig. 2) can be repeated if users are not satisfied with the segmentation results.