Activity Detection in Scientific Visualization

Sedat Ozer, Deborah Silver, Karen Bemis, and Pino Martin

Abstract—For large-scale simulations, the data sets are so massive that it is sometimes not feasible to view the data with basic visualization methods, let alone explore all time steps in detail. Automated tools are necessary for knowledge discovery, i.e., to help sift through the data and isolate specific time steps that can then be further explored. Scientists study patterns and interactions and want to know when and where interesting things happen. Activity detection, the detection of specific interactions of objects which span a limited duration of time, has been an active research area in the computer vision community. In this paper, we introduce activity detection to scientific simulations and show how it can be utilized in scientific visualization. We show how activity detection allows a scientist to model an activity and can then validate their hypothesis on the underlying processes. Three case studies are presented.

Index Terms—Activity modeling, activity detection, activity recognition, simultaneous event detection, Petri Nets, feature tracking, group tracking, time-varying scientific data analysis and visualization

1 Introduction

Today's state-of-the-art simulations generate high-resolution data at an ever increasing rate. Such simulations produce data with billions of mesh points (or voxels) for each time step and thousands of such time steps with multiple variables. Time-varying data can easily reach peta- and exabyte scale. Visualizing these massive data sets is still an ongoing problem. Even after visualizing this data, viewing each variable at each time step is practically impossible in thousands of time steps. Simulations become too complex for the scientist to analyze manually. In such time-varying data sets, scientists want to know *"where and when events happen"* or *"how long an event lasts."* Finding these events in thousands of time steps is not possible with standard visualization tools. What scientists need are routines, procedures, and visualizing techniques to help filter massive data and help focus on areas and events of interest.

In many simulations, a scientist has a hypothesis about events occurring in the data and would like to test and refine his/her assumptions in the hypothesis. Allowing a scientist to model an event and then search for that event over thousands of time steps would both filter massive data and enable scientists to focus on regions of interests in both space and time.

Detection of events has been an active research area in video analysis and there have been a large number of techniques and tools proposed (see Section 2). However, currently there is no tool available for scientists to define, model and automatically search for *complex events*, i.e., activities, in their time-varying 3D scientific data. Most feature-based visualization and analysis routines are still focused on a single time step. Available visualization routines for feature-based time-varying data are mostly concerned with the correspondence problem that involves correlating objects from one time step to the next. These routines do not provide the scientist the ability to model complex spatiotemporal patterns or to answer the fundamental issue of *where, when, and how* "interesting things" occur.

Fundamentally, activity detection is an automated search process of finding a specific and complex pattern (activity) in a large data set containing many different types of patterns. Activity examples include formation of features (such as galaxies, halos, storms or blood clots), anomalous interaction or behavior (anomaly detection), Merge-Split, and ignition events. These patterns are distributed over a large number of time steps. Different instances of the pattern can happen in different durations, i.e., one instance of an event may take 20 time steps and another instance of the same event could take 4.

The problems facing any attempt to localize complex events (activities) automatically in time-varying 3D scientific data can be summarized as how to

1. provide an appropriate way for users to define an event of interest;
2. find an appropriate formalism to model this event;
3. apply the model to detect many instances of the event of interest in simulation data; and
4. present the detected events to users in an appropriate visual form.

In this paper, we introduce activity detection and discuss its applicability for both data analysis and visualization purposes of scientific data with observable features or events. Our main goal is to develop a framework that a scientist can use to first model a spatiotemporal pattern and then search through massive data sets to find instances of such a pattern. The natural way of modeling events or activities is using a graphical and state-based approach that can convert or translate the semantics of an event into a graph-based model. Therefore, in this paper, we propose

- *S. Ozer and D. Silver are with the Vizlab, Department of Electrical & Computer Engineering, Rutgers University, Piscataway, NJ 08854. E-mail: {sedat, silver}@rutgers.edu.*
- *K. Bemis is with the Department of Marine & Coastal Sciences, Rutgers University, Piscataway, NJ 08901. E-mail: Bemis@rci.rutgers.edu.*
- *P. Martin is with the Crocco Lab, Department of Aerospace Engineering, University of Maryland, College Park, MD 20742. E-mail: mpmartin@umd.edu.*

Manuscript received 2 Jan. 2013; revised 20 May 2013; accepted 4 Aug. 2013; published online 19 Aug. 2013.
Recommended for acceptance by R. Machiraju.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2013-01-0001.
Digital Object Identifier no. 10.1109/TVCG.2013.117.

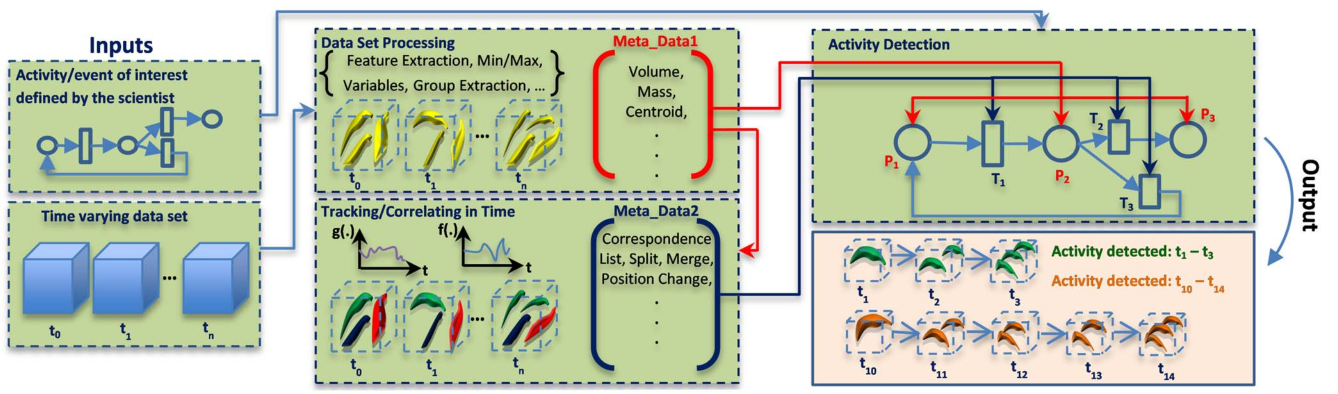

Inputs
Activity/event of interest defined by the scientist
Time varying data set
Data Set Processing
Feature Extraction, Min/Max, Variables, Group Extraction, ...
Meta_Data1
Volume, Mass, Centroid, ...
Tracking/Correlating in Time
Meta_Data2
Correspondence List, Split, Merge, Position Change, ...
Activity Detection
Activity detected: t1 – t3
Activity detected: t10 – t14
Output

Fig. 1. An activity detection framework with Petri Nets: The data come into the framework on the left. The activity or event of interest is defined by the scientist and it is the event that the scientist is interested in searching for within the data set. The way the scientist describes the event is through the use of a Petri Net. The next step is processing the data. In this step features, variable changes, and so on, are computed and stored as metadata. This step also involves correlating features/variables and their changes over time. Once this information is computed it can be searched using the Petri Nets to find an activity of interest. The output of the Petri Net is a list of time steps where the event occurs and a list of features that participate in the activity.

using Petri Nets (PNs) for modeling and searching for events in scientific simulations. An overview of the proposed framework is shown in Fig. 1. First, a scientist defines and models the pattern using a Petri Net and then the instances of this pattern are isolated automatically in the time-varying data set. The result is a list of participating features and specific time intervals, where that specific pattern occurs.

Our contributions in this paper include introduction of the concept of activity detection for scientific visualization, the use of Petri Nets to model and detect activities in scientific visualization, an enhancement of Petri Nets to include the dynamics of scientific phenomena, and demonstration of the use of activity detection on three different 3D time-varying data sets as case studies.

1.1 Definitions and Use-Case Scenarios in Scientific Data

An *event* is a spatial and temporal (spatiotemporal) pattern that happens over a course of time steps and can include the interactions of different types of objects. Events can be further divided into two groups: atomic events (*actions*) and complex events (*activities*).

An *atomic event* or *action* is a primitive event that is easier to define and detect and can be inferred by comparing the current time step to a specific (reference) time step. Actions usually occur between two consecutive time steps, such as merge, split, birth, and death [1]. A *complex event* or an *activity* is an event that spans multiple time steps and includes multiple object types, object states or object interactions. Complex events usually occur over more than two time steps. An *actor* is an object that participates or is involved in an activity. In this work, we will use the terms *actor*, *object* or *feature* interchangeably to refer to a region of interest that performs the activity or is participating in the activity. We will use the terms *metadata* or *attributes* to describe a set of computed quantitative properties of objects.

Activities are common interests in video analysis as in [2], [3], [4], [5], and [6]. For example consider a security surveillance system at an airport. There are hundreds of locations and thousands of hours of video that must be monitored. One situation of interest to security personal is a "left-baggage" activity where a person walks in with a bag, puts the bag down and then walks away without the bag. A model of this activity is shown in Fig. 2. Searching for that sequence in hours of video footage (and real time) is a crucial task.

Fig. 2 illustrates the activity "a walking person leaves a bag unattended" as a sequence of key and atomic (primitive) events without considering all the possible configurations. Similar to Fig. 2, there are many cases where the scientists are looking for complex events of features. Examples are: formation of a packet [7], formation of a galaxy as in [8], and combustion events as in [9]. Three specific examples we use in this paper include "Anomalous Plume Bending," "Packet formation," and "Merge-Split." In oceanography, scientists are interested when an anomaly occurs, i.e., unusual changes in direction or magnitude of the bending of the plumes in response to local ocean currents (Anomalous Plume Bending). As another example [7], in wall-bounded turbulent direct numerical simulations (DNS) there are hundreds of hairpin vortices (features) interacting with each other in each time step. One event scientists are interested in is finding when packets form, i.e., "Packet Formation" and its duration. (A packet is a group of features acting coherently). This event involves numerous "young" hairpin vortices that come together and eventually form a group (packet). The third example is the "Merge-Split" activity in which first multiple features merge to form a single feature, and then this single feature splits into multiple features again within a specified time interval. Similar events have been described in [34] and [47] previously.

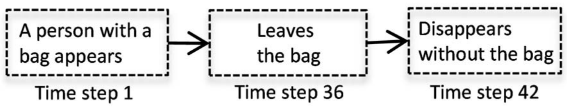


Fig. 2. A simple sequence of an activity (left-baggage) formed of three different key atomic events (actions) over the course of 42 time steps.

