8-2006

Automatic Image Based Time Varying 3D Feature Extraction and Tracking

Lu Zhang
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Lu Zhang entitled "Automatic Image Based Time Varying 3D Feature Extraction and Tracking." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Hairong Qi, Major Professor

We have read this thesis and recommend its acceptance:

Jian Huang, Michael J. Roberts

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Lu Zhang entitled "Automatic Image Based Time Varying 3D Feature Extraction and Tracking". I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Hairong Qi
Major Professor

We have read this thesis
and recommend its acceptance:

Jian Huang

Michael J. Roberts

Accepted for the Council:

Anne Mayhew
Vice Chancellor and
Dean of Graduate Studies

(Original signatures are on file with official student records.)

Automatic Image Based Time Varying 3D Features Extraction and Tracking

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Lu Zhang
August 2006

To my dear parents for their loves

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Hairong Qi, for her unwavering support over the time when I am working in the AICIP (Advanced Imaging and Collaborative Information Processing) Laboratory. The association with Dr. Qi has been inspiring and a great learning experience. If not for her, this thesis work and the work behind it would have not become a reality. I would like to take this opportunity to thank Dr. Jian Huang and Dr. Michael J. Roberts for serving on my thesis committee and for providing me valuable suggestions on my thesis work.

I thank our collaborators at the SeeLab of the Department of Computer Science, UT for their precious suggestions and support in this project. I especially thank Dr. Huang for introducing me into this field, and my co-worker Colin to help me collect the raw data sets, and give a lot of good advice on this thesis.

I would like to give my special thanks to my labmates at the AICIP lab. Chris, Dayu, Hongtao, Lidan, Ortal, and Raghul- thank you for all your support.

I am extremely grateful to all my friends at UT, who have given me wonderful life in the USA for the past two years. Most special thanks to Scott, who supports my work and shows great patience on me.

I dedicate this thesis to my parents Rongxue Zhang and Aijun Fang, they show me the most glorious love in the world, and support and encourage me to persevere through life's challenges. They have been my idol and I am so proud of being their daughter.

Abstract

3D time-varying data sets are complex. The intrinsics of those data cannot be readily comprehended by users solely based on visual investigation. Computational tools such as feature extraction and tracking are often necessary. Until now, most existing algorithms in this domain work effectively in the object space, relying on prior knowledge of the data. How to find a more flexible and efficient method which can perform automatically to implement extraction and tracking remains an attractive topic.

This thesis presents a new image-based method that extracts and tracks the 3D time-varying volume data sets. The innovation of the proposed approach is two-fold. First, all analyses are performed in the image space on volume rendered images without accessing the actual volume data itself. The image-based processing will help to both save storage space in the memory and reduce computation burden. Secondly, the new approach does not require any prior knowledge of the user-defined "feature" or a built model. All the parameters used by the algorithms are *automatically* determined by the system itself, thus flexibility and efficiency can be achieved at the same time.

The proposed image-based feature extraction and tracking system consists of four components: feature segmentation (or extraction), feature description (or shape analysis), classification, and feature tracking. Feature segmentation is to identify and label individual features from the image so that we can describe and track them separately. We combine

both region-based and edge-based segmentation approaches to implement the extraction process. Feature description is to analyze each feature and derive a vector to describe the feature such that the subsequent tracking step does not have to rely on the entire feature extracted, but instead a much smaller and informative feature descriptor. Classification is to identify the corresponding features from two consecutive image frames along both the time and the spatial domain. Feature tracking is to study and model the evolution of features based on the correspondence computation result from classification stage. Experimental results show that the image-based feature extraction and tracking system provides high fidelity with great efficiency.

Contents

List of Figures

Chapter 1

Introduction

First, let's imagine we are traveling by airplane and now we are 30,000 feet high in the air. Take a few minutes to look out of the window. Do you notice the clouds around you? Can you visually separate them from the blue sky and from each other? Can you tell where they are going and what kind of movements they make? Yes, of course you can and you do so with ease.

This is a very simple task for human beings, though actually requires the cooperation of your eyes with billions of neurons in your brain. Light reflects from the surface of clouds and back into your eyes striking each retina. Rods and cones in the retina react to the stimulation from the light, translate it into neural signals which are transmitted into your brain, the most sophisticated "computer". After thousands of complicated operations, our brain tells us what we want.

Now, imagine that scenario as a *computer engineer*. How can we use a computer to implement the same thing—to separate each cloud from the blue sky and track their movements, as our brain does for us? In a very focused context, the answer to this question is the research goal of this thesis. In particular, we have developed a software system to obtain a

similar result using the advanced *image processing*, *computer vision*, and *pattern recognition* algorithms. The goal of the proposed algorithms is to mimic the processing procedures in our brain and give similar output.

3D time-varying feature extraction and tracking is a very popular topic in the field of *Computer Vision*. It has wide spectrum of applications, including Scientific Analysis, 3D Reconstruction, Virtual Reality, Visualization, etc. The 3D feature tracking literature is particularly extensive because there are almost as many approaches as target applications, and for the same problem, different algorithms can be adapted. In this thesis, we will focus on image-based automatic algorithms rather than 3D volume-based methods which will be reviewed later.

1.1 Motivation

Targeting and tracking 3D time-varying fluid data sets is difficult because of the immense amount of data to be processed and understood. This kind of data contains many evolving amorphous regions which are difficult for humans to observe. However, scientists commonly use such data sets in a wide spectrum of applications, ranging from automotive, aerospace, and turbomachinery design, to weather simulation and meteorology, climate modeling, and medical applications, with many different research and engineering goals and user types. Once the evolution events have been captured, scientists can attempt to understand the underlying cause and build the mathematic models. For example, just like we discussed before, if we can track the clouds in a certain area and build a predictive model for them, we can predict their movement in the next several hours or days.

Much existing literature has attempted to solve the problem in the 3D object space [2, 26, 35]. However, the difficulty in dealing with 3D time-varying fluid data sets is the overwhelming amount of data. Other techniques have also studied the problem in the 2D image

space by tracking the boundaries of the 2D regions [16]. This technique is not sufficient to fully define the 3D features. In this thesis, we aim to explore this problem in the 2D image space. In addition to finding the boundaries of the 2D regions, we also identify individual features and track their movements across a sequence of time-varying data sets.

1.2 Literature Review

In the computer vision field, reconstruction or visualization techniques provide tools that help researchers identify observed objects or phenomena in either real world situations or in scientific simulation. To be useful, these tools must allow the user to extract regions, classify and visualize or reconstruct them, abstract them for simplified representations, and track their evolution or movement. Usually, researchers have to deal with a large amount of flow-based sequence data in 2D or 3D space. For example, the total size of the original 3D time-varying fluid data set used in this research is about sixteen Gigabytes, and uses only one hundred time steps. However, feature-based approaches can greatly ease the process of investigating large data sets efficiently. The original input data is described by basic robust intrinsic parameters, which represent the interesting objects or features in the data. Therefore, the original data set may no longer be needed. If the researchers can find some features representing the original data sets very well and the features can be described very compactly, an enormous data reduction can be achieved. Therefore, *Feature Extraction and Tracking* plays an important role in the research in computer vision and visualization. Two steps are involved in this feature-based approach: *feature extraction* and *feature tracking*.

In computer vision field, the visualized motion can be categorized into two general groups: the rigid motion and the non-rigid motion. Rigid motion has been studied intensively and a large collection of approaches have been developed because of its simplicity and elegance. However, recent research on non-rigid structure and motion visualization has

drawn more attention since, in the real world, non-rigid motion of objects is more common. Non-rigid motion can be generally classified into three groups: the motion of rigid parts, the motion of coherent objects, and the motion of fluids [1]. Motion of rigid parts occurs in situations where individual rigid parts of an object move independently of one another, which means that the motion of each constituent part is rigid, but the motion of the whole object is non-rigid. On the other hand, non-rigid motion of coherent objects includes motions such as the motion of a heart, the waving of a cloth, or the bending of a metal sheet, where the shape of the object deforms under certain constraints. Most studies conducted in the above two categories have been focusing on tracking single object which may be composed by different sub-objects. Only recently, has this research been extended to multiple objects [11, 12]. The motion of fluids is the most complex case. There are several problems existing in this category: unlike solids where mass and the amount of force exerted on the mass controls the motion, fluids move in response to density and pressure. Additionally, fluids contain topological variations and turbulent deformations which add another degree of complexity. In this thesis, we will mainly discuss feature-based techniques applied to track the motion of fluids.

We can also classify different tracking approaches based on whether or not the prior models are used. Obviously, automatic feature extraction and tracking is much more difficult than approaches using a prior model, especially when the original data sets include noise or ambiguities, and in the case of fluid motion, objects are continually evolving and interacting. Automatic feature tracking is necessary in cases where no prior models are available, or existing models are too restricted and would not be able to recover all objects. Though the usage of models is more robust, it is also limited when solving real-world problems.

1.2.1 The Motion of Rigid Parts

The motion of rigid parts can be categorized as *articulated* motion [15]. First, approaches without using prior models will be introduced; then we will turn to model-based tracking methods.

In [16], Kurakake located the joints of articulated objects through motion using extracted ribbons. Small motion between the consecutive frames was assumed to simplify the correspondence problem. Raw image sequences were input into the system and part descriptions without depth information being produced. In [15], the researcher used 2D contours as shape representations for segmentation and motion estimation. Joint locations were detected as the center of the overlapping area of two connected contours. In [22], Qian and Huang analyzed the articulated object as a collection of rigid parts linked by points. After decomposing the whole structure into several subparts and estimating the articulated motion, they also set some constraints, such as coplanar motion, fixed axis, and at least one known joint, to aid the reconstruction procedure.

In feature extraction and tracking, well defined constraints and prior models are estimated and imposed first to eliminate invalid feature matches and distinguish unique connections. In the early research [23], correspondence was established by minimizing the difference between the expected position of each point in the previous frame and the actual position of the corresponding point in the following frame. The prediction of the point position was based on the observation that the velocity of the rigid parts, such as human bodies, varied smoothly, while the underlying structure was relatively fixed. Then, approaches based on 2D contours were proposed by Kurakaka [16], in which he classified ribbons and their symmetric axes into layers in terms of their widths. Feature correspondence and tracking were implemented by Kakadiaris [13]. In his paper, he calculated the correspondence in the process of accumulating knowledge of the 2D deformable model,

which was derived solely from previous images. An object was initially defined as a single part according to the prior model. After being tracked in several successive frames, the object was divided into the fixed zone, the bending zone, and the relocation zone. Hence, the model deformed into different part models accordingly.

For 3D data sets, tracking becomes more complicated due to complicated movement of 3D structures. Similar to the points in the 2D space, vertices are used as the primitive for feature matching. Based on prior knowledge of constraints such as shape rigidity, planarity, and small inter-frame motion, the algorithm produces groups of three or more points in each group. For selecting the proper correspondence set and eliminating mismatched sets, the criterion for determining the optimal match is the sum of the square error of all possible attributes. In the last stage, the global consistency of point sets is guaranteed throughout the whole image sequence, eliminating the possibility of certain alignment.

On the other hand, model-based approaches can overcome the difficulties encountered in feature correspondence if we know the object shape. Take human body movement analysis as an example. We can use either a stick figure or a volumetric model to represent a human's body. Some of the algorithms [4] trained a human body model by firstly using sensors embedded on real human body and then tracking them down to see which features can actually be treated as point features. Other papers [17] used a proposed 2D ribbon model containing two components: the basic body model and the extended body model. Some also used elliptical cylinders to model the human forms. In summary, to gain prior knowledge about the model is the key in model-based algorithm design. If the object can be represented by a reasonable set of sub-objects, to obtain a robust estimation of movement and then visualize this model is not a difficult task.

1.2.2 The Motion of Coherent Objects

In the domain of general deformable motion, most approaches extract and track features based on an existing model and then try to model the deformations as variations to the model parameters. The correspondence problem is simplified as the incorporation of shape models significantly reduces the search space in the feature matching process. In some cases, the correspondence problem is transformed to a parameter estimation problems. Using physically-based modeling primitives has the added advantage of resolving visualization problem [29]. These methods are most effective when prior knowledge about the motion structure is available. However, if the characteristics of the motion are not known, model-based methods could cause errors in motion estimation and visualization. In such cases, approaches that assume no prior knowledge about the motion or object structure can first be utilized to determine the general properties of the non-rigid motion, then one can select the proper modeling primitives to refine or enhance the analysis. Therefore, if these two general approaches can be integrated reasonably, we can achieve a more reliable scheme for analyzing general non-rigid motion.

The feature extraction and tracking stage for coherent objects such as human hearts, the waving of a cloth, and the bending of a metal sheet, is usually the primary step in visualization and reconstruction. Unlike situations in which the rigidity assumption is valid, when deformation occurs, image features that bear reliable structural and geometric information about the objects usually do not exist. This significantly limits the type of matching primitives that can be used to analyze the non-rigid motion. In general, two levels of features have been employed: low level features such as points [9, 18], and high level features such as contours [5].

Points features are widely used because, in some cases, they are the only type of feature type that can be found, and are invariant with respect to deformable transformation. Since

points are low-level representations, the accuracy of the matching results depends heavily on the validity of the motion smoothness assumption. Contour features are high-level representations. They provide richer information than collections of points. Contours also contain structural information about the object, thus it can supply more robust results than points features. The major problem, however, is that the extraction of reliable contours or surfaces from a sequence of non-rigid shapes is not an easy task itself. In some cases, the contours could not represent the real shapes of objects.

There are two general approaches for feature correspondence in non-rigid motion analysis: explicit and implicit matching methods. In explicit matching [5, 18], the finite set of image features extracted from image sequences is examined and compared to arrive at one-to-one mappings based on certain likelihood measurements. In implicit matching [9], distinct features are not isolated. Instead, an energy function incorporating internal forces (e.g., smoothness constraints) and external forces (e.g., feature energy) is formulated and the feature tracking problem is converted to an optimization problem.

1.2.3 The Motion of Fluids

Compared with the two groups introduced above, the visualization and reconstruction of fluid objects has offered great challenges for research. In many practical cases, researchers have to process very large data sets which consist of multivariate data at large numbers of grid points in many time steps and at different space positions. Due to the complexity and uncertainty of the problem itself, the spectrum of fluid visualization techniques is very rich, spanning multiple dimensions of technical aspects, such as 2D and 3D techniques, and techniques for processing steady and time-dependent data. The state-of-the-art in fluid visualization techniques can be categorized into four groups [21]: direct flow visualization, texture-based flow visualization, geometric flow visualization, and feature based

visualization. In this section, we will present techniques used mainly in feature based flow visualization. Among those four categories of algorithms, both feature based algorithms and geometric algorithms are high level methods which first represent the raw data sets by extraction methods, and subsequent analysis and processes are conducted on the abstraction data instead of the original large data sets.

In *direct flow visualization* [21], the data is directly visualized, without much preprocessing, for example by color coding or marking arrows. These techniques are also called global techniques, as they are usually applied to the entire domain, or a large part of it.

In *textured base flow visualization*, visualization is directly applied based on the texture property of the data sets. This kind of visualization is usually in two dimensions or on the surfaces. As in the visualization of a whirlpool of water flow. This group has been tightly integrated with the feature-based flow visualization methods tightly in the past several years because, in some practical cases, people extract the texture property of the data sets as one robust feature to recognize and reconstruct their three-dimensional shapes.

Geometric flow visualization integrates the flow data and uses geometric objects as a basis for flow visualization. The resulting integral objects have a geometry that reflects the properties of the flow, for example, the streamlines, streaklines, and pathlines used when trying to visualize the movement of water flow.

