ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Activist Stochastic Machine-Base Activity Models Quick Movement Finding

[I]PK Kumaresan, [II]T Geetha, [III]P.Praveen Kumar, [IV]AV Ramesh Kumar, [V]A. Muthamilselvi
[I]Prof CSE, [II,V]Asst Prof CSE, [III]Asst Prof IT, [IV]Asso Prof IT
[I,II,III,IV,V]VMKV Engg College Salem

Abstract
The capacity to display a nonstop stream of fine-grained data for the event of positive high-level activities. Analytically way every minute action making massive streams of time stamped remark data, possibly from multiple synchronized activities. The difficult of capably identifying events of high-level actions from such enclosed data streams. The previous years on displaying events using probabilistic models. A time-based probabilistic graph so that the elapsed time between observations. Temporal multiactivity graph to collection various events that need to be synchronously observed. Temporal Multiactivity Graph Index Creation based on this records construction, examines and links observations as they occur. Algorithms for attachment and bulk attachment into the tMAGIC index and show that this can be competently accomplished. The evidence difficult that attempts to catch all events of an activity within a given structure of clarifications. The identification difficult that attempts to catch the movement that best matches a structure of clarifications. The tMAGIC has time and space difficulty direct to the size of the input, and can capably recover occurrences of the watched events.

Keywords
Activity Detection, Indexing, Stochastic Automata, Time Stamped Data.

I. Introduction

THERE are numerous applications where we need to monitor whether certain (normal or abnormal) activities are occurring within a stream of transaction data. For example, an online store might want to monitor the activities occurring during a remote login session on its Web site in order to either better help the user or to identify users engaged in suspicious activities. A company providing security in an airport might want to monitor activities in a baggage claim area or in a secure part of the tarmac in order to identify suspicious activities. A bank might want to monitor activities at its automatic teller machines for similar reasons. It is well recognized that models of activities are likely to be uncertain. We can rarely predict exactly how a particular activity may be executed, especially as a large number of irrelevant activities might be intermixed together. As a consequence, though early models of activities were "certain" about what constituted an activity and used logical methods or context-free grammars, more recent activity detection is based on either graphical models, or stochastic automata in which vertices correspond to observable atomic events.

However, most existing work on stochastic activity recognition has two main limitations. First, they often do not account for the time between observations associated with an activity. For instance, Fig. 1 shows an example of an online bill payment activity. Each vertex corresponds to an observation (made by the system) of the activity. However, in almost all online systems, there are temporal constraints about how much time can elapse between one observation and another for the two to jointly constitute part of a single occurrence of an activity. For instance, if we observe that a user checks his balance and then 6 hours (or 6 days or 6 months!) elapse between that observation and "go Billpay," then should these two observations be counted as part of the same activity? The answer is "it depends" upon the application.

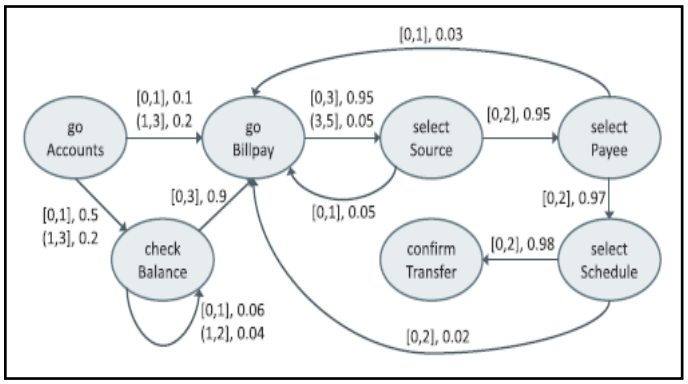


Fig. 1 : Example of temporal stochastic activity

Our contribution in this paper, The extends stores both activities and observations and enables us to quickly answer .The Evidence and Identification problems, where time and uncertainty together play a role in the definition of an activity. The propose two pruning strategies that improve the time and space performance of the tMAGIC index while guaranteeing the correctness of the results. A data structure, called a temporal multiactivity graph , to merge multiple activity graphs together and enable concurrent monitoring of multiple activities.

II. Temporal Stochastic Activity Model

Timespan Distribution: A timespan distribution ϕ is a pair (I,ד) where,

Definition 2.1 (Timespan Distribution). *A timespan distribution ω is a pair (I, τ) where:*

- *I is a set of time intervals[1] such that $\forall [x, y] \in I, x \leq y$;*
- *$\forall [x, y], [x', y'] \in I$ with $[x, y] \neq [x', y']$, the time intervals $[x, y]$ and $[x', y']$ are disjoint;*
- *$\tau : I \rightarrow [0, 1]$ is a function that associates a value $\tau(x, y) \in [0, 1]$ with each time interval $[x, y] \in I$.*

We use Ω to denote the set of all possible timespan distributions. Given a timespan distribution $\omega = (I, \tau)$, we use $S(\omega)$ to denote $\sum_{[x,y] \in I} \tau(x, y)$—we require that $S(\omega) \leq 1$.