TABLE 1
Ontology of Coherent Structures and Their Evolution

Objects			Move		Interact		
					Come together / Apart		
Bubble	Finger	Pint	Advect	Stream	Accrete	Disassemble	Pair
Blast wave	Gyre	Ring	Entangle	Swirl	Aggregate	Disrupt	Plow
Blob cloud	Hairpin	Roll	Disperse	Transport	Align	Finger	Reflect
Critical pt.	Helix	Separatrix	Flow	Wind	Bind	Fission	Scatter
Eddy	Hole	Spike	Hop	Bend	Bifurcate	Focus	Spike
Favor	Packet	Spiral	Migrate		Breakup	Fold	Split
Filament	Patch	Striation			Burst	Fuse	Striate
Shockwave	Plume	Vortex			Collapse	Roll-up	Strip
					Condense	Wind about	Merge

Many other domains have the need for detection of activities such as blood clot formation in blood flow simulations [10], magnetic storm formations in space-weather simulations [11], and extinction and reignition in turbulent flames in combustion simulations [12]. All these examples involve *actors* performing *activities* and all these activities can be modeled by using a common formalism. Activities occur over multiple time steps in which feature attributes or feature types may change over a course of an experiment/simulation. The duration of the occurrence of an activity usually changes from one instance of the activity to the next. For example, one such occurrence could happen over 10 time steps and another could take 100 time steps. *The ability to model an activity considering such variance in durations is necessary prior to detecting the activity.* The natural and best way of describing such activities is via the key feature states and key atomic (primitive) events (i.e., actions) that characterize the complete activity. Once the key actions and feature states are determined, we can express such complex events as a sequence (or a combination) of these feature states and actions. In such a sequence model, the detection starts as soon as the first primitive event of the sequence occurs and completes with the occurrence of the final primitive event. In a graph-based technique, each node represents one of these key atomic events or feature states. A Petri Net is such a graphical technique that uses nodes as feature states or their primitive events (See Section 3).

Table 1 categorizes some of the available domain specific terms based on objects (actors, features), and their interactions/atomic events for general computational fluid dynamics (CFD) simulations. These are the events that can describe the actions or the entire activity depending on the complexity of the interaction in CFD domain and these are the types of events that can be modeled for activity detection. Each of these listed items can represent a node in a Petri Net model of an activity.

Similar to Table 1, a library formed of common atomic event definitions and feature types can be created and then can be used to describe more complicated activities for scientific simulations. Such a library would also help scientist to describe how to express their domain specific actions and features. Moreover, meaningful combinations of such object and interaction types could yield exploratory knowledge discovery.

Activity detection can be utilized in many other different ways in scientific visualization. Example applications are

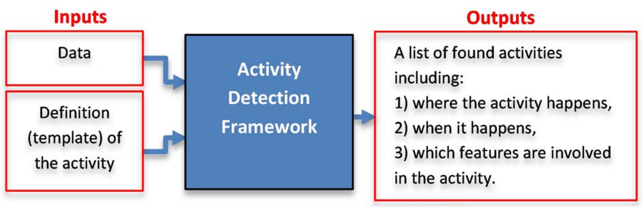

Inputs
Data
Definition (template) of the activity
Activity Detection Framework
Outputs
A list of found activities including:
1) where the activity happens,
2) when it happens,
3) which features are involved in the activity.

Fig. 3. Inputs and outputs of a typical activity detection framework, the definition of the activity comes from the domain knowledge.

model or hypothesis validation in simulations, time-varying data analysis, and time-varying transfer function generation.

We present the related activity detection work in the next section. In Section 3, first we discuss Petri Nets and the enhancements needed to operate within a scientific environment. We then describe the implementation and how visualization can be enhanced with this approach. In Section 4, three case studies are given to demonstrate how Petri Nets can be used to identify events within time-varying data and we conclude this paper in Section 5.

2 Related Work

Detection of activities is a fundamental and important step of fully automated intelligent systems and applies to many fields where the data come from various types of sensors including cameras, medical devices, and sensor networks as in [13], [14]. It has been an active research area under the "activity detection" name, although the terms "activity recognition," "action detection," and "action recognition" have also been used interchangeably, (see the survey [15]).

Activity detection is related to data mining. *Data mining* is the process of finding new and nontrivial information within a data set. This information can be in the form of a pattern, a model of the process that generated the data set or the correlation information between the available variables. Activity detection specifically deals with the complex patterns that are in the form of sequences (or combinations) of simpler patterns. A successful data mining technique requires domain knowledge [16]. This is also true for a generic activity detection framework. Inputs and outputs of a generic activity detection framework are shown in Fig. 3.

While supervised and unsupervised techniques can also be used for activity detection, they do not provide an intuitive and easy way to express a hypothesis. Our focus in this paper is presenting a technique that allows a scientist to specify an event and search for it. Therefore, in this paper, we propose to use Petri Nets to express and model an activity. A Petri Net model is a graph abstracting an activity. Therefore, the same Petri Net model (graph) can be used in different simulations with different parameters. Compared to the learning-based data mining techniques, their performance depends on the model description as opposed to forming new training data. Choosing between learning-based algorithm and graph-based algorithm balances a tradeoff between needing additional data for training and needing to specify an accurate description. A graph-based approach is more likely to meet our objective of enabling hypothesis testing.

As a framework, activity detection involves numerous steps including segmentation, tracking, and computing the feature attributes. Therefore, in this section, first we review activity detection in computer vision and then we give a list of the available segmentation and tracking techniques in scientific visualization.

2.1 Activity Detection in Computer Vision

In computer vision, the activity detection problem is treated with different approaches including clustering, machine learning or semantic-based techniques using rules or graphs such as Petri Nets. All of these approaches require the domain knowledge in different forms. For example, in the case of machine learning, the domain knowledge is embedded within the training data. The learning process and the accuracy of the technique depend on the validation. Validation requires *ground truth*, which is a set of expected or real outputs (labels) from the technique. Picking (or generating) the ground truth or the training data is equivalent to manually picking and labeling multiple instances of the activity within the data set. This is almost the same as exploring the data manually in each time step and, therefore, is an extremely time consuming task [17].

Semantic-based activity detection approaches simply generate a model of the given activity from its description. These approaches do not require training data or a certain type of cost or similarity function to be derived. Instead, the domain expert defines the activity in a sequential form in a timely manner and this description is used to search for the event. The sequential form can be in the form of a set of rules (as in [18]) or a graph (as in [19], [20], [21], [22]). Both rule-based and graph-based techniques are fundamentally logic-based (*if-then* based) techniques. Among those, the graph-based techniques use a state-based approach in which the various stages of an activity are described as the individual nodes (e.g., finite-state machines, Petri Nets). Petri Nets encompasses both rule-based techniques and finite-state machines (a finite-state machine is a subclass of Petri Nets, see Section 3). Using the Petri Net formalism, a scientist models an activity in a graphical fashion where objects in the simulation pass through different stages on the way to being classified as an activity. Once the graphical model is available, a Petri Net algorithm evaluates the model to search for the instances of the activity automatically. (This is analogous to knowledge-assisted visualization [25].)

In computer vision, activity detection applications mainly focused on human-related activities. These include interactions or relations between: humans and humans [26], humans and vehicles [27], humans and websites [28] or certain human behaviors or their situations in certain environments as in [13]. Various techniques including Bayesian techniques, hidden Markov models, and conditional random fields are used to "learn" and detect activities in such examples. A detailed list of available activity examples and detection techniques can be found in the review papers [3], [15], [29], and [30]. Recently, Petri Nets gained the attention of researchers in both data mining and activity detection communities as in [2], [5], [19], [20], [21], and [22]. This is due to the fact that Petri Nets can be used as a natural way of modeling semantic descriptions of activities or events.

While all the activity detection related Petri Nets works aimed to work with video data, they do not incorporate the "dynamic" properties of the time-varying environment (e.g., split, merge, appear, and disappear events) within the Petri Net formalism. Some of the above-mentioned treatments such as timed Petri Nets or Stochastic Petri Nets still lack supporting the dynamics of a "time-varying" system.

2.2 Related Work in Visualization

Event visualization in video data has been studied in the visualization community. For example, Botchen et al. [4] presented a video visualization technique, VideoPerpetuoGram, for action visualization in video data. In their method, they treated the stacks of 2D time-varying video data as 3D volume data and visualized actions in such volume data. Parry et al. [6] presented a hierarchical event selection algorithm for event visualization in video data and applied their method on snooker video. A list of available applications and techniques can be found in the survey paper [31]. Woodring and Shen have used a combination of wavelet transforms and clustering to detect and visualize the temporal trends in time-varying data sets in [32]. Dou et al. [33] detected and visualized interesting events (trends) in text data. However, none of these papers allows scientist to model a specific scientific activity.

Related work in feature-based visualization includes extraction and tracking. Extracting atomic events and features from time-varying data has been widely studied in scientific visualization. This information is crucial to activity detection and provides the metadata input to an activity detection framework. Feature extraction extracts the objects (actors) within the time-varying data set for activity detection. In different domains, appropriate extraction tools can be used to define and extract features as in [1], [9], [34], [35], [36], [37]. Various feature tracking models have been proposed to track features and to visualize their evolution over time as in [1], [9], [38], [39], [40], [41], [42], [43]. Besides extracting features and tracking them, recently, Ozer et al. [44] proposed a group tracking model that also extracts the groups of features in the data and then tracks them over time. In their paper, a list of useful attributes is also provided. Such a list of attributes can be used to define a feature's state or an action.

In general, the activity detection framework that we propose in this paper can use all of the available segmentation and tracking techniques in scientific visualization. In the next section, we discuss Petri Nets and the enhancements needed to operate them within scientific environment.

3 Petri Nets

In this paper, we introduce using PNs for detecting activities in 3D scientific data sets and we further enhance them for feature-based scientific data processing and visualization. Petri Nets are graph-based techniques that can model and visualize various behavior types including parallelism, concurrency, resource sharing, and synchronization [23]. A Petri Net is a finite-state machine that allows multiple inputs and multiple outputs (A traditional finite-state machine is a Petri Net in which each transition is

TABLE 2
Examples of Token, Place, and Transition in Scientific Simulations

Token	Place	Transition
An instance of an object, entity or a variable type **Examples**: Bubble, plume, cell, pressure, temperature, cloud, hairpin, vortex, packet, eddy, A certain group of features.	A type of an object state **Examples**: "bar shaped vortex", "small feature", "Bent plume", "45° bent plume", "single feature", "A packet", "an eddy", "a group of neighbour features".	Action or a condition **Examples**: "exactly *n* features merge", "feature splits", "feature changes its group", "features elongate", "feature rotates less than 45°", "Volume difference is less than %5".

restricted to have exactly one output and one input [24]). The more general property of allowing multiple inputs and outputs makes Petri Nets an ideal candidate for activity and hypothesis modeling in scientific simulations.

In this section, we assume that all the features have already been segmented, tracked, and their attributes are available. Then, we can say that activity detection with Petri Nets has two fundamental steps: 1) model the activity of interest (i.e., modeling); and 2) create a data structure for the modeled Petri Net and run the algorithm over the time-varying data (i.e., execution). This is analogous to first training a machine learning algorithm and then running such algorithm with the learned parameters on the test (or new) data. The activity is modeled by the scientist for activity detection with Petri Nets.