Feature-based flow visualization is an approach for visualizing the flow data at a high level of abstraction. The flow data is described by features, which represent the interesting objects or structures in the data. The original data set is then no longer needed, because often, only a small percentage of the data is of interest and the features can be described very compactly and an enormous amount of data reduction can be achieved. This makes it possible to visualize even very large data sets interactively. The first step is feature

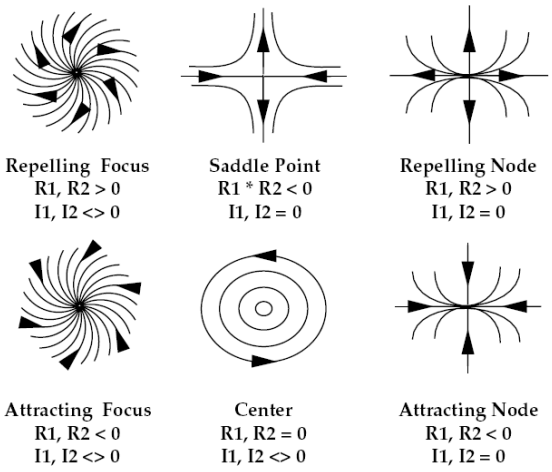
extraction. The goal is to determine, quantify, and describe the best representative features in the data set. The definition of features varies, including vertices, points, separation and attachment lines, boundaries, etc. Due to the complexity of individual cases, most feature detection techniques are specific for a particular type of feature which can be applied only in a certain case. In general the techniques for feature extraction can be divided into three approaches: image-based processing, topological analysis, and physical characteristics [21]. The algorithms proposed in this thesis are based on image processing methods. In the following, we review different techniques used for feature extraction and feature tracking, respectively.

1.2.4 Feature Extraction Algorithms

In this section, we summarize the state-of-the-art feature extraction algorithms, including methods based on topology analysis, physical characteristics, and selective visualization.

Vector Field Topology

Topological analysis of 2D linear vector fields was introduced by Helman and Hesselink [7,8]. It is based on detection and classification of critical points. The critical points of a vector field are those points where the vector magnitude is zero. The flow in the neighborhood of critical points is characterized by eigenanalysis of the velocity gradient tensor, or Jacobian of the vector field. The eigenvalues of the Jacobian can be used to classify the critical points as attracting or repelling nodes or focus, as saddle points, or center, see Fig. 1.1. The eigenvectors indicate the directions in which the flow approaches or leaves the critical points. These directions can be used to compute tangent curves of the flow near the critical points.



(a)

Figure 1.1: Vector field topology: critical points classified by the eigenvalues of the Jacobian [7].

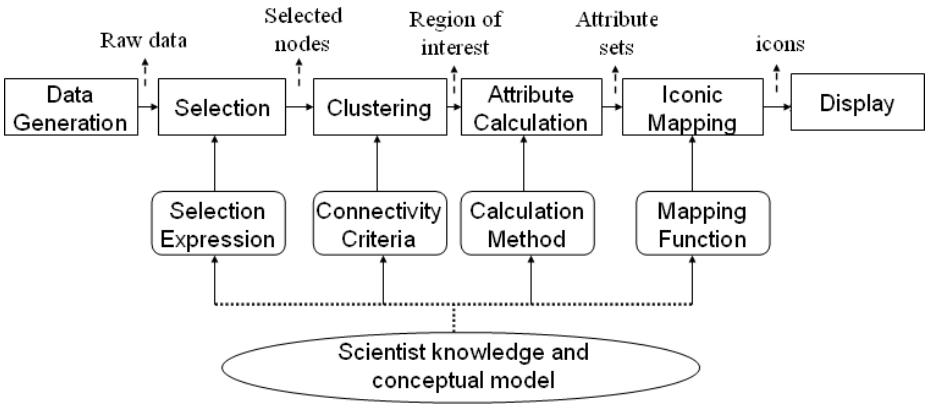
Tricoche, et al. [32] presented a topology-based method for visualizing time-varying 2D vector fields. This method performs time tracking of critical points and closed streamlines by temporal interpolation. They are able to find and characterize topological events or structural changes (*bifurcations*), such as the pairwise annihilation or creation of a saddle point and an attracting or repelling node.

Leeuw and Liere [6] introduced a technique for visualizing flow structures using multilevel flow topology. In high-resolution data sets of turbulent flow, the huge number of critical points can easily clutter a flow topology image. The algorithm presented attempted to solve this problem by removing small-scale structures from the topology. This was achieved by applying a pair of distance filters which removed small topology structures such as vortices, but did not affect the global topological structure. The threshold distance, which determines which critical points are removed, can be adapted, making it possible to visualize the structure at different levels of detail at different zoom levels.

Tricoche, et al. [31] also performed topology simplification in 2D vector fields. They simplified the topology, moreover, preserved the underlying vector field, thereby making it possible to use standard flow visualization methods, such as streamlines or LIC, after simplification. The basic principle of removing pairs of critical points is similar to the technique of De Leeuw [6], but in this algorithm the vector field surrounding the critical points is slightly modified, in such a way that both critical points disappear.

Physical Characteristics

The feature extraction based on physical characteristics is widely employed in solving visualization problems. Often, features can be detected by characteristic patterns in, or properties of, physical quantities, for example, by low pressure, high temperature, or swirling flow. These properties often follow directly from the feature definitions used.



(a)

Figure 1.2: The feature extraction pipeline (redrawn from [24]).

Selective Visualization

A generic approach to feature extraction is Selective Visualization, which is described by Van Walsum [34]. The feature extraction process is divided into four steps as illustrated in Fig. 1.2.

The first step is the selection step. In principle, any selection technique can be used, that results in a binary segmentation of the original data set. A very simple segmentation can be obtained by thresholding of the original or derived data values; also, multiple thresholds can be combined. The data set resulting from the selection step is a binary data set with the same dimensions as the original data set. The binary values in this data set denote whether or not the corresponding points in the original data set are selected. The next step in the feature extraction process is the clustering step, in which all points that have been selected are clustered into coherent regions. The third step is the attribute calculation step, in which the coherent regions are quantified. Attributes of the regions are calculated, such as position, volume and orientation. We now speak of objects, or features, with a number of attributes, instead of clusters of points. Once these quantified objects have been

determined, we do not need the original data anymore. With this, we accomplish a data reduction factor of 100 or more. In the fourth and the final step, iconic mapping, the calculated attributes are mapped onto the parameters of certain parametric icons, such as ellipsoids, which are easy to visualize.

1.2.5 State-of-the-art Feature Tracking Algorithms

Various techniques have been developed for features tracking of time varying data sets. In this section, we discuss three types of feature tracking algorithms based on region correspondence, attribute correspondence, and references.

Region Correspondence

Region correspondence involves comparing the regions of interest obtained by feature extraction. Basically, the binary images from successive time steps are compared on a cell-to-cell basis. Correspondence can be found using a minimum distance or a maximum cross-correlation criterion [10] or by minimizing an affine transformation matrix [14]. It is also possible to extract iso-surfaces from the four-dimensional time-dependent data set [35], where time is the fourth dimension. The correspondence is then implicitly determined by spatial overlap between successive time steps. This criterion is simple, but not always correct, as objects can overlap but not correspond, or correspond but not overlap.

Attribute Correspondence

With attribute correspondence, the comparison of features from successive frames is performed on the basis of the attributes of the features, such as the size, volume, position, orientation, etc. These attributes can be computed in the feature extraction stage, and can be used for description and for visualization of the features, as well as for feature tracking.

The original grid data are not needed anymore. Samtaney, et al. [25] used the attribute values together with user-provided tolerances to create correspondence criteria. For example, for position the following criterion could be used

$$dist(pos(O_{i+1}), pos(O_i)) < T_{dist} \tag{1.1}$$

where $pos(O_i)$ and $pos(O_{i+1})$ are the positions of the objects in time steps i and $i+1$, respectively, and T_{dist} is the user-provided tolerance.

Reference Correspondence

In [27], the authors used the object segmentation technique to classify points above threshold into a set of spatial objects based on the connectivity between data points. A *seed* point is found and then all of its neighbors are tested recursively for inclusion. Depending on the comparison, a new object consisting of the point is created till all the points in the neighborhood have been detected. If the point is adjacent to more than one object, merge all of these adjacent objects into a bigger object. After all the points above the threshold have been detected, they are restored in the octree data structure for the future tracking stage.

In the subsequent tracking stage, Silver and Wang [26,27] explicitly use the criterion of spatial overlap in four dimensions. They prevent correspondence by accidental overlap, by checking the volume of the corresponding features and taking the best match. The function for testing correspondence is

$$\frac{O_A^i * - O_B^{i+1}}{\max\left((O_A^i), (O_B^{i+1})\right)} < Tolerance \tag{1.2}$$

where "$*-$" is defined as

$$O_A^i * -O_B^{i+1} = \max\left(O_A^i - O_B^{i+1}, O_B^{i+1} - O_A^i\right) \tag{1.3}$$

The *tolerance* is a percentage value normalized to the maximum volume of objects being tested. It is chosen by the user and is domain-dependent. This algorithm is very sensitive to the tolerance. This equation can capture the differences and tell how the features overlap with each other in consecutive frames. After doing the feature tracking steps recursively, we can capture the evolution of objects along the time axis.

This algorithms is intuitive and generates very good result. However, there exist two problems. The first problem is that the user has to point out *seeds* for the initial data set; Furthermore, if the user pointed out a wrong seed, the computation time would increase, and even worse, the feature extraction and tracking procedure might fail. The second problem is the computation time, in the feature tracking stage, each feature in the previous time step has to compare with all the 3D features in the next step, and the data the users have to deal with is a huge 3D data sets and storing the result in Octree data structure is a very time-consuming procedure.

In addition to the above main-stream algorithms, another approach introduced by Tzeng and Ma [33] employed an artificial neural network to perform the learning of features, then using the training set to test and detect similar object in other frames. Feature extraction was done by explicitly separating the feature of interest from the raw data sets, and creating either a volumetric or geometric representation of the extracted feature such as the data from 1D transfer function, the histogram value of data, and the time step. The representation of feature of interest was imported into the neural network to train the network. After the neural network has been trained, new frames are imported, the object which has similar representations will obtain the highest correspondence value and thus be selected.

1.3 Thesis Contributions

This thesis work concentrates on introducing a new image-based feature extraction and tracking method for interpreting large scale 3D fluid data sets. Large scale 3D fluid data sets will be first projected into a 2D image space, after applying advanced image processing algorithms, features will be extracted and tracked, hence the evolution of objects in 3D space is evaluated. There are four main procedures involved in the process, feature segmentation (or extraction), feature description (or shape analysis), classification, and feature tracking.

The biggest contribution of the thesis work is that the proposed approach is completely *automatic* in that it does not require the availability of any prior models or any interaction between the user and the data set. Therefore, the proposed image-based approach is more flexible and can be used to analyze any 3-D fluid data sets. In addition, since the processing is conducted entirely in the image domain instead of the 3D object space, the computation burden and memory space are largely alleviated.

1.4 Thesis Outline

The organization of the thesis is as follows: In Chapter 2, after an overview of the proposed image-based processing procedures, the algorithms designed for feature extraction will be described. Chapter 3 presents the proposed methods for feature tracking. The experimental results from this thesis research are presented in Chapter 4. These results will be compared with those generated from state-of-the-art algorithms to evaluate the performance of the proposed approach. Finally, we conclude in Chapter 5 and identify future research directions.

Chapter 2

Automatic Feature Extraction

In essence, two issues need to be resolved in order to realize automatic identification and tracking of features entirely in the image space, the extraction of individual features from each image and the identification of the same feature from images generated at adjacent time steps for tracking purpose. In the area of digital image processing, these two issues are also referred to as image segmentation and image registration, respectively. Although there have been advanced approaches to these problems, the highly dynamic and highly overlapping characteristics of the volume rendered images present new challenges.

The proposed image-space automatic 3D time-varying features extraction and tracking algorithm is a four-step process, shown in Fig. 2.1, that contains feature segmentation, feature description through shape analysis, feature classification, and feature tracking. The focus of this chapter is on feature segmentation and feature description. Chapter 3 will discuss feature classification and tracking. In the feature extraction step, features are distinguished from the background and their neighboring features based on several advanced image segmentation methods; then the intrinsic parameters which can represent features just by a descriptor vector is derived in the shape analysis step; the classification step uses

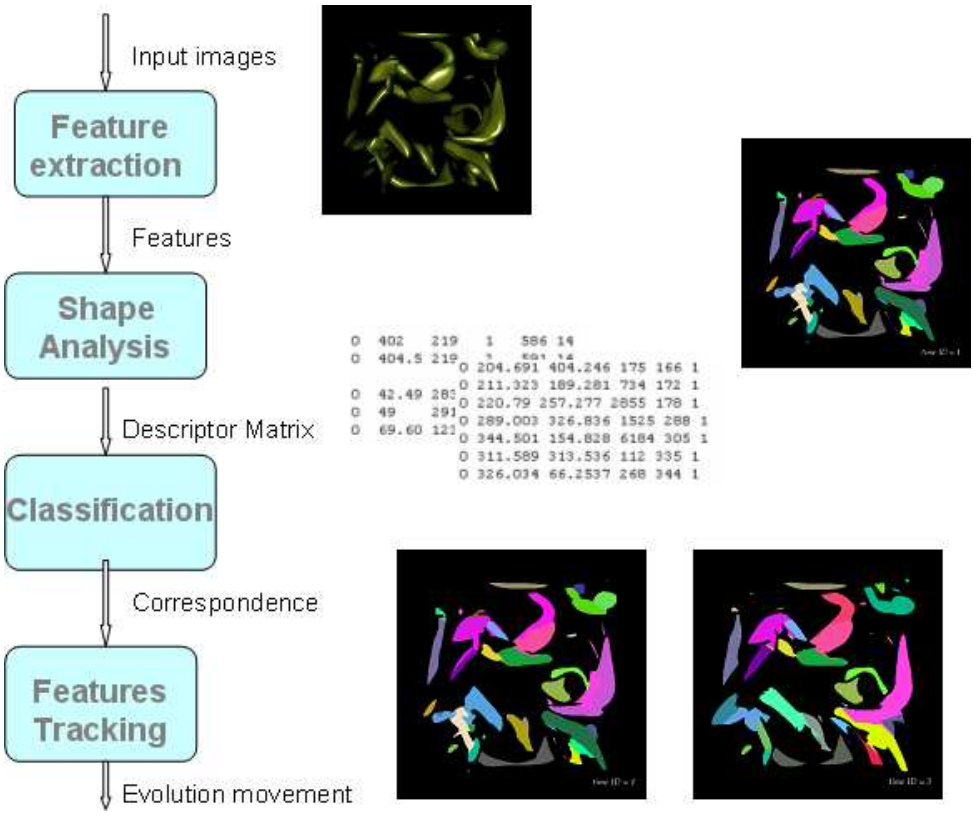


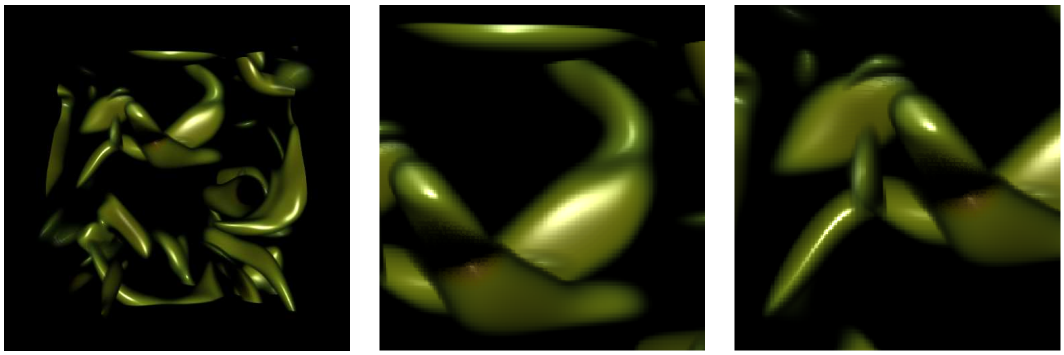
Figure 2.1: The system diagram for the proposed image-based approach.

pattern recognition method to calculate the correspondence of features in image sequences, the tolerance distance will be estimated by the system itself based on the current data set; and the tracking step labels and tracks the corresponding features derived from subsequent time steps for visualization purpose.