A typical Petri Net model consists of places, transitions, and directed arcs. Places (circles in a Petri Net graph) represent the types of possible object states in the domain. Transitions (rectangles in a Petri Net graph) sit between places and represent conditions or actions. Directed arcs (the edges) define the connections between places and transitions in a Petri Net model.

Execution of a Petri Net is done by means of tokens (solid dots in a Petri Net graph). A token can represent an instance of an individual feature or an instance of a set of features. Token examples include "a vortex," "a hairpin," "a bent plume," "a big eddy" or "a set of features" (such as a packet as in [44]). Various examples of tokens, places, and transitions can be found in Table 2. The location of a token describes the current status of an activity in a given Petri Net model. This can be imagined as tokens flowing from one place to the next and the transitions act like the gates controlling this flow by being open or closed.

Fig. 4 shows an example Petri Net model of the "left-baggage" scenario. The complete activity is decomposed into key actor states (*Person, Person with bag, Person without a bag*) and actions (*carries a bag, walks, leaves the bag, picks up the bag, person disappears*). Then, the activity is modeled graphically as a sequence of these actor states and actions. A token in this Petri Net represents a person from a video surveillance data and the token's location shows the current status of the person in the Petri Net.

A typical PN is the tuple $\mathbf{PN} = (\mathbf{P}, \mathbf{T}, \mathbf{I}, \mathbf{O}, \mathbf{C_P}, \mathbf{C_T}, \mathbf{S}, \mathbf{E}, \mathbf{M})$.$\mathbf{P}$ is a set of places such that $\mathbf{P} = \{P_1, P_2, P_3, \ldots, P_n\}$, $\mathbf{C_P} = \{C_{P1}, C_{P2}, C_{P3}, \ldots, C_{Pn}\}$ a set of place conditions,

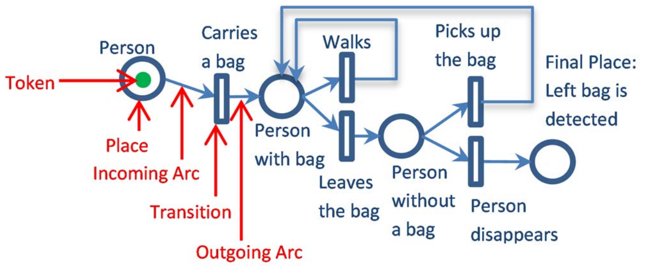


Fig. 4. A Petri net model for the left-baggage example.

where C_{Pn} is the place condition for the place P_n, $\mathbf{C_P} = \{C_{P1}, C_{P2}, C_{P3}, \ldots, C_{Pn}\}$ a set of place conditions, where C_{Pn} is the place condition for the place P_n, $\mathbf{T} = \{T_1, T_2, T_3, \ldots, T_m\}$ a set of transitions, $\mathbf{C_T} = \{C_{T1}, C_{T2}, C_{T3}, \ldots, C_{Tm}\}$ a set of transition conditions, where C_{Tm} is the transition condition for the transition T_m.

In Petri Nets, a place can only connect to transitions and transitions can only connect to places through directed arcs (see Fig. 4). Directed arcs are further categorized into two subcategories namely incoming arcs and outgoing arcs. The set $\mathbf{I}$ defines the set of incoming arcs and the set $\mathbf{O}$ defines the set of outgoing arcs. An incoming arc describes the connection from a place to a transition and an outgoing arc describes the connection from a transition to a place. The set of all the incoming arcs is usually defined in a matrix form such that $\mathbf{I}(j,i) = w$ is the arc weight from jth place to the ith transition. Similarly, the set of all the outgoing arcs is usually defined in a matrix form such that $\mathbf{O}(j,i) = w$ is the arc weight from ith transition to the jth place. When the arc weight (w) is not specified, it is assumed to be one. The state of a Petri Net is defined by the marking $\mathbf{M} = \{\mu_1, \mu_2, \mu_3, \ldots, \mu_n\}$, where μ_n is the number of tokens in place P_n. A Petri Net is executed based on the number of tokens in each place and the object attributes with the *firing* rule [23], [24]. Therefore, tokens determine the state of a Petri Net. $\mathbf{S} = \{S_1, S_2, S_3, \ldots, S_r\}$ is the set of initial (starting) places where the activity starts ($r < n$) and $\mathbf{E}$ is the set of final places where the activity ends.

A transition T_i is *enabled* if and only if there are enough tokens in each input places, i.e., $\forall j : \mathbf{M}(j) \geq \mathbf{I}(j,i)$ and if the C_{Ti} is satisfied. *Firing* an enabled transition is the process of moving the tokens from the incoming places to the outgoing places. Firing a transition T_i consumes (removes) $\mathbf{I}(i,j)$ tokens from each of its input place i, and produces $\mathbf{O}(j,i)$ tokens in each of its output places j. Thus, the new markings can be computed with the following equation:

$$\mathbf{M}_k = \mathbf{M}_{k-1} + (\mathbf{O} - \mathbf{I})\mathbf{E}_k, \tag{1}$$

where $\mathbf{E}_k$ is the vector representing the transitions that are fired at time step k [21]. $\mathbf{E}(i) = 1$ if T_i is fired and 0 otherwise, where $i = 1, 2, \ldots, m$. See [45] for more details.

In computer vision, actor states and actions are identified by using several low-level detection, segmentation, and classification techniques. In scientific visualization, feature or group extraction extracts actors and computes various attributes for each extracted actor in the data, and feature (or group) tracking correlates the actors in time. Attributes are used to define places (actor states). Both attributes and

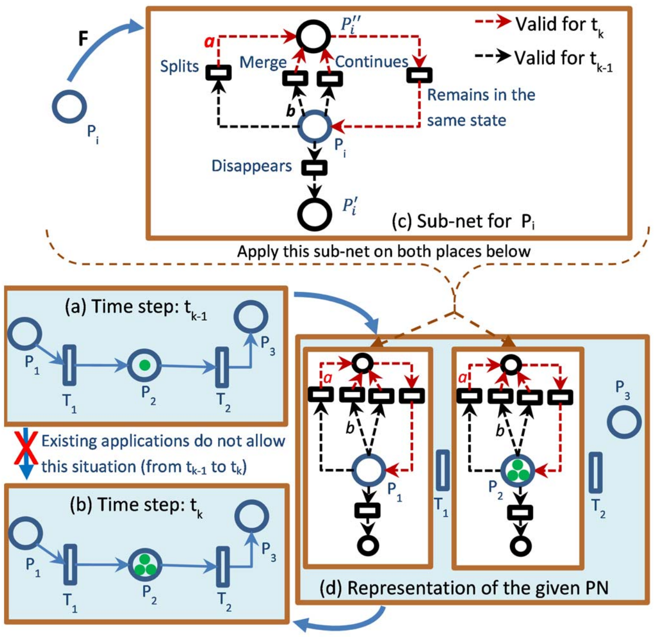

F
P_i
P_i''
a
Splits
Merge
Continues
b
Remains in the same state
Disappears
P_i'
Valid for t_k
Valid for t_{k-1}
(c) Sub-net for P_i
Apply this sub-net on both places below
(a) Time step: t_{k-1}
P_1 T_1 P_2 T_2 P_3
Existing applications do not allow this situation (from t_{k-1} to t_k)
(b) Time step: t_k
(d) Representation of the given PN

Fig. 5. (a) A scientist given Petri Net and its marking at the time step t_{k-1} is shown, (b) the marking of the PN changes in t_k since the token splits into three objects. However, because there is no transition is fired, the marking cannot be changed in traditional Petri Nets; (c) to solve this, each place is replaced with a sub Petri Net (subnet) in TTPN. The subnet can model and allow the change in the marking of PN during the transition from t_{k-1} to t_k. (d) Each place (except the final place) in the PN is replaced with this subnet and once the execution of the subnet is completed for each place, the update process is completed and the initial marking is obtained in t_k.

tracking information are used to define transitions (actions) in a Petri Net.

3.1 Token-Tracking Petri Nets (TTPN)

In many scientific simulations, there are multiple features and these features interact with each other. The total number of features changes in each time step because of the merge, split, continuation, birth and death events [1]. Petri Nets can be used to model "simple" merge and split events, where the number of merging (or splitting) features is fixed or known in advance. However, this is not the case in scientific simulations because the number of merging (or splitting) features change from one instance to the next and this information becomes available during the runtime (this is illustrated in Figs. 5a and 5b). Therefore, modeling such variability through the arc functions of the typical Petri Nets (including colored Petri Nets) is difficult. In addition, the time variance in Petri Nets has been limited to expressing the duration-based conditions (mostly for transitions) in timed Petri Nets and probabilistic Petri Nets [23], [24]. There is no Petri Net formalism proposed in activity recognition applications to update a token or the state of a Petri Net as a function of time.

We summarize the existing problems in Petri Net applications to model activities in scientific simulations as follows: 1) tokens (i.e., object attributes) change from time to time and existing Petri Net applications do not consider such time variance in a place; 2) the number of merging and splitting features varies from one instance to another and this number is computed during runtime. Therefore, it is difficult for the scientist to include such variability in a Petri Net model in advance; 3) tokens disappear or change their state from one time step to the next and such variability should also be considered in the Petri Net formalism even if these situations are not explicitly modeled in a given Petri Net model.

In this work, we enhance Petri Nets to handle above-mentioned issues existing in the previous activity detection works and call our enhanced Petri Net *token-tracking Petri Nets*. TTPN consider the feature dynamics by updating the tokens and their places automatically as the time changes. This is done by coupling the Petri Net with the tracking information. Therefore, the state of a TTPN is a function of time and is described by the tuple $(\mathbf{M}^+_{\boldsymbol{k}-1}, F)$, where $\mathbf{M}^+_{\boldsymbol{k}-1}$ is the final marking (see the next paragraph) obtained in the previous time step t_{k-1} and $\mathbf{F}$ is the updating function. The updating function $\mathbf{F}$ maps the existing tokens in a Petri Net from t_{k-1} to the extracted tokens in t_k such that

$$\mathrm{F}(\mathbf{M}^+_{k-1}) = \mathbf{M}^-_k, \tag{2}$$

where $\mathbf{M}^-_k$ is the initial marking in t_k. Therefore, the final marking of a TTPN at t_k can be computed by the tuple $(\mathbf{P}, \mathbf{T}, \mathbf{I}, \mathbf{O}, \mathbf{C}_P, \mathbf{C}_T, \mathbf{S}, \mathbf{E}, \mathbf{M}^+_{\boldsymbol{k}-1}, \mathbf{F})$.

Similar to colored Petri Nets [24], each token has an ID in TTPN. Therefore, the marking $\mathbf{M} = \{\mu_1, \mu_2, \mu_3, \ldots, \mu_n\}$ summarizes the distribution of tokens in a given TTPN. μ_n is the set of tokens in place P_n such that $\mu_n = \{\mathbf{X}_1, \mathbf{X}_2, \ldots\}$. A token $\mathbf{X}_a$ is n tuple such that $\mathbf{X}_a = (x_{a1}, x_{a2}, x_{a3}, \ldots, x_{an})$ where x_{an} is the nth attribute of the token $\mathbf{X}_a$. The initial marking $\mathbf{M}^-_{\mathbf{k}}$ of a Petri Net is the marking that passed on from the previous time step t_{k-1} and the final marking $\mathbf{M}^+_{\boldsymbol{k}}$ is the marking where all the enabled transitions have fired such that there is no further enabled transition remains in the time step t_k.

Once all the tokens and their places are *updated* via the function $\mathbf{F}$, the next step is evaluating the Petri Net by firing all the enabled transitions for each token. Similar to typical Petri Nets, firing is done by using (1) for a given token $\mathbf{X}_a$ such that

$$^{Xa}\mathbf{M}^+_k = {}^{Xa}\mathbf{M}^-_k + (\mathbf{O} - \mathbf{I}).\mathbf{E}_k, \tag{3}$$

where $^{Xa}\mathbf{M}^+_k$ represents the new location of the token $\mathbf{X}_a$ in the Petri Net at t_k. The same token needs to be in the all input places to enable a transition and the transition condition should be satisfied. In a given TTPN model, each arch weight is considered 1. This simplifies the process for the scientist because they do not consider the arc weights (also called arc functions) to model a hypothesis or an activity. Furthermore, TTPN considers the dynamics of the system (i.e., merge, split, appear, disappear, and continuation) internally, and therefore, a scientist does not need to consider these events to model. This process incorporates the time variance in Petri Nets and simplifies the modeling of an activity (see Fig. 5c). Fig. 5 illustrates the overview of how TTPN works. Consider the given Petri Net model with its tokens in time step t_{k-1} with its existing tokens in Fig. 5a. The green token (e.g., a feature) in P_2 splits into three tokens (e.g., three features) in time step t_k (shown in Fig. 5b). However, existing Petri Nets do not allow a token split while waiting in the same place. This situation is handled in TTPN by using subnets. A subnet is a Petri Net in which the time information is attached to both tokens and the arcs. In TTPN, each place with its tokens (except the final place) is first isolated from the given Petri Net model and converted into a subnet as shown in Fig. 5c.

The purpose of the subnet is correlating the existing tokens in a given Petri Net from t_{k-1} to the new tokens extracted in t_k by using the tracking information. For each place P_i, the subnet creates two additional (pseudo) places P_i'' and P_i' (shown with black circles). Correlating the tokens from t_{k-1} to the extracted ones in t_k is graphically represented by the combination of a black arc, a transition, and a red arc. The red arcs are defined only for the tokens from t_k and the black arcs are defined only for the tokens from t_{k-1}. Each token is assumed to perform one of the following events merge, split, continue or disappear during the transition from t_{k-1} to t_k. Therefore, these events can be used to model the token dynamics with Petri Nets. The transitions *merge, splits, disappears,* and *continues* are obtained from the tracking information. For the *merge* transition, the subnet removes b merging tokens from t_{k-1} and puts the merged token from t_k into P_i''. Similarly, the *splits* transition moves the splitting token from t_{k-1} in P_i and puts a number of corresponding tokens from t_k into P_i''. The values of the variables a and b (along with the token IDs) are obtained from the tracking information. The *continues* transition removes a token from t_{k-1} in P_i and puts the corresponding token from t_k into P_i''. The *disappears* transition removes disappearing tokens from P_i to P_i'.

Once the subnet reaches its final marking, the tokens remaining in P_i'' are the ones that changed their state during the transition from t_{k-1} to t_k, and the tokens in P_i' are the disappearing ones during the transition from t_{k-1} to t_k. Depending on the domain, the tokens in both places P_i'' and P_i' can be moved back into place P_i, can be discarded or can be moved back into one of the initial places. Fig. 5d illustrates the update process for the Petri Net shown in Fig. 5a. In Fig. 5d, first each place (except the final place) is converted into a subnet, and each of these subnets is executed independently. The update process replaces the single token in Fig. 5a with three tokens. Once the update process is completed, the isolated places with their updated tokens are put back into the given model. Therefore, while the direct transition from Figs. 5a to 5b is not defined in typical Petri Nets, this transition becomes possible through the TTPN (by defining and using subnets).

After running the update process, the new (updated) tokens can be used to execute the given model to obtain the final marking in t_k. This is done by using (3).

3.1.1 Modeling with TTPN

The scientist can model an activity as a combination of feature states and actions. Table 2 provides examples to illustrate what a token, place, and transition may represent in a Petri Net model. The activity model should be drawn by considering only one instance of an activity. (Many different Petri Nets could be drawn representing the same activity). That instance (the model) should start from an initial place where the activity starts and should end at a final place where the activity completes.

One important aspect of modeling an activity is that the scientist should consider the flow (movement) of tokens from one place to the next, when drawing the model. Since the purpose is detecting multiple events, essentially a token should represent the progress of an instance of the activity in a Petri Net. For example, assume that a place represents "two-people handshaking" in computer vision. In that case, a token represents a group of two people who are handshaking. Similarly, while a token can represent a feature in one place, in the next place a token can represent a certain group of features in scientific visualization. This is especially useful to simplify the modeling of *formation* type of activities where a feature eventually transforms into a superstructure (a group) [46].

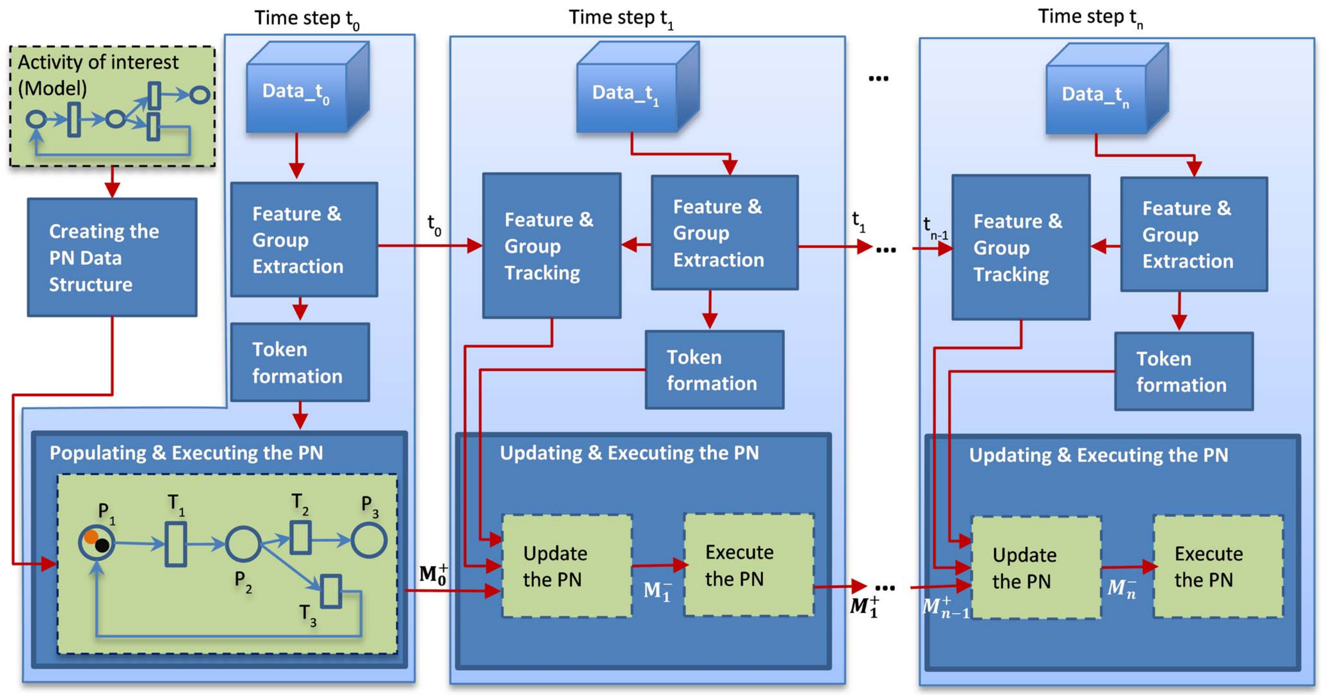
Since the merge, split, disappear cases are implicitly handled by TTPN (via the subnet shown in Fig. 5c), the scientist do not need to consider these cases in his/her model explicitly. This would greatly simplify the modeling process for the scientist.

3.2 Activity Detection Framework with TTPN

Fig. 1 illustrates the overview of our proposed framework and Fig. 6 shows the process in each time step. In Fig. 6, the input to the system is the data set and the Petri Net model defined by the scientist (see next section). In t_0, Petri Net data structure is created based on the given model. Simultaneously, the data at t_0 are processed. During the data process, features, groups, variable changes or other types of user interested entities are computed. Different types of features can be extracted by using appropriate tools (as in [1] or [35]) for a specific domain. The computed metadata may include volume, mass, centroid, max and min locations, max and min positions, orientation, shape information, and so on. The metadata (or a group) forms the tokens. Once all the tokens are formed, they are used to execute the Petri Net starting from the initial place. Both the metadata and the final marking $\mathbf{M}_0^+$ are passed into the next time step t_1.

In time step t_1, first the data at t_1 are processed to extract features and groups. Then, their metadata is computed. This metadata is transformed into tokens. Next step is correlating the extracted features and groups to the extracted ones in t_0. Any of the available tracking algorithms (such as volume overlap, prediction or time varying contour-based algorithms) can be used to correlate features and groups or it may be inherent in the simulation. Tracking step computes various attributes including the tracking history of the features (correspondence list), position changes, and any other value/attribute that is a function of two consecutive time steps. Both the newly formed (extracted) tokens and computed tracking information are fed into the Petri Net for activity detection. In Petri Net, the first step is correlating the existing tokens in the Petri Net (from t_0) to the tokens extracted in t_1. Once the Petri Net is updated by using the tracking information (as described in Section 3.1), the marking $\mathbf{M}_1^-$ is obtained. At this step, the Petri Net is executed to obtain the final marking $\mathbf{M}_1^+$.

Both the computed metadata at t_1 and $\mathbf{M}_1^+$ are fed into the next step. This process repeats itself recursively for each time step. The metadata that comes from the previous time step is used for tracking and the final marking that comes



Time step t_0
Time step t_1
Time step t_n
Activity of interest (Model)
Creating the PN Data Structure
Data_t_0
Data_t_1
Data_t_n
Feature & Group Extraction
Feature & Group Tracking
Token formation
Populating & Executing the PN
Updating & Executing the PN
Update the PN
Execute the PN
P_1 T_1 T_2 P_3 P_2 T_3
t_0 t_1 t_{n-1}
M_0^+ M_1^- M_1^+ M_{n-1}^+ M_n^-

Fig. 6. Flow diagram of the proposed framework. Activity model is used to create the Petri Net data structure. Tokens are formed based on the extracted feature (or group) attributes. Once the final marking is obtained, the feature (or group) attributes and the final marking (M_0^+) are passed into the next time step. In the next time step, first the features are extracted; their attributes are computed and tokens of the current time step are formed. Tracking information is combined with M_0^+ to correlate these tokens to the ones in the Petri Net in the update process. This yields the initial marking M_1^-. Once the Petri Net is updated, the execution process yields the final marking M_1^+. This process recursively continues in each new time step. The tokens that fall into the final place are the ones that complete the activity.

from the previous step is used to update the Petri Net in each new time step. Tokens that fall into the final place are the ones performing the complete activity.