2.1 Feature Segmentation

By observing one of the volume rendered images as shown in Fig. 2.2(a), we identify two potential problems that hinder a high-fidelity feature segmentation. First of all, the different lighting conditions (or different shading information) associated with some larger features would render conventional segmentation algorithms ineffective, as illustrated in Fig. 2.2(b). Secondly, the high-degree occlusion between features challenges both contour-based and region-based segmentation algorithms, as shown in Fig. 2.2(c). For human eyes, it is not a difficult task to distinguish individual feature's shape because human eyes are very sensitive to the color change, however, how to implement the similar function is a big challenge for a computer.

In the area of image processing, segmentation is the process to separate foreground target features from their background. Segmentation can be achieved through the detection of either the edge/contour of the feature or the region occupied by the feature. Edge-based segmentation identifies sudden intensity changes at the boundary of the feature, therefore, also being referred to as *segmentation by difference*. On the other hand, region-based segmentation identifies homogeneous areas occupied by the feature assuming that the intensity value of the same feature should not vary too much, and hence the name *segmentation by similarity*. In this thesis, we integrate both segmentation principles to improve the performance of the proposed system. The segmentation process is outlined in the block diagram of Fig. 2.3 and explained in the following four steps:



(a) (b) (c)

Figure 2.2: Several problems need to be solved in the following processes (a) The original input image: time-varying 3D volume dataset. (b) The false feature caused by reflection. (c) Adjacent and occluded features which is easy to be distinguished by human eyes but difficult for computers.

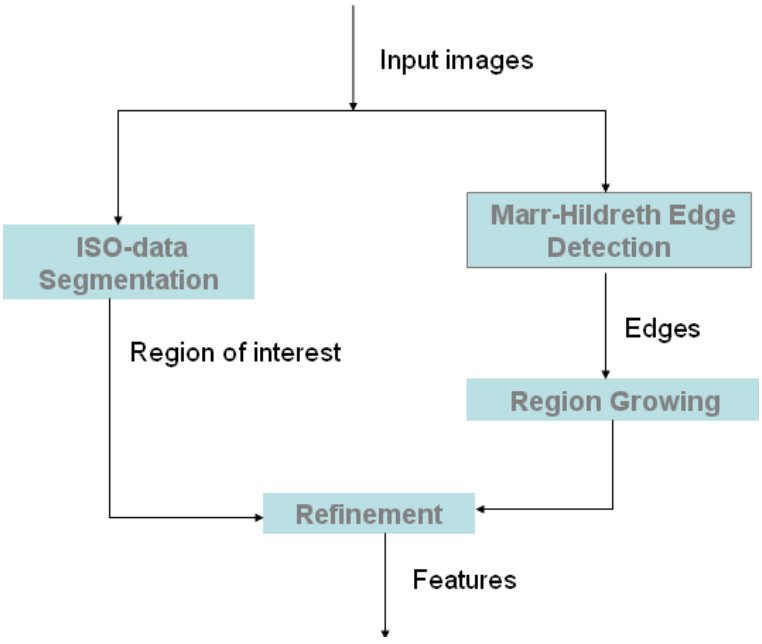
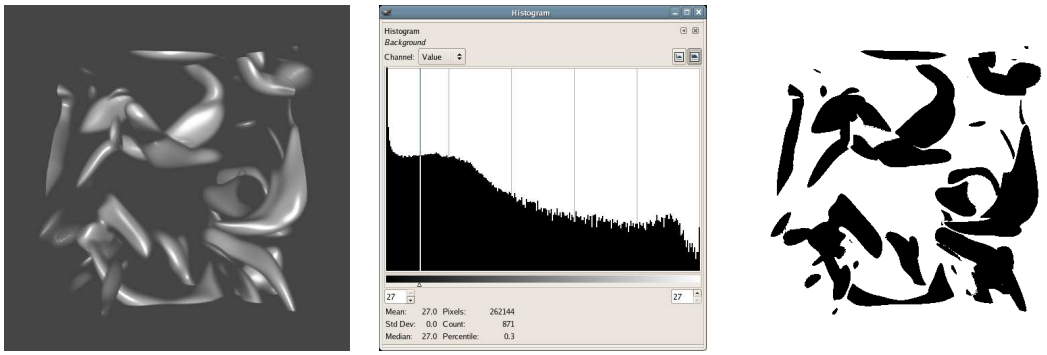


Figure 2.3: The block diagram for automatic feature extraction.

- Step 1: Apply the iso-data dynamic segmentation algorithm to separate features from the background
- Step 2: Apply the Marr-Hildreth edge detection algorithm to obtain closed contours for each individual region
- Step 3: Apply a dynamic region growing method to label each different region surrounded by edges obtained from the Marr-Hilldreth edge detector
- Step 4: Within the region of interest from step1, use the initial growing regions obtained in Step3 as seeds, expand the regions until they reach the boundary of the region of interest, or when more than one features meet with each other.
- Step 5: Remove the false contours from step2 , at the same time, remove the false contours which were caused by highlighting comparatively big features in 3D spatial domain. Refine the result of all labeled regions.

2.1.1 Thresholding based on ISO-data Method

In image processing, segmentation is always one of the first processing steps as segmentation can isolate regions of interest for further processing. In our case, the original input data sets are color images. Since the information revealed by the RGB values is very limited, the first pre-processing step is to convert the color images into gray-scale images whose intensity values range from 0 to 255. How to automatically choose an appropriate value, i.e., threshold, to divide the entire image into the foreground region and the background region is the first problem we will face. After studying several state-of-the-art thresholding algorithms, we choose to apply the ISO-data segmentation algorithm [30]. In addition, in order to gain a more accurate result, an improved ISO-data segmentation method has been



(a) (b) (c)

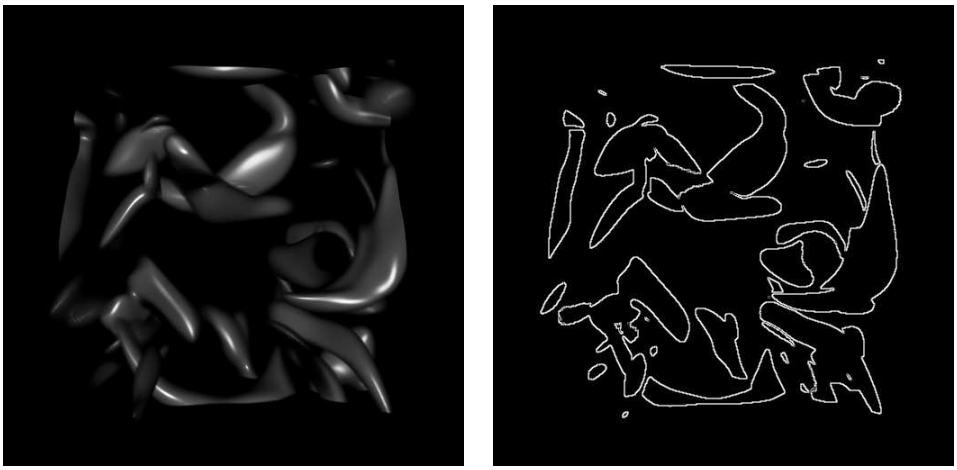
Figure 2.4: The input and output images of ISO-data segmentation method. (a) The gray scale image. (b) The histogram information of (a). (c) Pixels below the threshold θ will be labeled as background pixels with an intensity of 255 (white); those above the threshold will be labeled as object pixels with an intensity of 0 (black). ($\theta = 27$)

introduced into this system. The main reason to choose this method is its capability of conducting segmentation automatically.

The ISO-data segmentation algorithm starts from the histogram of the original image. An initial threshold is randomly chosen, and all the pixels with intensities above this threshold become foreground pixels and all those below the threshold become background pixels Fig. 2.4. The sample mean $m_{f,0}$ of the gray values associated with the foreground pixels and the sample mean $m_{b,0}$ associated with the background pixels are calculated. A new threshold θ_1 is then derived as the average of these two sample means. The process continues, until the threshold value does not change any more. That is

$$\theta_k = (m_{f,k-1} + m_{b,k-1})/2 \text{ until } \theta_k = \theta_{k-1} \tag{2.1}$$

where $m_{f,k-1}$ and $m_{b,k-1}$ are the average gray value of the foreground and the background pixels in iteration $k-1$; θ_k, and θ_{k-1} are the threshold values in iteration k and iteration $k-1$, respectively. To make the segmentation more flexible and accurate, one image can be



(a) (b)

Figure 2.5: Contours of the ROIs derived from applying the ISO-data segmentation. (a) The input gray-scale data set. (b) Edges after applying ISO-segmentation method.

divided into several small sub-images and ISO-data segmentation methods can be applied locally. Fig. 2.5 shows the contours derived after applying the ISO-data segmentation.

2.1.2 Marr-Hildreth Contour Detection

Many effective edge detectors have been proposed and studied. The most popular ones include the Sobel edge detector, the Marr-Hildreth edge detector, and the Canny edge detector. Generally speaking, these detectors may be grouped into two categories, gradient-based and second derivative-based. The gradient method detects the edges by looking for the maximum and the minimum magnitude in the first derivative of the image. The second derivative method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location. In the first derivative edge detection method, a pixel location is declared an edge location if the value of the gradient exceeds some threshold; in the second derivative edge detection methods, a pixel location is declared an edge location

if it is a zero-crossing point, that is, if its two direct neighbors along any direction change signs in their second derivatives.

As mentioned before, edges will have higher absolute gradient magnitude than those surrounding it. So once a threshold is set, we can compare the gradient magnitude to the threshold and detect an edge whenever the threshold is exceeded. For example, the Sobel edge detector belongs to this family. Furthermore, when the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. Marr-Hildreth edge detector belongs to this category.

In our system, we employ the second derivative contour detection method: Marr-Hildreth contour detector. The gradient edge detector is one of the most widely used edge detectors in image processing and generates accurate contour in a short time, however, it is highly sensitive to the threshold selected, especially when large amount of noise is present. On the other hand, the Marr-Hilldreth [19] detector detects the zero-crossings after applying the Laplacian of Gaussian (LoG) operator to the original image. The original image is first smoothed by a Gaussian low-pass filter to alleviate the effect of noise. Then the Laplacian filter is apply to the smoothed image to derive the second derivative. Fig. 2.6 shows a Gaussian and the results after applying three different edge detectors. The 2D Laplacian operator in second derivative can be written as:

The zero crossings are pixel positions where the LoG of the image changes signs at its two immediate neighbors. After being filtered by the LoG filter, zero crossing pixels are detected by observing the sign changes in four directions, North-South, East-West, SW-NE, and NW-SE, crossing this pixel. If any sign changes are detected in these four directions, the pixel is considered an edge pixel. The problem with this approach is that the zero

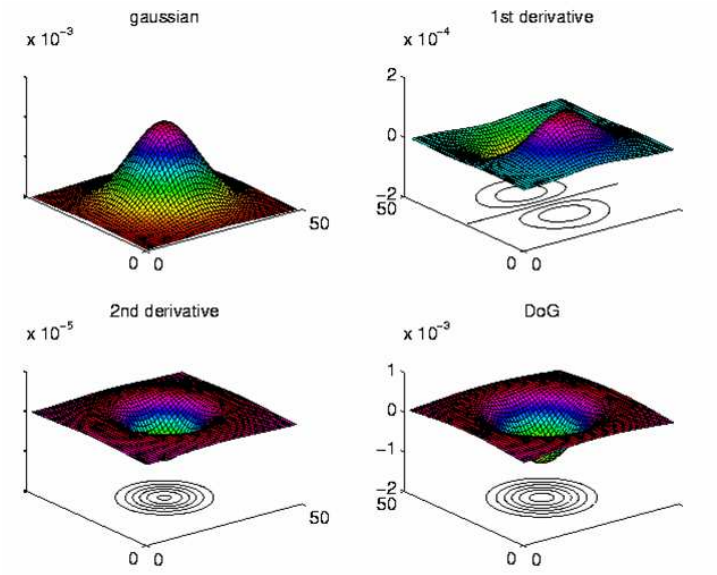


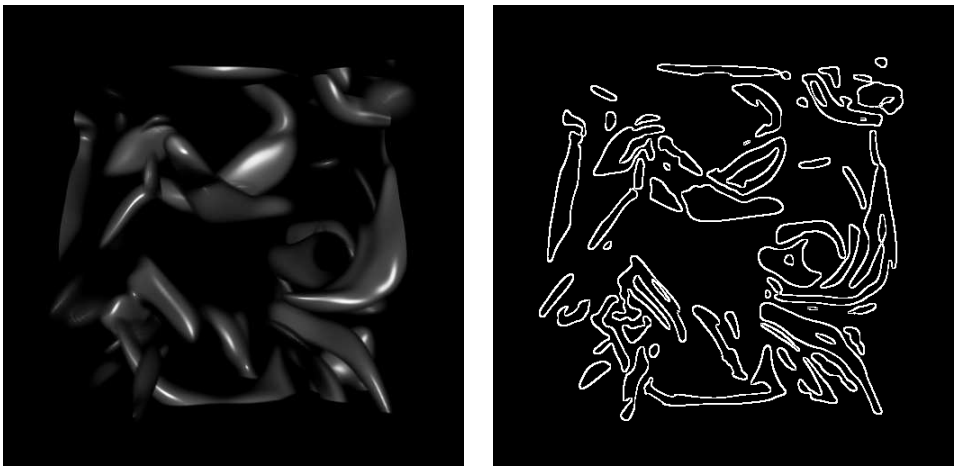
Figure 2.6: The Gaussian and the results after applying three different edge detectors.

crossings are strongly influenced by the radius of the LoG filter. The larger the radius,

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}[1 - \frac{x^2+y^2}{2\sigma^2}]e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.2}$$

the fewer zero crossing contours will be found. Those that do remain will correspond to features of larger scale in the image. Since the size of the features varies from very small to comparatively large, we choose a relatively small radius.