When the evaluation over time is completed, the list of tokens with their token histories in the final place can be used to generate an activity list. This list, then, can be used for visualization and further data analysis purposes.

3.3 Implementation

Our implementation is in C++ and uses a set of classes and linked lists. The Petri Net data structure is formed according to the model given by the scientist. Currently, the scientist provides the model along with all the place and transition conditions in a text-based *config* file. We are currently developing a better interface that will help scientist model an activity graphically. In a given Petri Net model, the tokens are the only variables/classes that change over time. Each token also has a *token-history*. A token history is a list that adds the triple tuple (t_j, P_i, ObjID) to the token history at each iteration, where the t_j is the jth time step and ObjID is the object (token) ID. In the merge case, all the merging object IDs form a triple tuple in a token-history.

In our TTPN implementation, we use logical or mathematical expressions formed of object attributes to describe a feature's state or action. A place condition is run at each time step to determine whether a token still remains in that place. Tokens that change their states are put into a vector for a further evaluation to check if they changed their places via the firing process. A transition condition is used to determine whether that transition can be enabled for a given token. If a token satisfies the transition condition, then a second step checks whether the same token exists in all the incoming places. Furthermore, a third step checks whether the object satisfies "at least" one of the output places' conditions. After passing the third step, the transition is enabled and ready to fire. Firing a transition for a token removes the token from all the incoming places, and puts it into the output places for which the token satisfies the conditions.

In each domain or in each application, different feature attributes can be computed and saved in different orders. Moreover, a different combination of the available feature attributes can be used as a condition for each place or for each transition. To cover such variability and flexibility in action and state definitions, in our implementation we use *Petri Net variables*. A Petri Net variable is either a specific feature attribute or a default action from a library (such as merge, split, continuation or new born) and can belong to either the current time step or the previous time step.

These variables take one of the following forms: "$t_cA_\#$," "$t_pA_\#$,""$t_cD_\#$" or "$t_pD_\#$," where the first two characters, t_c and t_p, stand for the current time step and previous time step, respectively. $A_\#$ is an integer number and represents the index (column) number of an attribute from a list of attributes for a given feature and $D_\#$ represents the index number of the predefined action from a library (such as Table 1). For example, "t_cD_4" means the fourth action from the library (which is the split action in our implementation) in the current time, and "t_cA_3" means the third attribute value of a feature in the current time step. Let us consider the transition: "*Volume increase is more than percent 40 of the*

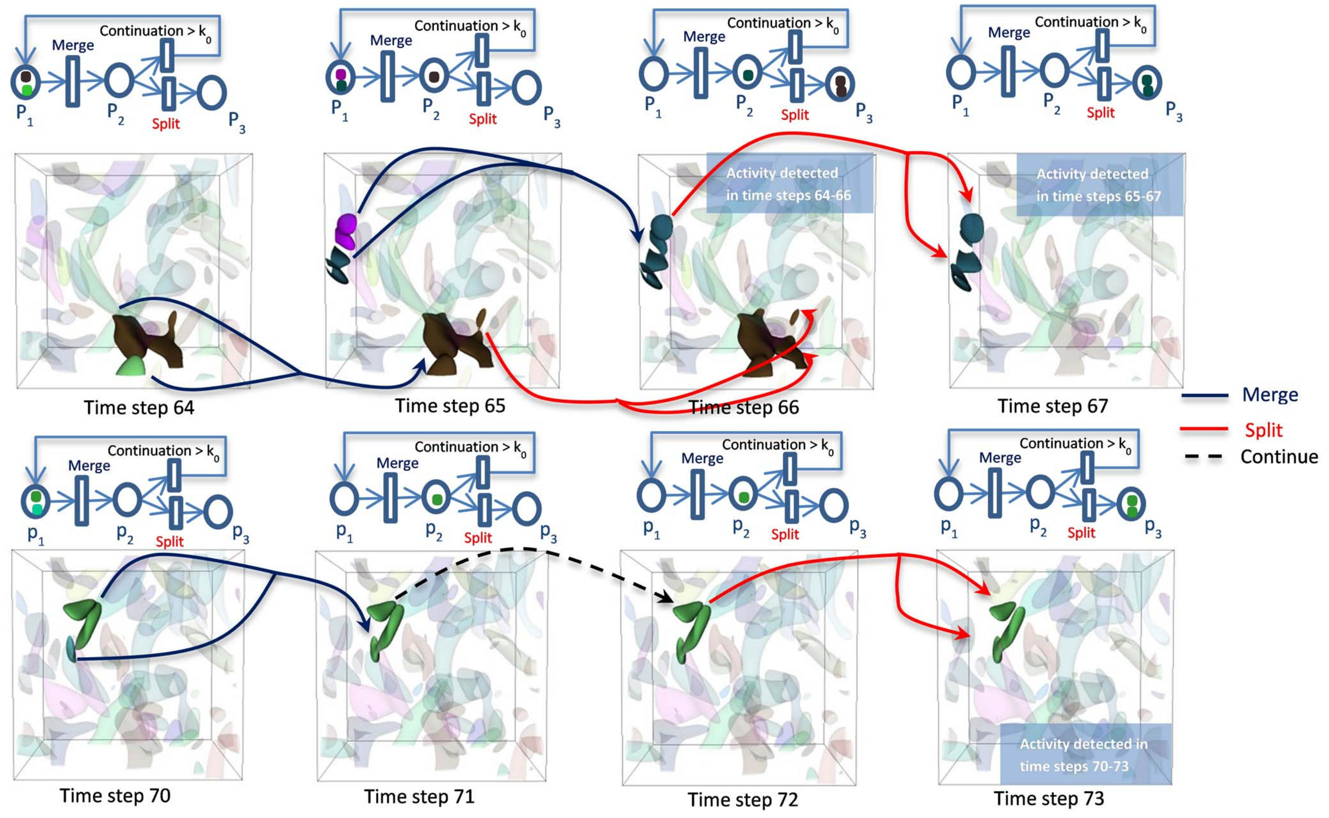


Fig. 7. Detecting activities in scientific data sets: This visualization shows the “Merge-Split” activity. The activity is modeled as a Petri Net shown above, and is defined as a feature merging and splitting within $k_0 = 5$ time steps. The activity detection process found 15 “Merge-Split” activities over 100 time steps. Three activities out of those 15 are visualized above. The colored dots in the Petri Net show the locations of the participating features for each time step in the associated feature colors. All other vortices that do not participate in an active Merge-Split activity are shown transparent in the visualization. Two of the found activities are shown in time steps 64-67. The associated time-varying transfer function is automatically generated for the visualization. Petri Nets encapsulate the components that define an activity and help in abstracting time. For example, another “Merge-Split” activity which occurs over four time steps instead of three is shown time steps 70-73. All of the detected 15 Merge-Split activities and the participating tokens are visualized in a video available at [52].

previous volume value.” This can be expressed as a difference of the volume values of the current and previous time steps. Assuming the third value in the attributes file represents the volume, we can construct the related transition condition as “$(t_cA_3 - t_pA_3) > (0.4 * t_pA_3)$”. This Boolean expression decides whether the condition is satisfied and serves as an *action* detector. Similarly, the place conditions can define the feature states. Our token-tracking Petri Net implementation allows the use of built in functions for constructing similar condition expressions.

Activity visualization uses the token history. In our implementation, the token history is captured in an output file summarizing all the detected activities. The output file includes the participating objects’ IDs and their corresponding place IDs in each time step for each detected activity.

3.4 Visualization

Effective time-varying visualization involves adapting and changing the visualization parameters such as thresholds, intervals or min/max values automatically over time. This also includes adopting the transfer function from one time step to the next or changing which feature is the primary focus in response to the changes in the simulation. Accordingly, the location of the camera (the view point) and the lighting positions can be altered automatically. All these processes require domain knowledge. Petri Nets can be used to identify time steps for these changes to take place.

Activity detection adds functionality and flexibility to time-varying visualization and allows event and state-based data abstraction. For example, it can identify time steps where the activity takes place. Moreover, activity detection allows different visualizations highlighting that activity. Different places of a Petri Net can be used to enhance visualization. For instance, the features (tokens) at the intermediate states, i.e., places, can be visualized separately at each time step. If the scientist is interested in seeing what features from time step 17 are at place 3 (P_3), those features can be highlighted in an isosurface or volume rendering. Conversely, a scientist can ask at which time steps features move into P_3. Some of such enhanced visualization techniques in our framework can be listed as follows: *Isolated activity visualization* is the visualization of the time steps of a single activity among all the detected activities. Only the features that are currently participating in one user specified activity are visualized. For example, time steps 70 to 73 in Fig. 7 visualize one user specified activity out of the 15 “Merge-Split” activities detected over 100 time steps. *Forecast activity visualization* is the visualization of all the features that “will” complete the specified activity starting from the first time step. It visualizes the features and their

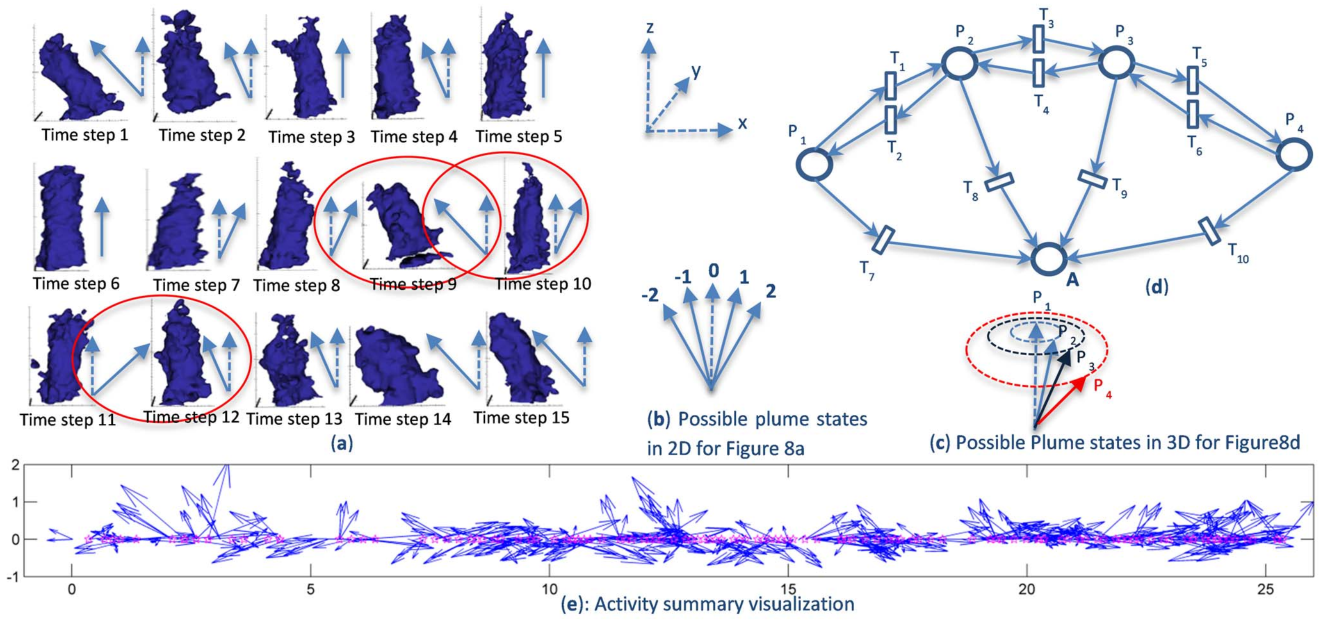


Fig. 8. “Anomalous Plume bending” detection in a time-varying 3D plume data set is visualized. (a) Sample visualization of time-varying acoustic imaging data shows one of the three plumes and the anomalous bending activities in the first 15 time steps of the data. The anomalies that do not fit the defined periodic movement between the plume bending states are circled in red. (b) Possible plume bending states are shown with blue arrows, where -2 and 2 represent maximum bending and -1 and 1 represents slight bending in *z*-*x* plane; these states are used to illustrate the anomalies shown in Fig. 8a. (c) Possible plume states are redefined in 3D for the PN shown in Fig. 8d. Each of these states is represented by a place in the Petri Net (with its respective place ID) (d) The PN model for the “Anomalous Plume Bending.” In the PN model, $T_1 = T_2 = T_3 = T_4 = T_5 = T_6$ represents the condition that the angular change in *x*-*y* plane is less than 45 degrees and the plume either stays in its current state or changes to the previous or next state. $T_7 = T_8 = T_9 = T_{10}$ represents the condition that the angular change in *x*-*y* plane is more than 45 degrees or the plume changes its state to one of the remaining states. (e) The plume bending data over the 26 days (formed of 479 time steps). The arrows represent the time steps and each shows the direction in *x*-*y* plane and relative magnitude (size) of the plume. The pink stars show the anomaly time steps.

evolutions over time performing the modeled activity. In Fig. 9a, time step 8 highlights all the features that are currently performing the packet formation event. This is an example of visualizing features that will form a group in the future. The single feature Feature_A in time step 8 evolves and eventually forms a group (Packet_A) in the future (next) time steps. *Activity summary visualization* is the visualization of all the detected activities along with the entire data set (or a portion of the data set, if the data set is excessively huge) in a single visualization. Fig. 8e is one such visualization of the entire data set. It shows how frequent the activities are and where/when they occur. *Graph-based activity visualization* is the visualization of the features in token form. In this visualization, tokens show the progress of the activity on a given Petri Net graph. Petri Net visualization of features in Fig. 7 illustrates the graph-based activity visualization. In addition to token-based visualization, a bar chart can be attached to each place. Each bar on a chart can visualize the total number of tokens in that time.

Activity detection can also help in transfer function design. A time-varying transfer function can be generated by using the activity detection results automatically, i.e., by using the list of the tokens and their activity histories in the final places. Examples of various visualizations are shown in Figs. 7, 8, and 9.

4 Case Studies and Results

In this section, we demonstrate the use of activity detection in three different case studies.

4.1 Test Case: Merge-Split Activity in Turbulent Vortex Structures

In our first application, we will look for “*Merge-Split activity*.” The Merge-Split activity is where a single feature first merges with another vortex and then splits again within k_0 time frames.

Similar activities were described in [34] and [47] previously. Here, we show how our Petri Net approach can simplify the process and successfully search for the instances of the Merge-Split activity in a given data set. The Petri Net model of this activity is shown in Fig. 7. For testing, k_0 was set to 5.

The data are a small data set from [1], which is a pseudospectral simulation of coherent turbulent vortex structures. The simulation data resolution is 128^3 and the data contain 100 time steps. The features (A set of connected nodes where the data value is above a certain threshold) are extracted, tracked, and their attributes (metadata) are computed [1]. Running the Petri Net on this metadata found 15 activities in 100 time steps. Three of them are visualized in Fig. 7. First one of these three activities takes place between time steps 64-66, the second one is detected between time steps 65-68, and the third one is detected between time steps 70-73. In the figure, each feature has a distinct color, except that splitting features have the same color. The validation process is done by visually inspecting all the detected activities. During the visual inspection, we observed that all the detected activities were indeed merging and then splitting within five time steps.

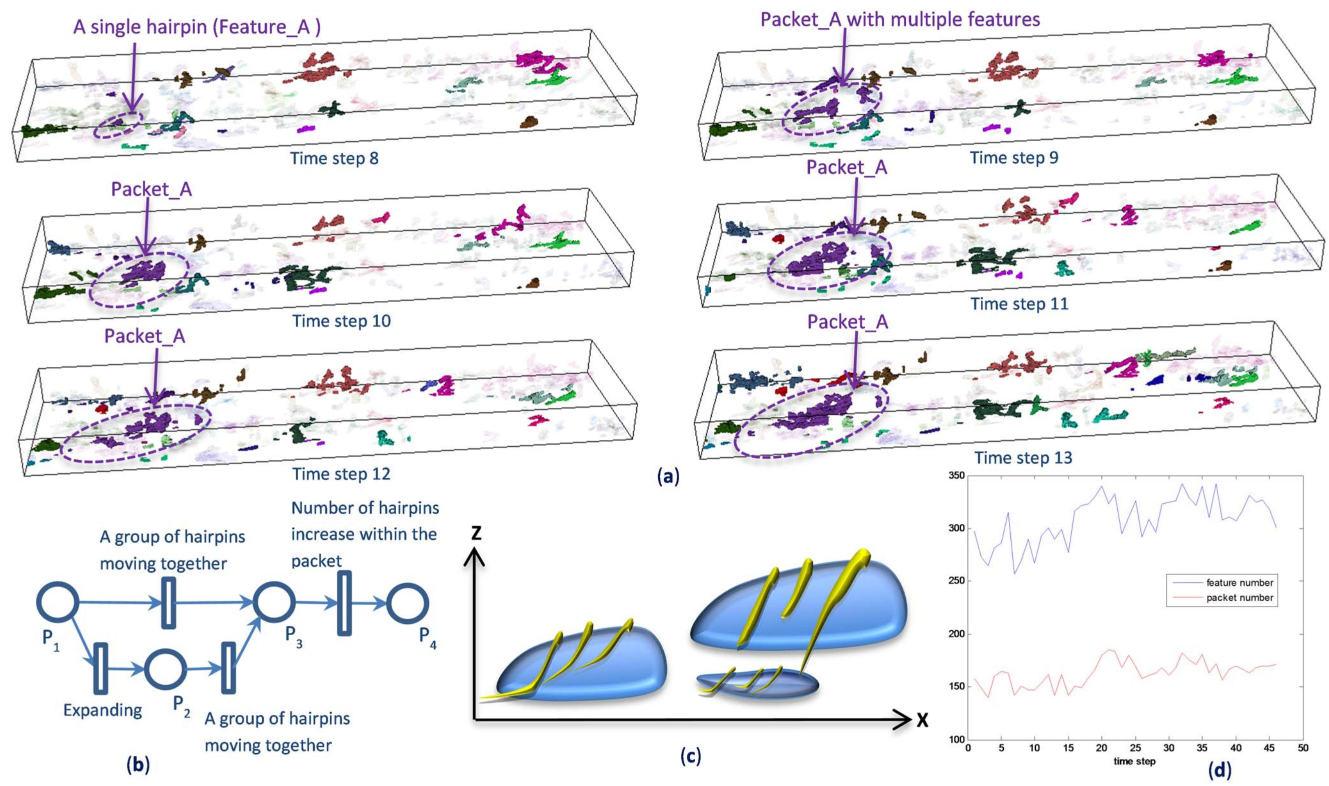

A single hairpin (Feature_A)
Time step 8
Packet_A with multiple features
Time step 9
Packet_A
Time step 10
Packet_A
Time step 11
Packet_A
Time step 12
Packet_A
Time step 13
(a)
A group of hairpins moving together
Number of hairpins increase within the packet
P1
P2
P3
P4
Expanding
A group of hairpins moving together
(b)
Z
X
(c)
feature number
packet number
time step
(d)

Fig. 9. Wall-bounded turbulence DNS, (a) sample visualization of packet formations in time steps 8-13, an example formation of a packet (Packet_A) is circled in purple, (b) a PN model for the packet formation event, where P_1 and P_2 represent single hairpin vortex, P_3 and P_4 represent packets including multiple hairpin vortices, (c) an illustration of a packet (a group of hairpin vortices) and a super structure formed of packets, (d) the total number of extracted packets and features (hairpin vortices) in each time step.

4.2 Tidal Forcing in Plumes

The COVIS project (a sonar platform connected to the NEPTUNE Canada undersea cabled observatory) acoustically images 3D hydrothermal plumes four times a day producing an ever-increasing number of time steps [48]. Seafloor hydrothermal plumes exist in a turbulent environment with a variety of currents, and as they rise the plumes bend in response to these currents [36].

Two different data sets capture the behavior of the plumes from Grotto Vent in the Main Endeavour Field on the Juan de Fuca Ridge. A 24-hr time series (with hourly sampling) data set was collected in 2000 using the ROV Jason. Fig. 8a shows one of three existing plumes in the 15 time steps of the data with 51^3 resolution. Each plume image is scaled at a different scale to fit in the figure and to focus on the bending rather than the texture or the length. A three week time series (with sampling every 3 hours) was collected using the NEPTUNE Canada cabled seafloor observatory (479 time steps available) in token 2010.

When the Endeavour node of the observatory went back on line in the autumn of 2011, it started producing 16 files (24 GB) of acoustic imaging data a day (5,696 files per year). As the data set becomes too large to process and view manually, activities of interest need to be modeled and searched for automatically. One such event, “anomalous-bending,” is an inconsistent plume bending that does not behave according to the expected cyclic movement.

For the first data set, we have used the metadata from [36] and have applied a Petri Net analysis to determine if the bending patterns observed in the plumes are consistent with a semidiurnal tidal cycle (the data sets have multiple plumes, in this example, we focus on only one of them).