Compared between edges derived from the Marr-Hildreth detector (Fig. 2.7) and those from the ISO-data segmentation (Fig. 2.5), we observe that the former tend to have closed contours that enclose smaller regions than supposed to, thus contours belong to different features tend to be better separated; while the latter generates more accurate edge locations. In addition, Marr-Hilldreth contour detector may generate edges in place where no edge pixels are supposed to exist and the computation time is comparatively long because of the size of the LoG filter. Therefore, we combine the contours from the Marr-Hildreth contour



(a) (b)

Figure 2.7: Contours derived after applying Marr-Hildreth contour detector. Notice that the shapes of features are already different from their real shapes; but the contours better separate features from each other compared to ISO-data segmentation. (a) The input gray-scale data set. (b) The output contours from Marr-Hildreth contour detector

Figure 2.8: Superposition of region of interests derived from ISO-data segmentation and the contours from the Marr-Hildreth edge detection.

detector with the foreground regions derived from ISO-data segmentation, as shown in Fig. 2.8. The region which we extracted from ISO-data segmentation contains the regions where the potential features locate, although several features might be mixed together. Nonetheless, the regions of interest do give us an outer boundary to refine the result from Marr-Hildreth contour detection. First of all, the restriction of the foreground regions will help remove the false contours from the Marr-Hildreth contour detection. Secondly, we can use the region enclosed by the contour derived from the Marr-Hildreth detector as seeds. By growing this seed, we extend the area covered to gradually converge to the real shape of the feature that is confined by the ROI derived from the ISO-data segmentation.

2.1.3 Canny Contour Detection

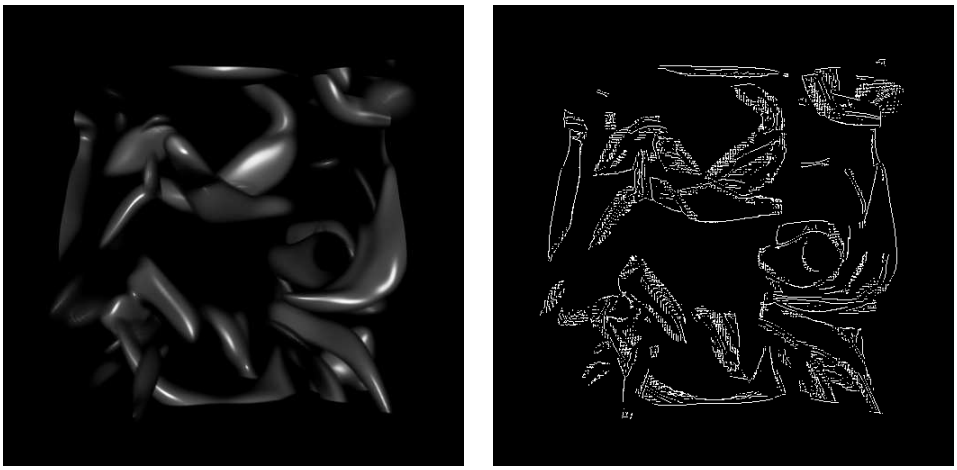
Besides the Marr-Hildreth contour detector, we also study the Canny contour detector, which is known as the optimal edge detector.

The Canny operator works in a multi-stage process. First of all the image is smoothed by Gaussian low-pass filter, a step very similar to that adopted in Marr-Hildreth edge

detection. Then a simple 2-D first derivative operator is applied to the smoothed image to highlight regions of the image with high first spatial derivatives, the Sobel edge detector is usually applied in this stage. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as non-maximum suppression. The tracking process exhibits hysteresis controlled by two thresholds: $T1$ and $T2$ with $T1 > T2$. Tracking can only begin at a point on a ridge higher than $T1$. Tracking then continues in both directions out from that point until the height of the ridge falls below $T2$. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments. Different from either the Sobel contour detector or the Marr-Hildreth contour detector, the Canny contour detector can give one pixel width contours.

The effect of the Canny operator is determined by three parameters: the width of the Gaussian mask used in the smoothing phase, and the upper and lower thresholds-$T1$ and $T2$-used by the tracker. Increasing the width of the Gaussian mask reduces the detector's sensitivity to noise, at the expense of losing some of the finer detail in the image. The localization error in the detected edges also increases slightly as the Gaussian width is increased. According to Fig. 2.9(b), the contours from the Canny contour detector contain more details than the contours from the Marr-Hildreth contour detector; and though it can give one pixel width contours, gaps can be observed from the obtained contours.

As we discussed in Sec. 2.1.2, The Marr-Hildreth contour detector can give closed contours, in this case, we can take advantage of this property in the following region growing stage. The pre-requisite for performing region growing is that it can only fill in the closed contours. Therefore, although the Canny contour can give more accurate contour shapes, we choose the Marr-Hildreth contour detector so that closed contours can be obtained.



(a) (b)

Figure 2.9: The result after applying the Canny contour Detector. Notice that the contours are not closed, Moreover, the image contains more extra edges because the intensity change in some places are not sharp enough. (a) The input gray-scale data set. (b) The output contours from the Canny contour detector

2.1.4 Region Growing based on Closed Contours

Since the Marr-Hildreth contour detector tends to result in closed contours, the iterative region growing method which will be introduced in this section is preferable to other techniques which are based on edge detection or line fitting. This algorithm is functionally identical to recursive region growing in that it returns a set of labeled pixels, meeting the adjacency and similarity criteria [28].

While this algorithm is functionally identical to traditional region growing, it is fundamentally different in concept and potential implementation. The objective of this algorithm is to find a way to achieve the results of region growing, but to do so "on the fly" with a single pass over the data. This result was achieved by using the concept of a content-addressable memory, serving like a lookup table.

The iterative region growing algorithm is based on the concept of equivalence relationships between the pixels of an image. Two pixels a and b are defined to be equivalent (denoted R(a,b)) if they belong to the same region of an image. This relation can be shown to be relative ($R(a,a)$), symmetric ($R(a,b) \Rightarrow R(b,a)$), and transitive (R(a,b) AND $R(b,c) \Rightarrow R(a,c)$); which makes it an "equivalence relation". The transitive property enables all pixels in a region to be determined by considering only local adjacency properties. The image will be scanned in a raster-scan manner, from left to right, top to bottom. The assignment of a region label to a pixel results from this comparison operation. The region labeling proceeds in a straightforward manner: For each pixel, first look to its left, consider that its neighbor and use its label. Then look up to its above, and consider that pixel its neighbor. If both are labeled and the labels are different, we choose the smaller of the two. To accelerate the process, we only consider 4 adjacent neighbors: the top and the left. This modification will save at most half of the operating time. The algorithm is described in the following according to [28]:

- $f(x, y)$ is the gray-scale value of the (x, y) pixel in the image memory.
- $(x, y)_i$ is the i^{th} adjacent neighbor of the (x, y) pixel.
- $f_i(x, y)$ is the gray-scale value of the i^{th} adjacent neighbor of the (x, y) pixel.
- $L(x, y)$ is the region label corresponding to the (x, y) pixel in the image memory.
- $L_i(x, y)$ is the region label number corresponding to the i^{th} adjacent neighbor of the (x, y) pixel.
- $K(i)$ is the contents or the i^{th} element in the equivalence memory.
- $K^*(i) = K(j)$ implies the sequence:
  - read $K(j)$
  - search K using i as a search key
  - write $K(j)$ to all positive responders of the search
- T is a threshold
- P is the highest numbered region label (initially 1)
- N is the number of pixels in a neighborhood (four connectivity)

The iterative region growing method fills in the region surrounded by the contours generated from the Marr-Hildreth contour detector, and labels the pixels in that region with a unique label value. This value can be treated as the key to distinguish this feature from others. After processing the data set in this stage, the basic features have been extracted. In order to view the distinguished features more clearly, we convert the gray-scale label image to a pseudo-color image with each label corresponding to a different color Fig. 2.10.

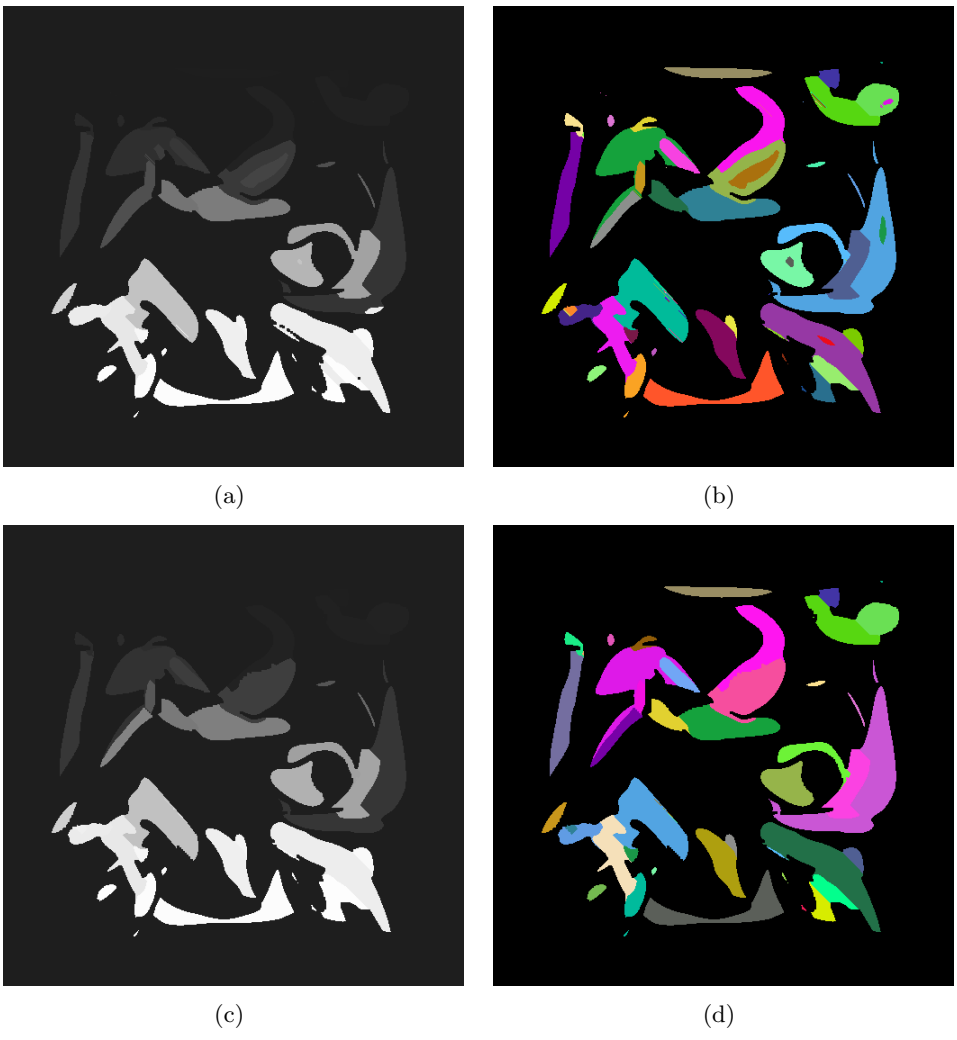
(a) (b)

Figure 2.10: Basic feature extraction. (a) The basic features in gray-scale image. (b) The basic features in pseudo-color image

2.1.5 Automatic Feature Expansion and Refinement

Although region growing based on the closed contours generated from applying the Marr-Hildreth detector has identified the rough areas of features, the boundary of the features are not accurate. Therefore, in this section, we use these initial regions as seeds and expand them outward until they reach the foreground regions derived from the ISO-data segmentation method or reach the boundary where two features meet with each other. At the same time, we also eliminate the false contours derived from Marr-Hildreth if they are out of the regions of interest. We name this approach *Automatic Feature Expansion*

There are two main problems which will affect the feature tracking task significantly: First, as we mentioned at the beginning of the chapter, the different lighting features associated with some larger features would cause false contours inside the real features, they have similar shapes but the false feature is completely included within the real feature, Fig. 2.10 shows an example of the false regions which are caused by reflection. This property provides us an intuitive solution: Observe each feature and extract the ones who is surrounded completely by other features and replace its label by the one surrounding it; the extra restriction is that they have similar intrinsic properties such as similar Center of Gravity values and the direction of principal axis, see Sec. 2.2; Secondly, in Sec. 2.1.2, we mentioned that one of the disadvantages of Marr-Hildreth contour detector is that it may mark contours at some locations that no contours exist. The way to solve this problem is to rely on the region of interests we obtained in Sec. 2.1.1 to eliminate those contours. Fig. 2.11 demonstrates the region expansion and refinement results.



(a) (b)

(c) (d)

Figure 2.11: Results from region expansion and refinement. (a) Features expansion before removing false contours in gray-scale image. (b) Features expansion before removing false contours in color image. (c) Features expansion after removing false contours in gray-scale image. (d) Features expansion after removing false contours in color image.

2.2 Feature Description by Shape Analysis

Shape analysis can be defined as the operation to represent the features of interest through various parameters derived from the features. It is a procedure to mimic the procedure of human beings to recognize the complex features in an image. Humans are able to extract features by utilizing the attributes of shape, such as size, texture, pattern, shadow, and association, then recognizing those complex features by the quantitative data sets rather than the features themselves. Similarly, computer can take advantage of this process by converting the image data into a quantitative form which can preserve the intrinsic properties of the feature for the subsequent analyses. Moreover, by doing this, a large volume 2D data set can be represented using a small volume 1D vector with each of the element in the vector a quantitative parameter derived from the feature of interest. This operation will reduce the computation time significantly. The popularly used features are listed as follows [28]:

- *Center of Gravity (CoG).* The x and y coordinates of the center of gravity is written as $m_x = \frac{1}{N}\sum x$, and $m_y = \frac{1}{N}\sum y$, where x is the x-coordinate, y is the y-coordinate of each pixel

- *Area.* A count of all pixels included in the feature

- *Diameter.* The diameter describes the maximum chord - the distance between those two pixels on the boundary of the region whose mutual distance is maximum [25]

- *The principal axis.* The axis that minimizes the sum of squares of the perpendicular distances from the points in the region to it. *Direction of the principal axis.* The angle between the principal axis and the x-coordinate.

- *Average gray value.* The average value of the pixel intensity in a certain feature.

In our experiment, the average gray value varies a lot mainly due to the lights cast on the time-varying 3D volume data when they were projected onto image plane, and is not a property of features themselves, therefore, it is not a good intrinsic parameter to represent features; The perimeter and the diameter reveal similar information about the feature, therefore, we only use one of them.

2.2.1 The Principal Axis

After obtaining the regions of features, to describe its shape is a difficult task. In language, we usually describe a shape in abstract terms like "long", "thin", and the orientation of a region. In computer vision language, we make use of the parameters such as "diameter" to describe the shape of features.

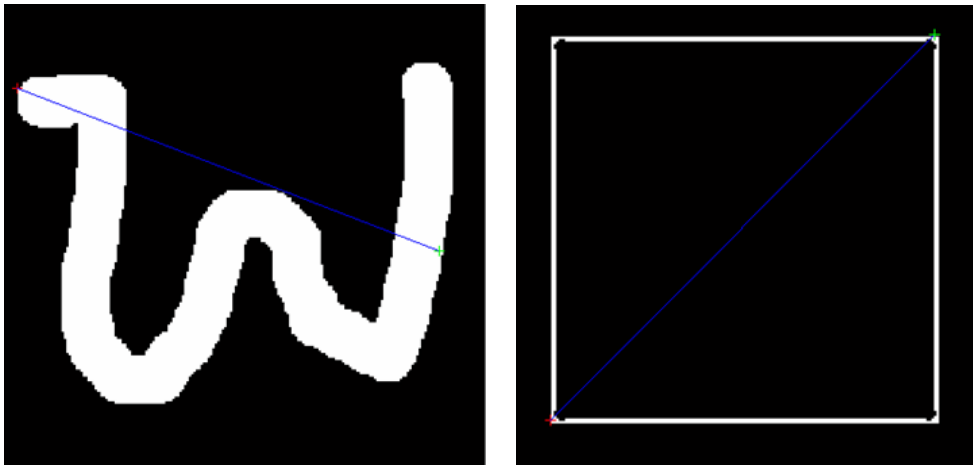
The technique introduced in this section will use eigenvalue analysis to find the "best" estimate of the major axis of the region, the principal axis, and define the extreme as the two points furthest out in the direction of that axis, diameter.

In two dimensions, principal axis is the line which minimizes the sum of squares of the perpendicular distances from the points in the region to the axis.

There are two algorithms that can help find this principal axis: the minimum square error (MSE) technique and the eigenvector line fitting (ELF) technique. In this thesis we choose the second technique as ELF is computationally more efficient than MSE, Fig. 2.12.

$$d^2 = \sum d_i^2 \tag{2.3}$$

d_i indicates the distance between the ithpixel in the region of interest and the principal axis. d is the summation of d_i. In order to find the minimum value of d, the principal axis must pass through the CoG of the region. Thus it is necessary only to find the slope of the axis. Take the CoG as the origin of the new coordinate system, the problem becomes:



(a) (b)

Figure 2.12: Basic feature extraction: finding principal axis. (a) The principal axis of character "w". (b) The principal axis of a square.

Given n pixels which have been normalized to have zero mean, find the line through the origin which minimize d^2. In matrix, this is a simple procedure to calculate the eigenvalues and the corresponding eigenvectors. The direction of the largest eigenvector is exactly the answer we want.

2.2.2 Diameter

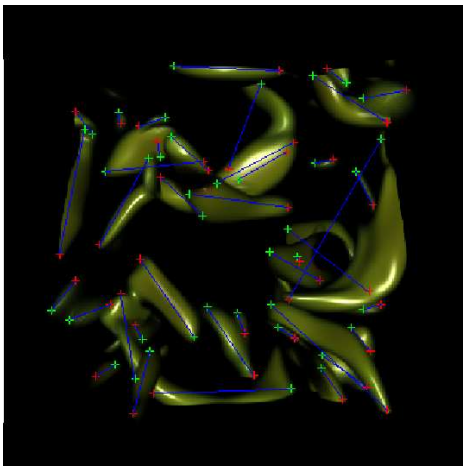
Having found the principal axis, one then treats each pixel on the boundary as a vector and projects each onto the principal axis. The extreme are then those two points on opposite sides of the CoG whose projections onto the principal axis have the maximum distance. This approach yields a solution which is an accurate representation of the "diameter" of the region.

For implement point of view, we divide the image into four Regions - the quadrants of the Cartesian coordinate system, with CoG being the origin, the principal axis being the abscissa. Since the pair of pixels should be located on the opposite sides of the origin, there are only four combined distance value among four pixels on the boundary: distance between the pixels in quadrant1 and quadrant3; distance between the pixels in quadrant1 and quadrant2; distance between the pixels in quadrant4 and quadrant3, and distance between the pixels in quadrant4 and quadrant2. The maximum value was selected as the diameter of the feature, Fig. 2.13.

The final descriptor vectors derived includes:

$$feature\ vector = [timeID, viewID, m_x, m_y, area, diameter, angle\ of\ principal\ axis, label] \tag{2.4}$$

In order to present the principal axis more accurate,we choose the angle between the principle axis and the x-axis before the normalization.



(a)

Figure 2.13: The principal axis for individual features. For each feature, one principal axis has been calculated, the diameter is the length of the principal axis within the boundary of the feature.

2.3 Summary

This chapter focuses on how to automatically conduct feature extraction. Several advanced image processing algorithms have been integrated and applied to generate the desired result. These algorithms include the usage of the ISO-data segmentation for foreground region identification, the Marr-Hildreth and region growing to generate the seed features which can be expanded in the region expansion stage. In Chapter 3, the main problem is to design the system which can implement correspondence computation and features tracking automatically.

Chapter 3

Descriptor-based Feature Tracking

The objective of this thesis work is to track the evolution of the 3D volume features based on a sequence of images in continuous time domain. The algorithms we developed in Chapter 2 have helped solve the problem of extracting features from the background and their neighboring features, representing features by a descriptor vector. This kind of pre-processing is needed for the following analysis on observing the relationship among a sequence of image-based data sets. In this chapter, we focus on how to find the relationship among features in a pair of subsequent 2D data sets in the time domain as well as the spatial domain.

3.1 Different Evolution Events

Over time, features evolve constantly and the transform functions are not necessarily affine transforms. Nonetheless, between two adjacent time steps, the transformation of the same feature should not be dramatic. For example, the position, size, and orientation of the feature should not vary too much. This is the rationale behind the proposed descriptor-based feature tracking. In another word, we derive a descriptor vector associated with each feature formed by descriptors (e.g., position, size, orientation, etc.) that under normal

circumstances would not generate large variations. We evaluate the closeness of descriptor vectors between two adjacent time steps and use this information to track features.

Similarly, if the positions of the view points do not change significantly, e.g. less than 15 degrees [3,20], the non-rigid stereo features can be reconstructed from at least two adjacent spatially-varying images. It is like the way that human beings observe the world around them. Two slightly different pictures are projected onto the retinas through the lenses of human eyes, which are then transformed, by the brain, into a spatial representation. The actual stereoscopic spatial observation is a result of this perception through both eyes. In projection geometry, under the acknowledgement of the intrinsic as well as the extrinsic parameters of the cameras, at least three pairs of 2D points on image plane can be targeted and localized on both image planes. Those 2D points on image planes are the projection of one 3D point in the 3D space onto two image planes respectively. In this thesis work, we mainly focus on tracking the features in the spatial domain rather than targeting those 2D points. It is a very useful pre-processing step to the future feature reconstruction task since it can narrow down our attention to correspondent features in a sequence of image-based data sets so that it is much easier to detect the related 2D points in the image sequences.

According to [25, 26], the evolution of time-varying 3D volume objects can be characterized as *continuation, creation, dissipation, bifurcation*, and *amalgamation*, in which *continuation* refers to the event that one feature continues from a data set at t_i to the next one when time steps to t_{i+1}, *creation* is where a new feature is created in time step t_{i+1}, *dissipation* refers to disappearing features, *bifurcation* indicates that a feature in time step t_i is divided into at least two sub-features in time step t_{i+1}, and *amalgamation* is where at least two features in the previous time step merged into a single feature in the next time step. Figure 3.1 illustrates examples of three evolutionary events.

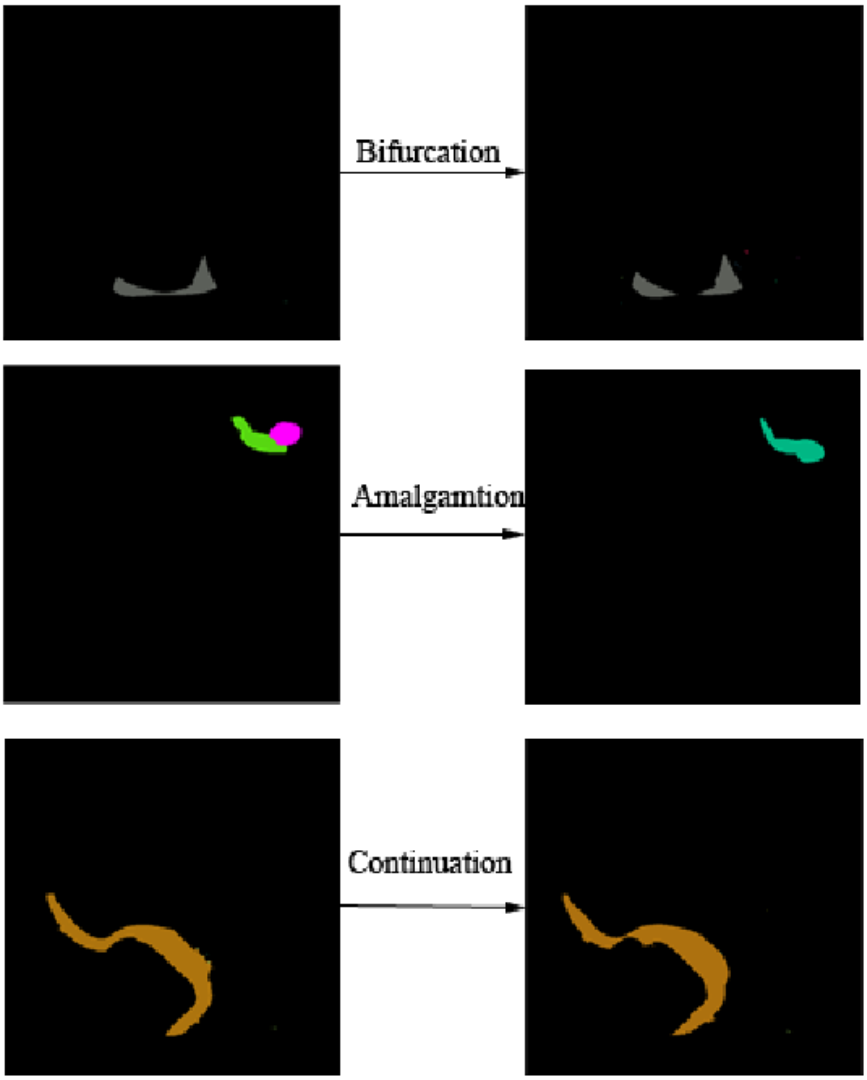


Figure 3.1: Examples of evolution events. The pair of images on the first row indicates one object separates into two objects which is known as bifurcation; the images on the second row indicates two objects merge into one which is defined as amalgamation; a pair of images on the third row shows one object transforms into another one from t_i to t_{i+1}

For visualization purpose, we will not only go through the time domain but also the spatial domain to target and recognize features . The definition of the evolution events in the spatial domain is totally different from that in the time domain, though we use the same terminology to describe it. The main difference between the time domain analysis and the spatial domain analysis is that, in the spatial domain, the volume data sets can be considered as rigid features, in other words, the velocity of the volume data sets will keep the same, and their shapes keep the same, too. However, human's perception shows that the feature shapes change if the features are observed from different positions at the same time. The reasons of this phenomenon are that the features are deformable, and nonsymmetrical objects, and more than one feature is overlapping or adjacent with each other in the research space so that some maybe invisible or partially visible if their positions are further away from the observer. Based on this agreement, the events *Dissipation* and *Creation* are due to: some features are totally blocked by those whom are closer to the view position; Since *bifurcation* passes the information that one feature is partially blocked by others, this feature is not suitable for visualization purpose any more while *amalgamation* implies that one feature starts facing to the observational image plane. In the spacial domain, only the evolution event *transformation* is the valuable event and only the features which remain their evolution pattern as transformation can be used for further analysis.

3.2 Automatic Feature Tracking

In Chapter 2, the foreground features have been successfully extracted and represented by its robust intrinsic properties: a descriptor vector. Compared with the original size 512×512 for one data set, the memory space to store all the descriptor vectors is $M \times N$, in which M indicates the number of features, N indicates the length of one descriptor vector. One

descriptor vector is uniquely identified to represent one feature. In Sec. 2.2, eight intrinsic descriptors have been discussed and extracted.

The objective of this stage of processing is to find the relations of features between a pair of images obtained over consecutive time frames. This is a typical classification problem, in which the same feature is to be identified in two image frames.

3.2.1 Nearest Neighbor(NN) Measurement

Although the labels used in different data sets may be identical, it does not mean they represent the same feature. The function of those labels is to identify features in the same data set. The value itself has no meaning. The descriptor vectors of features from two adjacent time frames or view angles are compared through a distance measurement. Each feature is quantitatively described by a vector of five elements: m_x, m_y, diameter, angle of principal axis, area. Euclidean distance is calculated between pairs of the descriptor vectors and the distance sorted. We refer to this distance as the "correspondence value", and set up a "tolerance distance" of this correspondence value. We are only interested in correspondence values less than or equal to the tolerance distance. For two consecutive time steps t_i and t_{i+1}, do the following,

- Extract features from the two data sets and store them as nodes in a new data structure LabelTrack, an inherited class from list. One node represents a single feature in one time step. One list includes all nodes and preserves the useful information of according features in one time step. The structure of the node includes two sub-lists with one containing the information of its previously related node and the other containing the information of its related node in the next time step; the key value of each node is "Label", and the value of the node is a descriptor vector to describe a single feature's intrinsic properties

- For each node in t_i, compute the correspondence feature values with all nodes in t_{i+1} based on the diameter value, center of gravity and areas

For each feature in t_i, if only one feature at t_{i+1} satisfies the tolerance requirement, we determine the evolution event as continuation; if more than one features in t_{i+1} meets the tolerance requirement, we determine the evolution event as bifurcation; if no feature in t_{i+1} meets the requirements, it will be determined as dissipation. If at t_i, more than one feature has correspondence with a feature at t_{i+1}, then the evolution event is amalgamation; and finally if at t_i, there is no feature that corresponds to a feature at t_{i+1}, then this evolution event is creation.

3.2.2 Dynamic Implicit Tolerant Distance Selection

As we have declared at the beginning of this Thesis, the most important contribution of this system is automatic processing. In the feature tracking module we have faced the problem of dynamically deciding the tolerant distance threshold.

Determine the tolerant distance for intrinsic value CoG:

The resolution of the 2D image-based data set is 512×512, the tolerance distance is calculated by

$$dist = \frac{L}{2N_{features}} \tag{3.1}$$

where L is the number of pixels in width from the original data set. $N_{features}$ is the number of features which has been extracted from the same data set. This equation shows that we assume the features are adjacent with each other to fill the whole image space and there is no overlap between them. At the most their movement from one time frame to the next one should not exceed their average width.

Determine the tolerant distance for diameters: To evaluate the tolerant distance for diameters, the first step is to rearrange the diameter values in one data set by descending

order. The feature whose diameter value locates in the middle of the descending list is selected out, and the half of the diameter value is considered as the tolerance distance for correspondence calculation between features in current time step and features in the next time step. As illustrated in the previous section, the region of some features is big, while the others have relatively small regions. Therefore, they have accordingly very different diameters. To select the half diameter value of the $N/2th$ smallest features is a reasonable tolerance value that we assume at least half of the features will not change their shapes too much in a relatively short time interval.

3.3 Data Structure in Computer Memory

In order to store and manage the features after doing the correspondence calculation, we design a new class *LabelTracking* to save all the information for each feature. The features in one data set are preserved as nodes as illustrated in Fig 3.2 in one vector data structure, the length of the vector is the number of extracted features in one data set at a certain time view. A set of vectors has been arranged by the key value "labelID" of four fields.