In analogy with the in- and out-coming of coastal tides, the changes in direction of tidal flow were anticipated to imply a stagnant period in between directions during which the plume would be vertical. Furthermore, the basic pattern was assumed to be a simple back and forth between narrowly defined states. The initial Petri Net assumed motion only in a vertical (x-z) plane and defined five states that include the vertical stagnant case and two degrees of bending in each of two directions. These states are shown in Fig. 8b. We then use this Petri Net to model the assumed normal process of gentle progressive changes and detect the anomalies (abrupt changes) along the x-z plane.

We observe three instances of anomalous-bending at times steps 9, 10, and 12 (circled in red in Fig. 8a) within the first plume data set. Our Petri Net algorithm provides results that match the visually picked results. As a test case, this confirmed that the Petri Net algorithm could detect abrupt changes in bending direction.

However, when applied to a larger data set, the results of the initial Petri Net model let the scientist to see that the initial model did not consider the 3D nature of ocean currents and plume responses; the classification of all bending directions as N (+) or S (−) resulted in treating some very small changes in direction as anomalies because the change spanned the definition boundaries. Therefore, the states and conditions were redefined to allow for a greater variability in direction while still restricting the magnitude of changes over a single time step. Consider the angles in 3D between z-axis and x-y plane (Fig. 8c). This model specifies several states of bending between vertical and highly bent and expects that the change over a single time step neither exceed shifting one bending state or changing direction by more than 45 degrees. Any larger shifts in bending or direction are considered an anomaly. The new Petri Net model is shown in Fig. 8d and it has detected 131 time steps that showed anomalies out of the 479 available time steps. The available 479 time steps are plotted along the x-axis (where the unit is in days) in Fig. 8e. For each day, the magnitude of Plume A bending and its major direction is represented using an arrow. The anomaly events are highlighted with a pink star.

As the visualization suggests, a large ($\sim$30%) fraction of the time steps have been identified as anomalous. The scientist validated these Petri Net results. The high percentage value indicated that the plume bending varies more and changes more erratically than anticipated. Future refinements of the model include broadening the allowed direction change or reducing the number of bending states. Alternative models looking for time periods of no change or high change are also studied. These results have been further presented and discussed in [48] and [49].

4.3 Packet Formation in Wall-Bounded Turbulence Flows

In DNS of wall-bounded turbulence flow, scientists have been interested in searching for the existence of groups of coherent but unconnected features, their formation, dynamic evolution, and number of these groups [7], [46]. An illustration of such a group is shown in Fig. 9c [50]. The yellow hairpin vortices (features) move coherently inducing a secondary (blue) fluid mass of low momentum. These coherent structures are called packets. (This is analogous to groups of humans walking coherently in a crowd or to a school of fish). Notice that, in general, the hairpins are not connected within a packet. Each packet includes varying number of hairpins where these hairpins should be aligned at a downstream-leaning angle (γ) and the distance between the hairpins should not exceed a predefined physically meaningful value. Some of these packets lead to the formation of younger packets over time. Moreover, among all these packets, some act coherently forming super structures (as illustrated in the grouping of Fig. 9c) inducing meandering regions of low momentum. The activity we are interested in is the "packet formation" event which is characterized by a single hairpin evolving into a packet formed of multiple features over time [7].

The data sets are currently at a resolution of $2{,}520 \times 1{,}120 \times 110$. In this work, we took a subset of the data with the resolution of $384 \times 256 \times 69$ to test the hypothesis. This simulation data (shown in Fig. 9a) has 46 time steps. Fig. 9b shows the Petri Net model for packet formation. In Fig. 9b, P_1 and P_2 represent a packet formed of a single feature; P_3 and P_4 represent a packet formed of multiple features in Fig. 9b. The activity (packet formation) starts at P_1 (initial place) and ends at P_4 (final place). Notice that the transition "A group of hairpins moving together" can be replaced with another Petri Net to detect and identify groups.

Group dynamics needs to be computed as a part of the tracking algorithm. This is done by group tracking [44]. Feature extraction is performed at the threshold $0.1 * 10^{-3}$ via a region growing algorithm and the objects with the volume lower than 25 are filtered. The average number of extracted features is 308 and the average number of found groups is 163 in 46 time frames. Fig. 9d demonstrates the total number of found features and packets in each time step. The PN model yielded 288 packet formation activities over 46 time steps. Fig. 9a visualizes the portion of the activities that take places in the time steps between 8 and 13. It is apparent that the single Feature_A (circled in purple) transforms into the Packet_A in the following time steps. All other packets that are not currently performing the modeled activity are transparent. Preliminary results of this study have been presented in [51].

5 Discussion and Conclusion

Defining and refining the Petri Net model was an interactive process for both scientists in case studies 4.2 and 4.3. The future work will involve creating a better interface that allows the scientist to define activity models graphically. We are also extending the framework to operate in-situ, where the data are produced. By considering questions such as when an event occurs or what events are occurring at a particular time, we can visualize the data in a more compact way.

In this work, we introduce the concept of activity detection for scientific visualization and show how a scientist can utilize activity detection and automate the process of searching for important events in time-varying simulations. Moreover, we demonstrate how activity detection with Petri Nets can be used to validate a hypothesis in scientific simulations.

A scientist can formulate an idea about how features interact and then search for that activity among thousands of time steps. Moreover, we enhance Petri Nets to consider feature dynamics in scientific simulations and to detect simultaneous activities in time-varying data sets. Petri Nets are model-based techniques and do not require training data to model the activity. Instead, they use the domain scientist's knowledge and let the scientist represent the activity in a semantically meaningful way. The different case studies demonstrate that while the domains and actors are different, the concept of activity detection can be applied to all. Our framework went over all the time steps and pulled out the relevant features and time steps effectively into more manageable chunks. Multiple subnets can be defined to model various domain dynamics in TTPN.

In this paper, all the applications were based on physically observable coherent features. However, notice that Petri Nets are not limited to the detection of the activities of only coherent features. The proposed framework can also be applied to detect the activities of specific nodes (or quantities) in both Lagrangian and Eulerian simulations. For example, activities such as "the minimum pressure remains constant for five time steps" can still be modeled and detected by Petri Nets. In this case, the segmentation step of the framework would become trivial because either each node, a quantity or the entire domain would become a token in TTPN.

One of the main challenges in scientific data analysis is creating the training set or ground truth for the use of available machine learning or data mining techniques in scientific simulations. Petri Nets can also create the necessary training data set from the semantic descriptions for further analysis with other data mining techniques. A semantic-based approach (such as Petri Nets) allows exploratory knowledge discovery besides detecting certain events in time-varying data sets.

Data rates are increasing exponentially. Scientists are computing ever larger simulations and collecting increasing amounts of complex sensor data. Automated activity detection techniques are necessary to filter the data and provide a useful and meaningful way for scientists interact with the data.

Acknowledgments

The authors gratefully acknowledge the support of SciDAC Institute for Ultra-Scale Visualization, http://vis.cs.ucdavis.edu/Ultravis/, DOE #DE-FG02-09ER25977 and US National Science Foundation (NSF) grant 0825088. They would also like to thank the anonymous reviewers for their constructive comments.

References

[1] D. Silver and X. Wang, "Tracking and Visualizing Turbulent 3d Features," *IEEE Trans. Visualization and Computer Graphics,* vol. 3, no. 2, pp. 129-141, Apr.-June 1997.

[2] M. Albanese, R. Chellappa, V. Moscato, A. Picariello, V.S. Subrahmanian, P. Turaga, and O. Udrea, "A Constrained Probabilistic Petri Net Framework for Human Activity Detection in Video," *IEEE Trans. Multimedia,* vol. 10, no. 8, pp. 1429-1443, Dec. 2008.

[3] J.K. Aggarwal and M.S. Ryoo, "Human Activity Analysis: A Review," *ACM Computing Surveys,* vol. 43, article 16, 2010.

[4] R.P. Botchen, S. Bachthaler, F. Schick, M. Chen, G. Mori, D. Weiskopf, and T. Ertl, "Action-Based Multifield Video Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 4, pp. 885-899, July 2008.

[5] C. Castel, L. Chaudron, and C. Tessier, "What Is Going on? A High Level Interpretation of Sequences of Images," *Proc. Workshop Conceptual Descriptions Images (ECCV),* pp. 13-27, 1996.

[6] M.L. Parry, P. Legg, D.H.S. Chung, I.W. Griffiths, and M. Chen, "Hierarchical Event Selection for Video Storyboards with a Case Study on Snooker Video Visualization," *Proc. IEEE Visualization '11,* 2011.

[7] M.J. Ringuette, M. Wu, and M.P. Martin, "Coherent Structures in Direct Numerical Simulation of Turbulent Boundary Layers at Mach 3," *J. Fluid Mechanics,* vol. 594, pp. 59-69, 2008.

[8] S. Bhattacharya, S. Habib, and K. Heitmann, "Dark Matter Halo Profiles of Massive Clusters: Theory vs. Observations," *Astrophysical J.,* vol. 766, 2011.

[9] J. Wei, H. Yu, R.W. Grout, J.H. Chen, and K.-L. Ma, "Visual Analysis of Particle Behaviors to Understand Combustion Simulations," *IEEE Computer Graphics and Applications,* vol. 32, no. 1, pp. 22-33, Jan. 2012.

[10] J.A. Insley, L. Grinberg, and M.E. Papka, "Visualizing Multiscale, Multiphysics Simulation Data: Brain Blood Flow," *Proc. IEEE Symp. Large Data Analysis and Visualization (LDAV),* 2011.

[11] J.G. Luhmann, S.C. Solomon, J.A. Linker, J.G. Lyon, Z. Mikic, D. Odstrcil, W. Wang, and M. Wiltberger, "Coupled Model Simulation of a Sun-to-Earth Space Weather Event," *J. Atmospheric and Solar-Terrestrial Physics,* vol. 66, pp. 1243-1256, 2004.

[12] K.-L. Ma, E.B. Lum, H. Yu, H. Akiba, M.-Y. Huang, Y. Wang, and G. Schussman, "Scientific Discovery through Advanced Visualization," *J. Physics: Conf. Series,* vol. 16, no. 1, p. 491, 2005.

[13] E.M. Tapia, S.S. Intille, and K. Larson, "Activity Recognition in the Home Using Simple and Ubiquitous Sensors," *Proc. Second Int'l Conf. Pervasive Computing,* pp. 158-175, 2004.

[14] A.I. Hernandez, G. Carrault, F. Mora, L. Thoraval, G. Passariello, and J.M. Schleich, "Multisensor Fusion for Atrial and Ventricular Activity Detection in Coronary Care Monitoring," *IEEE Trans. Biomedical Eng.,* vol. 46, no. 10, pp. 1186-1190, Oct. 1999.

[15] P. Turaga, R. Chellappa, V.S. Subrahmanian, and O. Udrea, "Machine Recognition of Human Activities: A Survey," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 18, no. 11, pp. 1473-1488, Nov. 2008.

[16] M. Goebel and L. Gruenwald, "A Survey of Data Mining and Knowledge Discovery Software Tools," *ACM SIGKDD Exploration Newsletter,* vol. 1, no. 1, pp. 20-33, 1999.