- *timeID*. The identification of the time step that the current feature belongs to
- *viewID*. The identification of the view angel that the current feature belongs to
- *classID*. The number of the old labeling value from extraction stage to identify the current feature
- *labelID*. After applying nearest neighbor classification, the newly assigned labeling value of the current feature
- *RGB*. The red, green, and blue values for displaying the current feature after assigning a pseudo RGB color on the screen

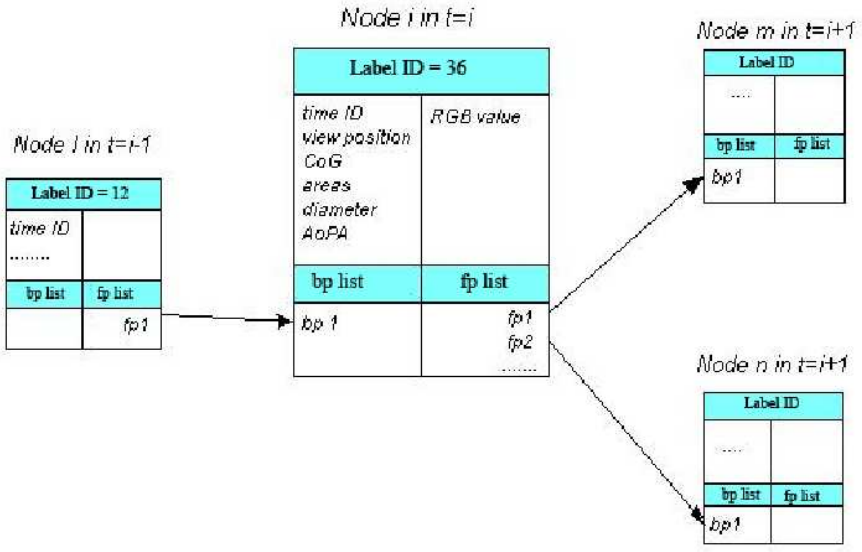


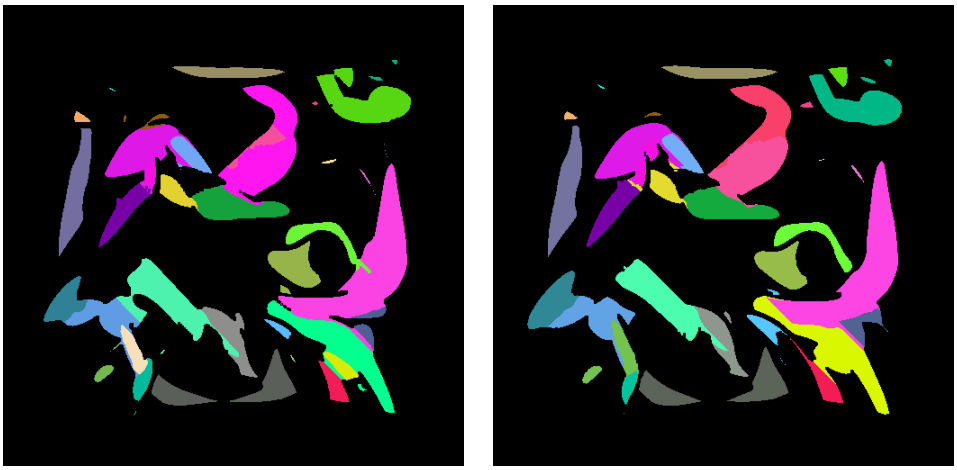
Figure 3.2: Structure for new class "LabelTrack". Illustration of four nodes (features) in three consequential time steps. Each node holds one descriptor vector, preserving the intrinsic properties, and two sub-lists, one containing the key value of the related features in the previous time frame, the other the key value of the related features in the next time frame.

- *Descriptor vector.* A vector which contains the intrinsic parameters to represent a single feature.
- *average gray value.* The average value of the pixel intensity in a certain region, etc.
- *backwardlabel pointer.* A list of feature labels in $step_(i-1)$, they represent the features who are related with the current feature
- *fowardlabel pointer.* A list of feature labels in $step_(i+1)$, they represent the features who are related with the current feature

3.4 Pseudo-color Assignment

The final stage of the system is to display the extraction and tracking results. Our rule to assign the pseudo-color is defined as follows:

- different features in certain time view and position view will be assigned different colors
- *Transformation.* The correspondent features in the following step will be assigned the same color
- *Creation.* New created feature will be colored by the color which has not been used in the current or previous time steps
- *Amalgamation.* The corresponding feature in the next step will be assigned by the pseudo-color from one of its previous features, the first feature which is related to the current feature will pass its color to the current one
- *Bifurcation.* The separated features in the next step will be assigned the same color as the corresponding feature in the previous step for the next step.



(a) (b)

Figure 3.3: Experimental results from two images of two consequent time steps. (a) The tracking result in the first step (b) The tracking result in the subsequent time step

- *Dissipation.* The RGB color will be returned, it can be used again after several steps, but not appear immediately in the following step, see Fig. 3.3.

3.5 Feature Tracking in the Spacial Domain

The processing in the spatial domain data sets is similar to that in the time domain data sets Fig. 3.4, however there is some basic differences as we mentioned in Sec. 3.1

From the above description and the experimental results, we can see that feature tracking based on descriptor vectors works effectively..

(a) (b)

Figure 3.4: Experimental results from two images of consecutive viewing angels with the same time index. (a) The tracking result in the first step (b) The tracking result in the consecutive viewing angle

Chapter 4

Experimental Results and Performance Evaluation

According to the discussions in Chapter 2 and Chapter 3, time varying 3D features extraction and tracking can be achieved automatically based on image-based algorithms. In this chapter, experimental results of features extraction and tracking are exhibited. First, the original data sets are introduced to give a general sense of what they are. Experimental results in the time domain are displayed in Sec. 4.2. In Sec. 4.2.1, evaluations of accuracy are demonstrated, and the performances are compared with manually observed results. Sec. 4.2.2 demonstrates experimental results in the spatial domain.

4.1 Image Formation

The test data sets we used in this project are 2D images which come from projecting the pseudo-spectral simulations of 3-D turbulent volume data with 128^3 resolution onto a virtual "image" plane. The entire pool of test data consists of 50 image-space data sets, with 5 continuous view positions of 10 degrees apart; and for a fixed view position, 10 images are

generated in consecutive time sequence. The algorithm to generate this simulation data pool was based on Silver and Wang's work [26]. In Fig. 4.1, four of the 50 data sets are shown. Note that multiple features are adjacent or overlapped when they were projected onto the image place.

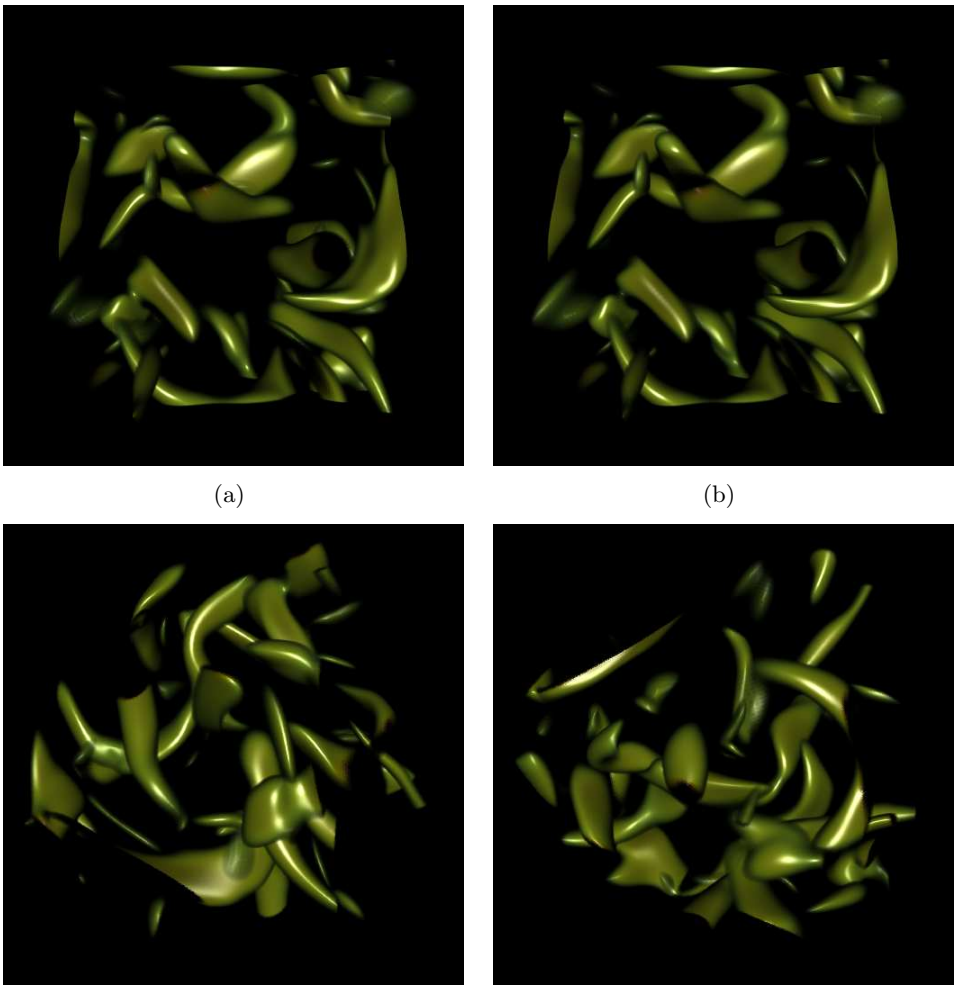
The resolution of each image is 512×512 pixels. To store the image in computer memory, we define brightness as a function of two spatial variables. For each pixel at point (x, y); $x, y \in \Re$, $f_1(x, y)$ indicates the brightness value in Channel 1(RED), $f_2(x, y), f_3(x, y)$ indicate the brightness value in Channel 2, 3(GREEN) and (BLUE) separately. Compared with the original test data sets, the size of image-based data sets has decreased , respectively. Compared with the original test data sets, the size of image-based data set has decreased to some extent, which is further reduced largely by representing the image-based data sets using a descriptor vector.

4.2 Tracking Results in the Time Domain

In this section, we will show the features which have been extracted, tracked, and colored from a sequence of images in the time domain. We show 4 experimental results from 10 frames of image-based data sets.

From Fig. 4.2 and Fig. 4.3, we can see that most of the time-varying 3D volume features from the original data sets have been successfully extracted, tracked and colored. The related features in successive frames have the same color to indicate their relations.

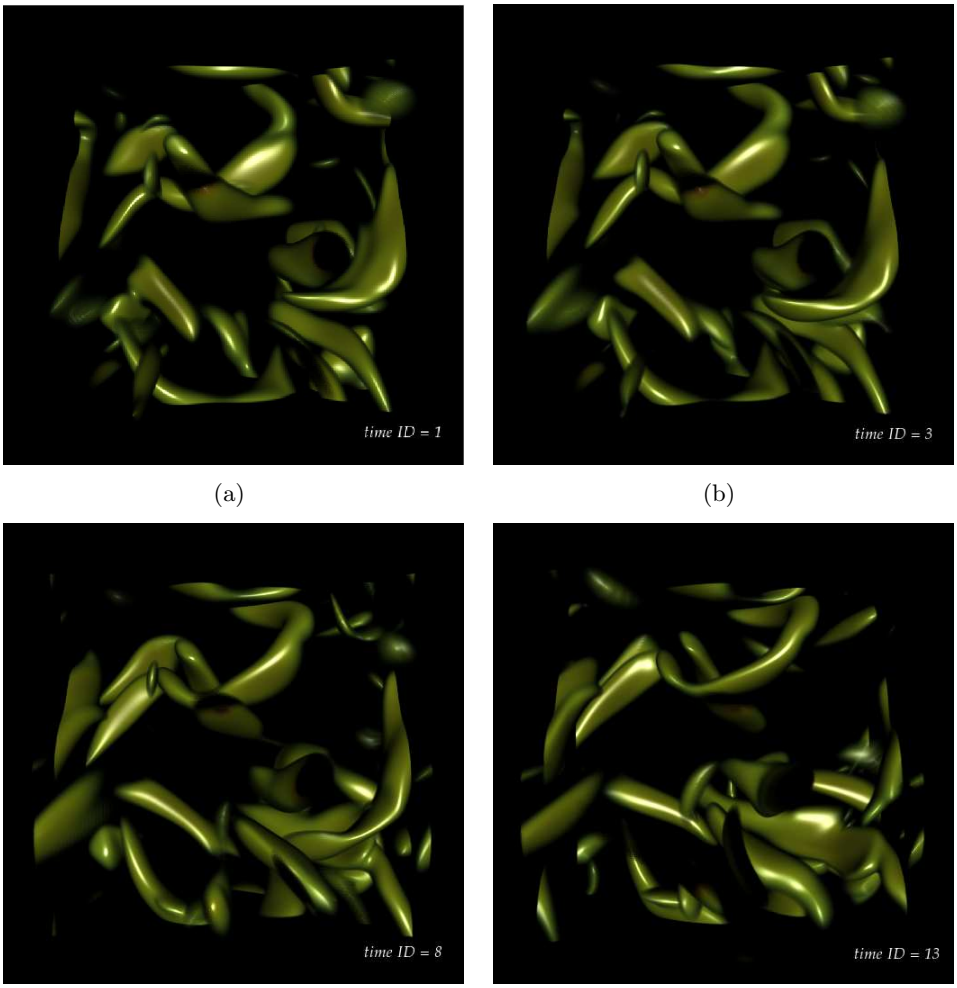
In order to observe the experimental results more closely, we select two features from the same data sets, and track their evolution movements as shown in Fig. 4.4.



(a) (b)

(c) (d)

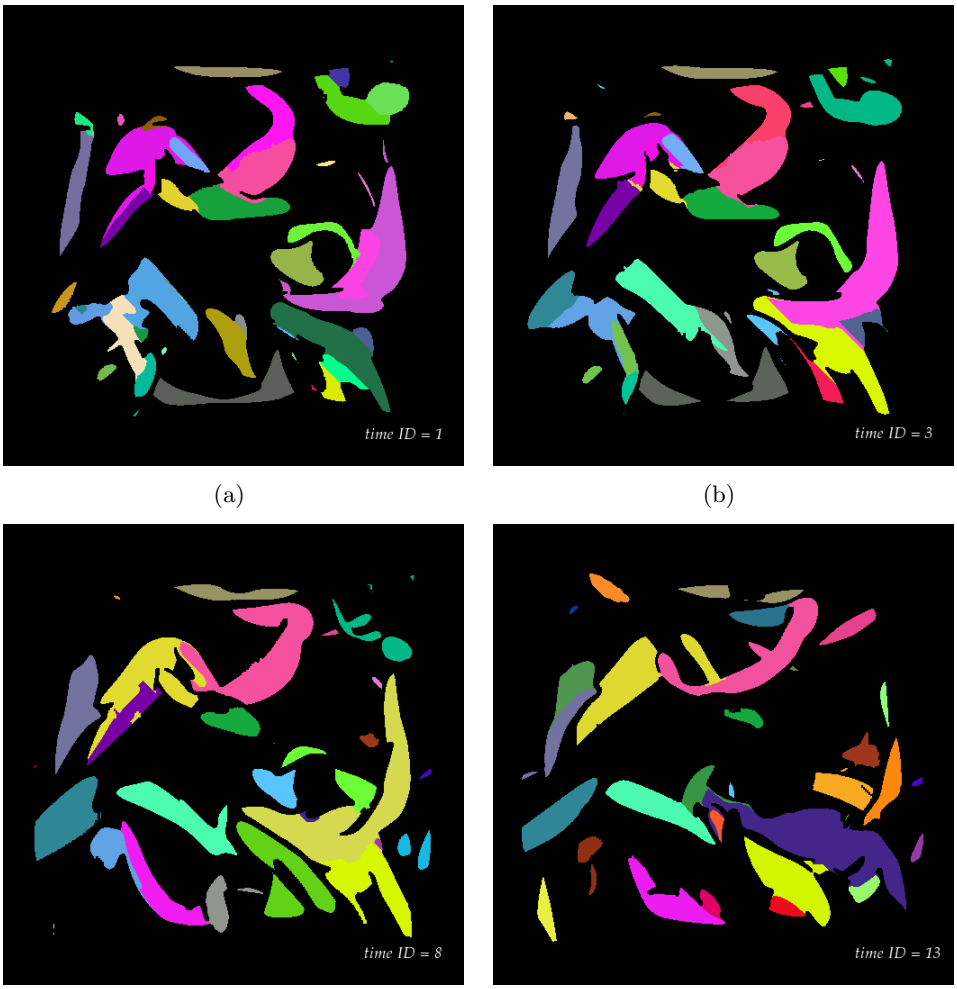
Figure 4.1: The original images selected from the 50 datasets.