[17] P. Rashidi, D.J. Cook, L.B. Holder, and M. Schmitter-Edgecombe, "Discovering Activities to Recognize and Track in a Smart Environment," *IEEE Trans. Knowledge & Data Eng.,* vol. 23, no. 4, pp. 527-539, Apr. 2011.

[18] H. Storf, M. Becker, and M. Riedl, "Rule-Based Activity Recognition Framework: Challenges, Technique and Learning," *Proc. Third Int'l Conf. Pervasive Computing Technologies for Haelthcare (PervasiveHealth '09),* pp. 1-7, 2009.

[19] N. Ghanem, D. Dementhon, D. Doermann, and L. Davis, "Representation and Recognition of Events in Surveillance Video Using Petri Net," *Proc. Conf. Computer Vision and Pattern Recognition Workshop (CVPR Workshop),* 2004.

[20] G. Lavee, M. Rudzsky, E. Rivlin, and A. Borzin, "Video Event Modeling and Recognition in Generalized Stochastic Petri Nets," *IEEE Trans. Circuits and Systems for Video Technology,* vol 20, no. 1, pp. 102-118, Jan. 2010.

[21] M. Perše, M. Kristan, J. Perš, G. Mušič, G. Vučkovič, and S. Kovačič, "Analysis of Multi-Agent Activity Using Petri Nets," *Pattern Recognition,* vol. 43, no. 4, pp.1491-1501, Apr. 2010.

[22] J.N.K. Liu, K. Wang, Y.-L. He, and X.-Z. Wang, "Formal Representation and Verification of Ontology Using State Controlled Coloured Petri Nets," *Reliable Knowledge Discovery,* pp 269-290, Springer, 2012.

[23] J.L. Peterson, *Petri Net Theory and the Modeling of Systems.* Prentice Hall, 1981.

[24] R. David and H. Alla, *Petri Nets & Grafcet.* Prentice Hall, 1992.

[25] M. Chen, D. Ebert, H. Hagen, R.S. Laramee, R. Van Liere, K.-L. Ma, W. Ribarsky, G. Scheuermann, and D. Silver, "Data, Information, and Knowledge in Visualization," *IEEE Computer Graphics and Applications,* vol. 29, no. 1, pp. 12-19, Jan./Feb. 2009.

[26] N. Oliver, B. Rosario, and A. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions," *Proc. Int'l Conf. Vision Systems (ICVS '99),* Jan. 1999.

[27] Y.A. Ivanov and A.F. Bobick, "Recognition of Visual Activities and Interactions by Stochastic Parsing," *IEEE Trans. Pattern Analysis & Machine Intelligence,* vol. 22, no. 8, pp. 852-872, Aug. 2000.

[28] J. Wei, Z. Shen, N. Sundaresan, and K.-L. Ma, "Visual Cluster Exploration of Web Clickstream Data," *Proc. IEEE Conf. Visual Analytics Science and Technology,* 2012.

[29] G. Lavee, E. Rivlin, and M. Rudzsky, "Understanding Video Events: A Survey of Methods for Automatic Interpretation of Semantic Occurrences in Video," *IEEE Trans. Systems, Man, and Cybernetics, Part C,* vol. 39, no. 5, pp. 489-504, Sept. 2009.

[30] R. Poppe, "A Survey on Vision-Based Human Action Recognition," *Image and Vision Computing,* vol. 28, pp. 976-990, 2010.

[31] R. Borgo, M. Chen, B. Daubney, E. Grundy, H. Janicke, G. Heidemann, B. Hoferlin, M. Hoferlin, D. Weiskopf, and X. Xie, "A Survey on Videobased Graphics and Video Visualization," *Proc. Eurographics (State of the Art Reports),* pp. 1-23, 2011.

[32] J. Woodring and H.W. Shen, "Multiscale Time Activity Data Exploration via Temporal Clustering Visualization Spreadsheet," *IEEE Trans. Visualization & Computer Graphics,* vol. 15, no. 1, pp. 123-137, Jan./Feb. 2009.

[33] W. Dou, X. Wang, D. Skau, W. Ribarsky, and M.X. Zhou, "LeadLine: Interactive Visual Analysis of Text Data through Event Identification and Exploration," *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST),* pp. 93-102, 2012.

[34] A. Gezahegne and C. Kamath, "Tracking Non-Rigid Structures in Computer Simulations," *Proc. IEEE 15th Int'l Conf. Image Processing (ICIP),* 2008.

[35] *Topological Methods in Data Analysis and Visualization,* V. Pascucci, X. Tricoche, H. Hagen, J. Tierny, eds. Springer, 2011.

[36] P.A. Rona, K.G. Bemis, D. Kenchammana-Hosekote, and D. Silver, "Acoustic Imaging and Visualization of Plumes Discharging from Black Smoker Vents on the Deep Seafloor," *Proc. IEEE Visualization '98,* pp. 475-478, 1998.

[37] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities," *IEEE Trans. Visualization & Computer Graphics,* vol. 12, no. 5, pp. 1053-1060, Sept. 2006.

[38] G. Ji and H.-W. Shen, "Feature Tracking Using Earth Mover's Distance and Global Optimization," *Proc. Pacific Graphics '06,* 2006.

[39] F. Reinders, F.H. Post, and H.J.W. Spoelder, "Visualization of Time-Dependent Data Using Feature Tracking and Event Detection," *The Visual Computer,* vol. 17, no. 1, pp. 55-71, 2001.

[40] G. Ji, H-W. Shen, and R. Wenger, "Volume Tracking Using Higher Dimensional Isosurfacing," *Proc. IEEE Visualization '03,* pp. 209-216, 2003.

[41] F.-Y. Tzeng and K.-L. Ma, "Intelligent Feature Extraction and Tracking for Large-Scale 4D Flow Simulations," *Proc. ACM/IEEE Conf. Supercomputing,* 2005.

[42] J. Caban, A. Joshi, and P. Rheingans, "Texture-Based Feature Tracking for Effective Time-Varying Data Visualization," *IEEE Trans. Visualization & Computer Graphics,* vol. 13, no. 6, 1472-1479, Nov. 2007.

[43] C. Muelder and K.-L. Ma, "Interactive Feature Extraction and Tracking by Utilizing Region Coherency," *Proc. IEEE Pacific Visualization Symp.,* Apr. 2009.

[44] S. Ozer, J. Wei, D. Silver, K.-L. Ma, and P. Martin, "Group Dynamics in Scientific Visualization," *Proc. IEEE Symp. Large Data Analysis and Visualization (LDAV),* 2012.

[45] S. Ozer, "Activity Detection in Scientific Visualization," PhD dissertation, Rutgers Univ., NJ, 2013.

[46] C. O'Farrrell and M.P. Martin, "Chasing Eddies and Their Wall Signature in DNS Data of Turbulent Boundary Layers," *J. Turbulence,* vol. 10, pp. 1-22, 2009.

[47] K.P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell, "Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames," *IEEE Trans. Visualization & Computer Graphics,* vol. 16, no. 2, pp. 248-260, Mar. 2010.

[48] K.G. Bemis, G. Xu, J. Rabinowitz, P.A. Rona, D.R. Jackson, and C.D. Jones, "Understanding Plume Bending at Grotto Vent on the Endeavour Segment," *Proc. AGU Fall Meeting,* 2011.

[49] K.G. Bemis, S. Ozer, G. Xu, P.A. Rona, and D. Silver, "Event Detection for Hydrothermal Plumes: A Case Study at Grotto Vent," *Proc. AGU Fall Meeting,* Dec. 2012.

[50] R. Adrian, C. Meinhart, and C. Tomkins, "Vortex Organization in the Outer Region of the Turbulent Boundary Layer," *J. Fluid Mechanics,* vol. 422, pp. 1-54, 2000.

[51] Y.C. Kan, C. Helm, and M.P. Martin, "Turbulence Structure and Wall Signature in Hypersonic Turbulent Boundary Layer," *Proc. 15st AIAA Aerospace Sciences Meeting,* 2013.

[52] Vizlab, Rutgers Univ., NJ http://coewww.rutgers.edu/www2/vizlab/gallery/, 2013.

Sedat Ozer received the BSc degree in electronics engineering from Istanbul University, the MSc degree in electrical engineering from the University of Massachusetts, Dartmouth, and the PhD degree in electrical and computer engineering from Rutgers University. His research focuses on pattern recognition, machine learning including kernel machines and statistical learning, signal/data/image processing, tracking, and activity detection-related problems in both computer vision applications and scientific simulations.

Deborah Silver received the BS degree from Columbia University School of Engineering and the MS and PhD degrees from Princeton University in computer science. She is a professor in the Department of Electrical and Computer Engineering at Rutgers, The State University of New Jersey and the executive director of the Professional Science Master's Program. Her area of research is in scientific visualization she has been a PI in the Vizlab at the CAIP Center, Rutgers University because joining the faculty. She has taught courses in computer graphics, visualization, data structures, software engineering and robotics. She is involved in different visualization projects including oceanic visualization, medical visualization, CFD visualization, and volume graphics. She has been a co-chair of the papers session and a program co-chair of the yearly IEEE Visualization conference, she has been the vice chair of operations for the IEEE Technical Committee on Computer Graphics (1993-2000), and she has been on the editorial committee of the *IEEE Transaction on Visualization and Computer Graphics (1995-2000)*. She was the associate dean for Continuing and Professional Education for the School of Engineering from 2008-2010.

Karen Bemis received the bachelor's degree from Rice University (in Houston, TX) in geophysics in 1988 and went on to the MIT/WHOI Joint Program in Oceanography to study heat flux. There, she learned to love hydrothermal systems and fluid dynamics, while working with Dick Von Herzen and Debbie Smith. She left the Joint Program in 1991 with the master's degree in oceanography. She received the PhD degree in geological sciences at Rutgers in 1995 on the morphology of volcanos, especially small explosive volcanoes called cinder cones. After a short internship at the NJ Geological Survey studying groundwater flow in the Passaic River System, she returned to hydrothermal systems in 1996 to work with Peter Rona as a postdoc in the Institute of Marine and Coastal Sciences at Rutgers. Through Peter, she was introduced to Deborah Silver and scientific visualization. Currently, she is a research associate in IMCS. She continues to study cinder cones in Guatemala and sonar imaging of hydrothermal systems while exploring other uses of scientific visualization in geologic problems and the psychology of understanding and using visualizations.

Pino Martin received the BEng degree from Boston University and the MS and PhD degrees from University of Minnesota. She is a professor at the Department of Aerospace Engineering, University of Maryland. Her research areas include computational fluid dynamics, numerical simulation of turbulent flows, direct numerical and large eddy simulation, numerical models for large eddy simulations and Reynolds-averaged Navier Stokes calculations, numerical methods for compressible turbulence, physics of compressible turbulence, shock waves and turbulence interaction, turbulence and finite-rate chemistry interaction, surface reactions, and fluid interaction.