(a) (b)

(c) (d)

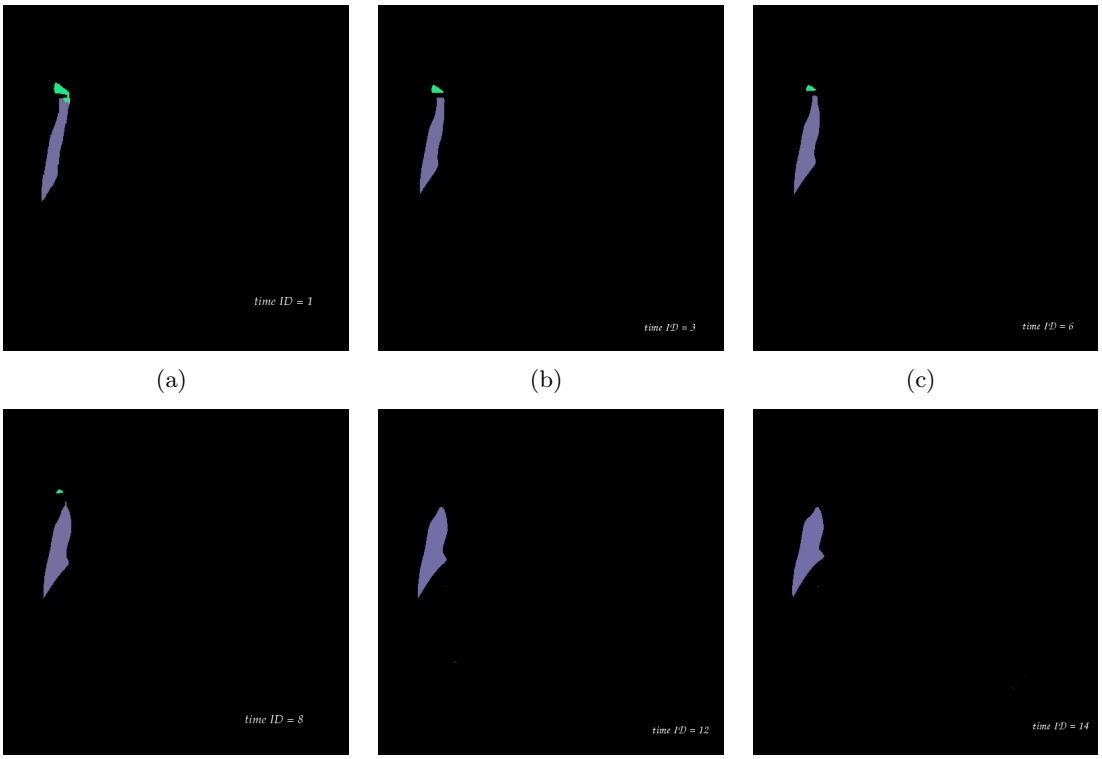
Figure 4.2: Pseudo-spectral simulation data sets in the time domain. The resolution of each projected image in 2D domain is 512×512 pixels.



(a) (b)

(c) (d)

Figure 4.3: Features are tracked and colored.



(a) (b) (c)

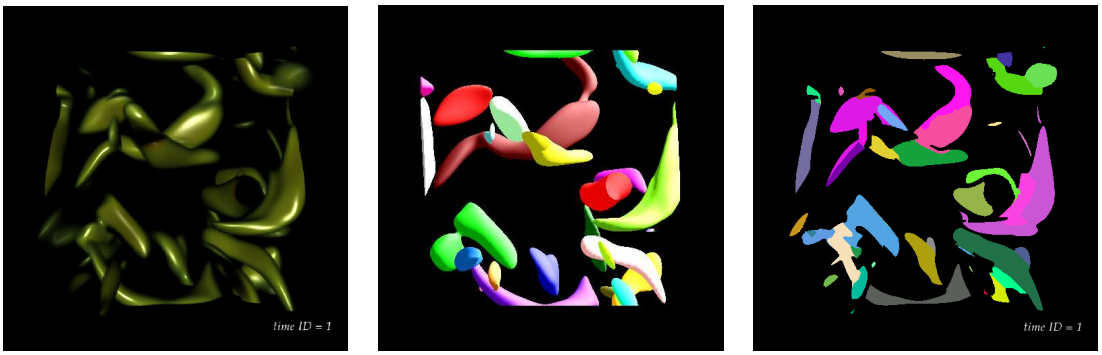
(d) (e) (f)

Figure 4.4: Extract two features from a sequence of time-varying data sets. The blue feature's evolution is continuation; the green feature's evolution is continuation from time=1 to time=8, and then dissipation.

4.2.1 Accuracy

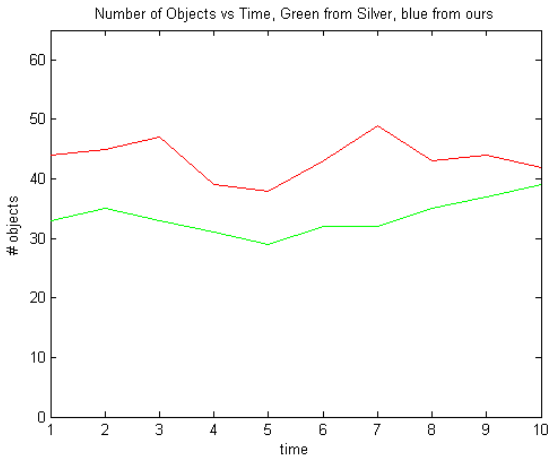
We evaluate the performance of the proposed approach with a substitute algorithm based on Silver and Wang's volume rendering algorithm [26] Silver and Wang's method started with manually selected "seeds" - the local extreme values for each 3D volume feature. Then it used the water-flood algorithm to render each "seed" until they reached their boundaries, and finally computed correspondence based on the velocity overlapping of features between a pair of continuous 3D data sets to tell their relationship. It gives an accurate extraction and tracking result, but it is very sensitive to the *tolerance* distance selection in the correspondence computation stage; moreover, it costs more than one hour to render features from the initial single "seed". The substitute algorithm which has been implemented by SeeLab in Department of Computer Science, UTK is based on ISO-surface rendering. The main procedure is nearly the same as Siver and Wang's method. The only difference is that it started rendering from a surface instead of a single "seed" which gave a higher speed to render features to their boundaries. However the substitute method has its own drawbacks: though rendering features based on an initial selected surface can give a faster speed, for the small features in one data set, their initial surfaces were too small to be observed so that fewer features were extracted and detected. On the other hand, features rendered by surfaces would lose their accuracy in shapes if their real shapes change sharply on the boundaries. While the automatic image-based time-varying feature extraction and tracking method gives the best speed, the accuracy is comparatively low Fig. 4.5. However, it is difficult to conclude which method performs better than the other. To give a more accurate evaluation, we manually extract the features by human eyes from the original data sets, and compare the results with the experimental results from the proposed algorithm.

Fig. 4.6 indicates the number of features extracted from the proposed approach and the number of features which are manually selected by human eyes. One can observe that the



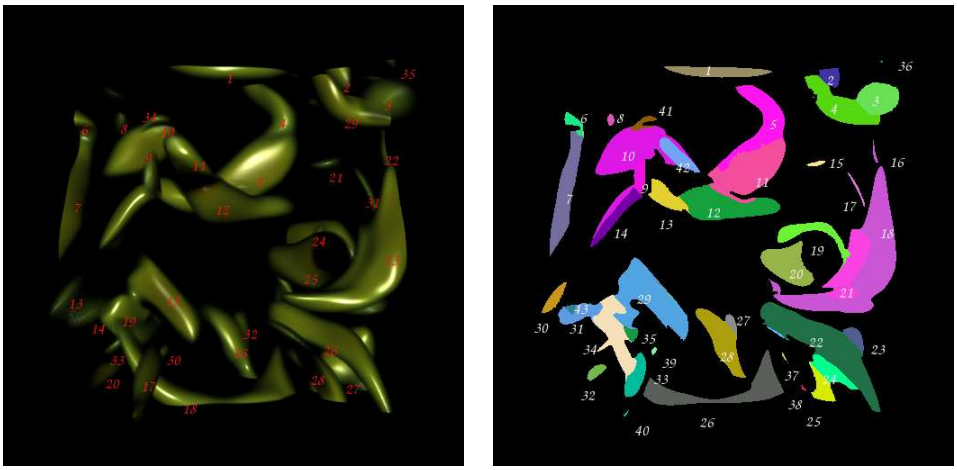
(a) (b) (c)

Figure 4.5: The comparison of results from the iso-surface rendering and the proposed image-based automatic approach. As can be observed, it is difficult to conclude which method is better than the other one. (a) The original test data set. (b) Some obvious features lost in the experimental results based on surface rendering method and the shapes of some features are distorted as well. (c) The experimental result from the proposed image-based method: some features failed to be detected because they were blocked by other features.



(a)

Figure 4.6: Number of time varying features with a fixed camera position in 10 time steps.



(a) (b)

Figure 4.7: Manual accuracy comparison: results generated from the image-base dataset. (a) Features extracted by human beings (b) Features have been extracted by the automatic image-based extraction and tracking system

number from the proposed approach is always bigger than the true number of features in the original data set. The main reason is that the image-based algorithm can not distinguish the features which locate behind those which are closer to the observers.

From Fig. 4.7 and Fig. 4.8, for one single image, 27 out of 35 features have been successfully identified and labeled. The accuracy of this system is 77.1%. Compared with the number of real features, 8 additional features have been extracted. The main reason of this situation is that one feature has been blocked by the features which are closer to the projection plane. When this scenario occurs, one feature will be separated into several features and be labeled with different numbers. This is the main disadvantage of observing the 3D space using monocular position. This shortcoming inspires us not to limit our algorithm in the time domain, but to extend it to the spatial domain so that more accurate results can be obtained. The statistic result of accuracy in 10 successive frames has been shown in Fig. 4.9 and Fig. 4.10.

Real features	Extracted correctly	New features	Error features	Real features	Extracted correctly	New features	Error features
1	Yes	1		2	Yes	2	
3	Yes	3		4	Yes	5	
5	No		11,13,14	6	Yes	6	
7	Yes	7		8	Yes	8	
9	Yes	9		10	Yes	10	
11	Yes	42		12	Yes	12	
13	Yes	30		14	No		43,31
15	No		29	16	Yes	28	
17	Yes	40		18	Yes	26	
19	No		34,35	20	Yes	32	
21	Yes	15		22	Yes	16	
23	No		18,21	24	Yes	19	
25	Yes	20		26	Yes	22	
27	No		23,24	28	No		37,28,25
29	Yes	4		30	Yes	39	
31	Yes	17		32	Yes	27	
33	No		N/A	34	Yes	41	
35	Yes	36					

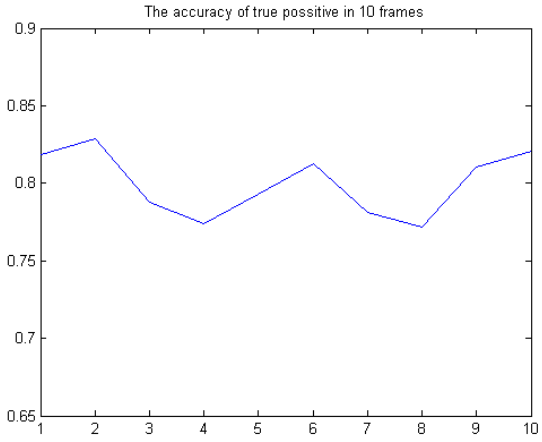
(a)

Figure 4.8: The comparison of original image-base dataset and results obtained from the proposed automatic system. Human labels are given in the first and the fifth columns; computer labeled features in the third and the sixth columns to show whether the real features have been identified successfully or not.

Time steps	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Step8	Step9	Step10
Features in our experiment	44	45	47	39	38	43	49	43	44	42
Features in original dataset	35	35	33	31	29	32	32	35	37	39
Correct	27	29	26	24	23	26	25	27	30	32
Cannot coalescent	4	5	6	6	5	4	5	5	3	4
Cannot distinguish	1	1	1	1	1	2	2	3	4	3
False positive	12	10	14	8	9	11	17	8	7	3
Cannot detected	1	0	0	0	0	0	0	0	0	1

(a)

Figure 4.9: The statistic results in 10 successive frames in the time domain. Manually counting the number of features in the original data set, and the number of features obtained from the automatic feature extraction and tracking system. True positive shows the number of features exist in the original data sets have been correctly detected by the automatic system; False negative shows the number of features exist in the original data sets but have not been detected by the system; False positive indicates the number of features which have been detected by the automatic system but do not exist in the original data sets.



(a)

Figure 4.10: Accuracy ratio in 10 time steps.

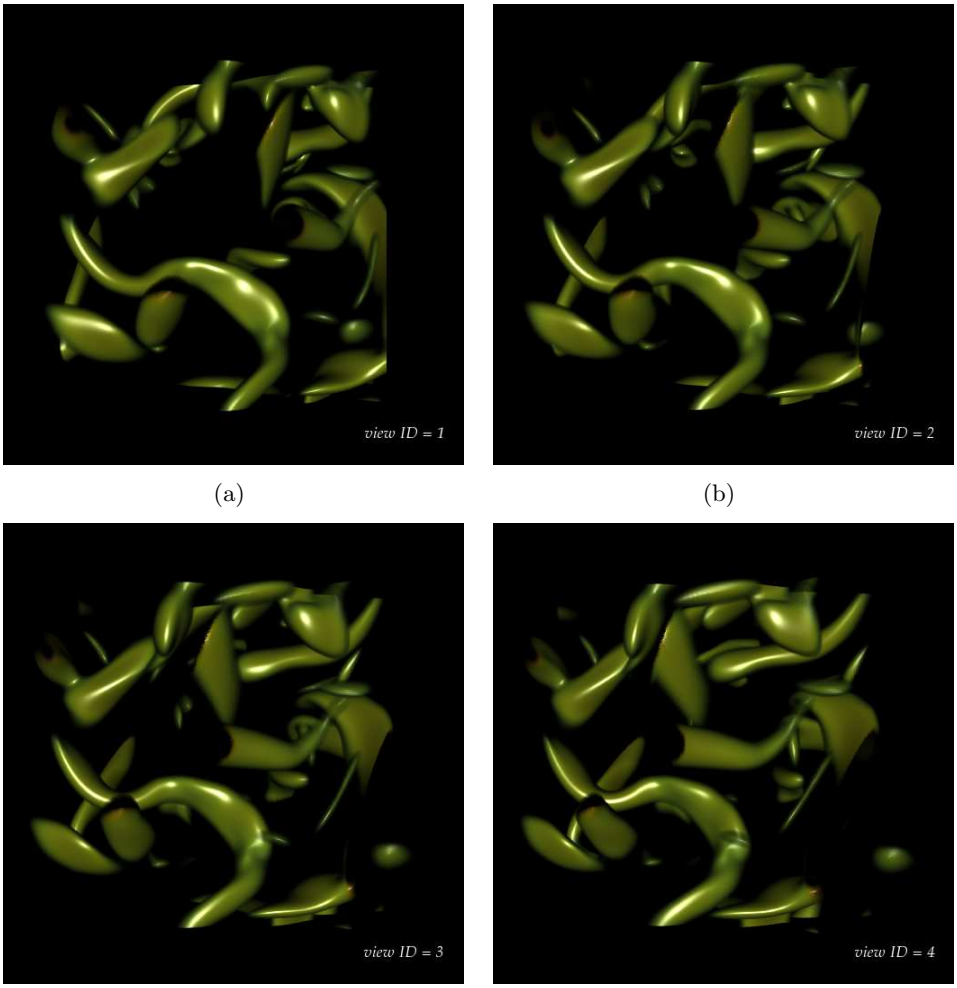
4.2.2 Tracking Results in the Spatial Domain

The processing in the spatial domain is similar to that in the time domain. However, there are some basic differences that we would like to emphasize:

In the spatial domain, we can treat each object as a rigid object since the only parameter changed in the spatial domain is the position of the observing view point, i.e. the camera positions. Therefore, though the similar definitions such as “dissipation”, “creation”, and “transformation”, etc., have been introduced to categorize the evolution events in the spatial domain, the physical meaning is totally different from that in the time domain. Moreover, in the spatial domain, only the features transformed from one spatial domain to the successive spatial domain are saved for further analysis, while other features are discarded. See Sec. 3.1. This essential fact that features will keep their shapes in the spatial domain makes it easier for 3D model reconstruction.

The features who have the evolution events “ bifurcation” and “ amalgamation” will be abandoned in the 3D features reconstruction step because they only indicate that the features have been blocked by others. See Fig. 4.11 and Fig. 4.12.

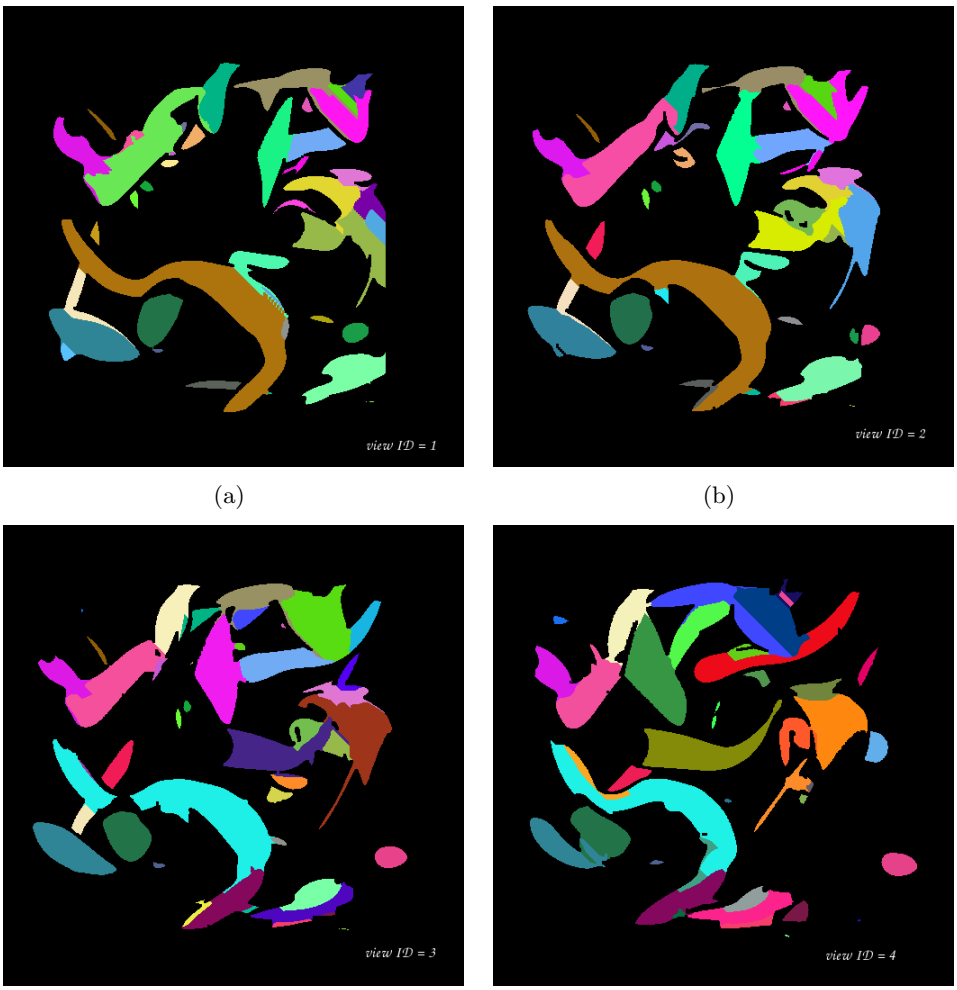
To give a clear understanding of the algorithms and the results from both the time domain and the spatial domain, we select two volume objects in the data sets. Fig. 4.13 shows feature extract and tracking results in both domains



(a) (b)

(c) (d)

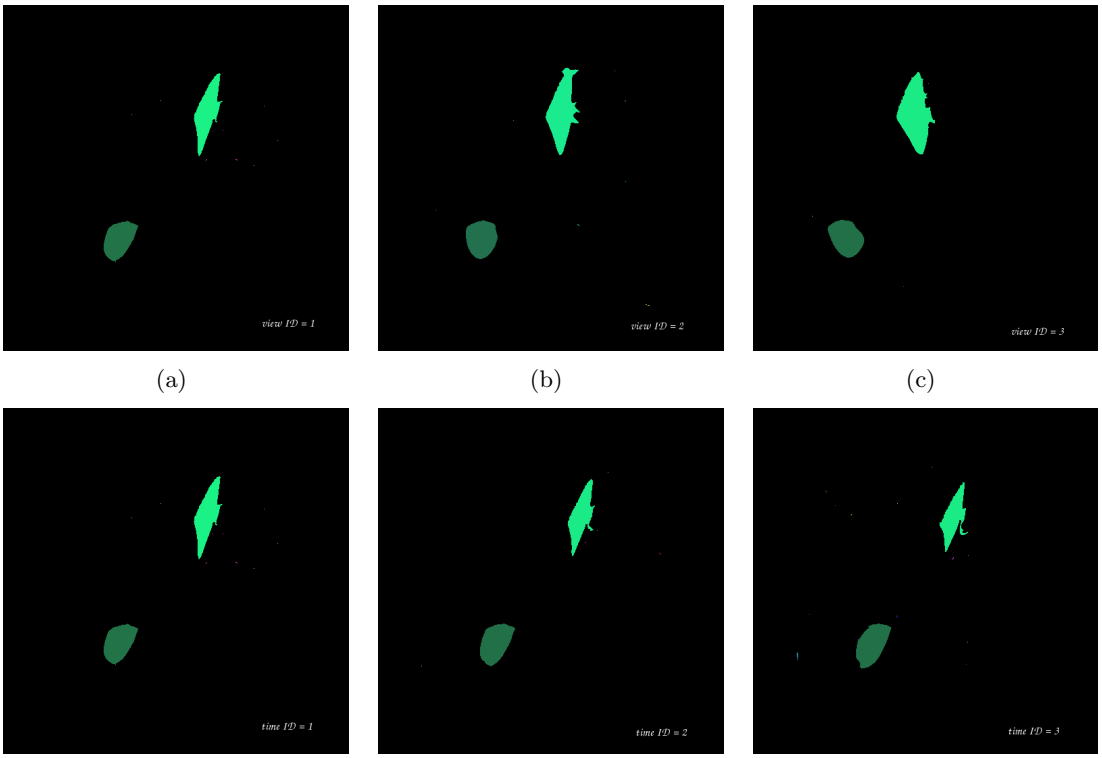
Figure 4.11: The original 4 data sets from different camera positions, at the same time moment. The images are generated with 10-degree viewing angle interval.



(a) (b)

(c) (d)

Figure 4.12: Features tracked in the spatial domain. The corresponding features have the same color in a sequence of images.



(a) (b) (c)

(d) (e) (f)

Figure 4.13: Extract two objects from 5 data sets. The 3 images in the first row show the tracking and targeting result from a sequence of position-varying data sets; the 3 images in the second show the tracking and targeting result from a sequence of time-varying data sets.

Chapter 5

Conclusions

"A journey of a thousand miles begins with a single step."

—Confucius

5.1 Summery of Contributions

In this thesis, we discussed an automatic 3D volume features extraction and tracking method in the image space. This method does not need any prior knowledge about the input data sets. The system can choose parameters automatically depending on the data set it is processing. The experimental results show that it generates high fidelity features extraction and tracking results while consuming less storage and computation power.

5.2 Future Work

First, Advanced image processing algorithms should be applied at the segmentation step, feature selection and extraction step: segmentation based on color images; edge-detection

using morphology image-processing algorithms; more robust features could be calculated to give a better evaluation of the original features.

the accuracy of this automatic system is very good: the average accuracy is around 80%. Except identifying the features which are overlapped together, all the other 3D time-varying features can be extracted and tracked correctly by this system.

Secondly, as we mentioned in the previous chapters, image-based features extraction and detection algorithm can not distinguish the features which are overlapped by others, therefore, though it could give a high accuracy for most of the features, it give a low performance on identifying features which are overlapped together. On the other hand, features extraction and tracking through a single view point is not enough to implement the feature visualization task. it is similar to let the human beings to observe the world by one single eye so that he will lose the depth information and can not perceive the 3D world. To extend our algorithms to the spatial domain gives the possibility to achieve volume visualization task.

Bibliography

Bibliography

[1] J. K. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Articulated and elastic non-rigid motion: A review. *Proc. IEEE Computer Soc.Workshop Motion of Non-Rigid and Articulated Objects*, pages 16–22, 1994.

[2] D. Bauer and R. Peikert. Vortex tracking in scalespace. *In Data Visualization '99. Proc. VisSym'99*, pages 233–240, 2002.

[3] Paul Beardsley, Phil Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. *European Conference on Computer Vision*, 1995.

[4] Z. Chen and H. J. Lee. Knowledge-guided visual perception of 3d human gait from a single image sequence. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 2(22):336–342, 1992.

[5] I. Cohen, N. Ayache, and P. Sulger. Tracking points on deformable objects using curvature information. *Proc. International Conference on Pattern Recognition*, pages 458–466, 1992.

[6] W. C. De Leeuw and R. van Liere. Visualization of global flow structures using multiple levels of topology. *In Data Visualization '99. Proc. VisSym'99*, pages 45–52, 1999.

[7] J. L. Helman and L. Hseeslink. Representation and display of vector field topology in fluid flow data sets. *IEEE Trans. Comput*, 8(22):27C36, 1989.

[8] J. L. Helman and L. Hseeslink. Visualizing vector field topology in fluid flows. *IEEE Transactions on Computer Graphics and Applications*, 3(11):36–46, 1991.

[9] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge. Tracking non-rigid objects in complex scenes. *Proc. International Conference on Pattern Recognition*, pages 93–101, 1993.

[10] H. J. W. in den Haak, and Spoelder and F. C. A. Groen. Matching of images by using automatically selected regions of interest. *Computing Science in the Netherlands*, pages 27–40, 1992.

[11] Aleksandar. Ivanovic and Thomas S. Huang. A probabilistic framework for segmentation and tracking of multiple non rigid objects for video surveillance. *International Conference on Image Processing*, 2004.

[12] Franco. Ivanovic and Carlo. Regazzoni. Adaptive tracking of multiple non rigid objects in cluttered scenes. *International Conference on Pattern Recognition*, 2000.

[13] I. A. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. *Proc. International Conference on Computer Vision and Pattern Recognition*, pages 980–984, 1994.

[14] D. S. Kalivas and A. A. Sawchuk. A region matching motion estimation algorithm. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 2(54):275–288, 1991.

[15] C. Kambhamettu, D.B. Goldgof, D. Terzopoulos, and T. S. Huang. Nonrigid motion analysis. *in Handbook of Pattern Recognition and Image Processing: Computer Vision*, 2, 1994.

[16] S. Kurakake and Nevatia. R. Description and tracking of moving articulated objects. *International Conference on Pattern Recognition*, pages 491–495, 1992.

[17] M. K. Leung and Y. H. Yang. An empirical approach to human body motion analysis. Technical Report 94, University of Saskatchewan, Saskatchewan, Canada, 1994.

[18] W. Liao, S. J. Aggarwal, and J. K. Aggarwal. Reconstruction of dynamic 3d structures of biological objects using stereo microscopy. *Proc. International Conference on Image Processing*, 1994.

[19] D. Marr and E. C. Hildreth. Theory of edge detection. *Proceedings of the the Royal Society*, 207:187–217, 1980.

[20] L.Van Moons, T.and Gool, M. Proesmans, and E. Pauwels. Affine reconstruction from perspective image pairs with a relative object-camera translation in between. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:77–83, 1996.

[21] Frits H. Post, Benjamin Vrolijk, Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. The state of the art in flow visualization: Feature extraction and tracking. *Computer Graphics Forum*, 4(22):775C792, 2003.

[22] R. J. Qian and T. S. Huang. Estimating articulated motion by decomposition. *In Time- Varying Image Processing and Moving Object Recognition*, pages 275–286, 1994.

[23] R. F. Rashid. Towards a system for the interpretation of moving light display. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):574–581, 1980.

[24] F. Reinders and F. H. Spoelder, H. J. W.and Post. Experiments on the accuracy of feature extraction. *In Proc. 9th EG Workshop on Visualization in Scientific Computing*, pages 49–58, 1998.

[25] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing features and tracking their evolution. *IEEE Trans. Comput*, 27(7):20–27, July 1994.

[26] D. Silver and X. Wang. Tracking and visualizing turbulent 3D features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, 1997.

[27] D. Silver and X. Wang. Tracking features in unstructured datasets. *Proceedings of the Visualization Conference*, October 1998.

[28] H. Snyder, W. E.and Qi. *Machine Vision.* Cambridge University Press, 2004.

[29] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 131–147, 1992.

[30] Julius. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles.* Addison–Wesley Publishing Company, 1981.

[31] X. Tricoche, G. Scheuermann, and H. Hagen. Continuous topology simplification of planar vector fields. *Proc. Visualization Conference*, pages 159–166, 2001.

[32] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen. Topology tracking for the visualization of time-dependent two-dimensional flows. *Computer and Graphics*, 2(26):249–257, 2002.

[33] F. Y. Tzeng and K. L. Ma. Intelligent feature extraction and tracking for visualizing large-scale 4d flow simulations. In *SC*, November 12-18 2005.

[34] T. van Walsum. *Selective Visualization on Curvilinear Grids.* PhD thesis, Delft University of Technology, The Netherlands, 1995.

[35] C. Weigle and D. C. Banks. Extracting iso-valued featurews in 4-dimensional scalar fields. *Proc. Symposium on Volume Visualition*, pages 103–110, 1998.

Vita

Lu Zhang was born in Nanjing, P. R. China, on October 19, 1981. She graduated from Nanjing University of Aeronautics and Astronautics, Nanjing, P. R. China in 2003 with a Bachelor of Engineering degree in Information Engineering from the College of Information Science and Technology. During her undergraduate study, she implemented the Virtual Electronic Test System based on the DAQ (data acquired) board NI PCI-5102 from the National Instrument Inc. From June 2003 to July 2004, she worked as a research assistant in Digital Communication Laboratory, Nanjing University of Aeronautics and Astronautics, P. R. China, worked on designing an embedded system, the High Sampling Rate Virtual Digital Stored Oscilloscope. In the fall semester of 2004, she came to The University of Tennessee and enrolled as a graduate student in Electrical and Computer Engineering department. She then joined the Advanced Imaging and Collaborative Information Processing Laboratory of Dr. Hairong Qi with the interests in Advanced Image Processing and Computer Vision. In the future, she would like to pursue a professional career in industry.