Intuitively, a timespan distribution (I, τ) specifies a set I of disjoint time intervals when an observation might occur, and an incomplete conditional probability distribution τ. $\tau(x, y)$ is the probability that the observation will occur during the time interval $[x, y]$, given the previous observation. τ may not be complete as an observation may not occur at all after another given observation.

Temporal Stochastic Activity

Definition (Temporal Stochastic Activity). *A temporal stochastic activity (or just activity) is a labeled graph* (V, E, δ) *where*

- V *is a finite set of observations;*
- E *is a subset of* $(V \times V)$;
- $\exists v \in V$ *s.t.* $\nexists v' \in V$ *s.t.* $(v', v) \in E$, *i.e., there exists at least one* start node *in* V;
- $\exists v \in V$ *s.t.* $\nexists v' \in V$ *s.t.* $(v, v') \in E$, *i.e., there exists at least one* end node *in* V;
- $\nexists v \in V$ *s.t.* $(v, v) \in E$, *i.e., no self-loops are allowed;*
- $\delta : E \to \Omega$ *is a function that associates a timespan distribution with each edge in the graph, such that* $\forall v \in V \sum_{\{v' \in V | (v,v') \in E\}} S(\delta(v, v')) = 1$.

If $A = (V, E, \delta)$ is an activity and $v \in V$ is a node, $A.p_{max}(v)$ denotes the maximum product of probabilities on any path in A from v to an end node.

Evidence And Identification Problems

This section formalizes the Evidence and Identification problems. Without loss of generality, we assume that observations are stored in a single relational observation table, denoted D. Each tuple t ἐ D corresponds to a single observation, denoted t:obs, which is observed at a given time, denoted t:ts. When our framework is used for realtime activity detection, our proposed insertion algorithm processes each observation as it is received, updates the index, and stores a tuple in the observation table. Conversely, when the framework is used to detect activities in a previously acquired body of data, our bulk insertion algorithm can pull all the observation tuples from the table and build the whole index.

Additionally, in some applications, each observation may be associated with context information (e.g., IP addreess, full name, spatial location), which might help discriminate between observations belonging to different activity occurrences. However, we do not assume this information to be available in general. For instance, in an intrusion detection system, multiple attackers engaged in different activities, may need to perform some common steps, and they may appear to come from the same origin if they use proxies to conceal their real identities. We use t:context to denote context information for observation tuple t, and propose a restriction where two tuples are considered to be part of the same activity occurrence only if their context information is "equivalent." Note that t:context can generally be used to represent the result of the evaluation of a given predicate on t.

id	ts	obs	id	ts	obs
1	1.0	goAccounts	11	7.1	selectPayee
2	2.0	goAccounts	12	8.0	checkBalance
3	3.0	checkBalance	13	8.9	goBillPay
4	3.5	goBillpay	14	9.0	selectSchedule
5	4.0	checkBalance	15	10.0	confirmTransfer
6	4.9	checkBalance	16	10.9	selectSource
7	5.0	selectSource	17	11.0	selectPayee
8	6.0	goBillpay	18	11.1	selectSchedule
9	6.9	selectSource	19	12.0	confirmTransfer
10	7.0	selectPayee	20	13.8	selectPayee

Fig. 2 : Example log (Time is in minutes from the origin 0)

III. Temporal Multiactivity Graph Index

In order to monitor an observation table for occurrences of multiple activities, we first merge all temporal activity definitions from A= $\{A_1 ..., A_k\}$ into a single graph. We use id(A) to denote a unique identifier for activity A and I_A to denote the set $\{ id(A_1),..., id(A_k)\}$.

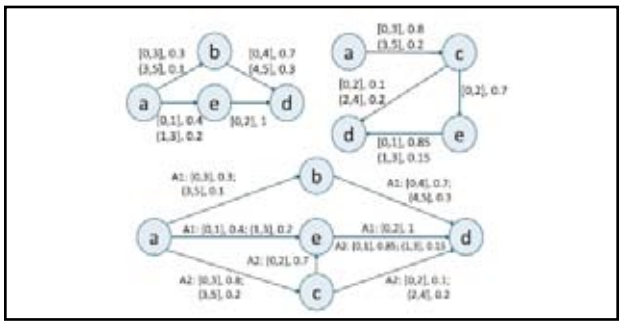


Fig. 3 : Temporal stochastic activities (top) and corresponding multi activity graph (bottom)

Definition 4.1 (Temporal Multiactivity Graph). *Let* $\mathcal{A} = \{A_1, \ldots, A_k\}$ *be a set of temporal stochastic activities, where* $A_i = (V_i, E_i, \delta_i)$. *The* Temporal Multiactivity Graph *for* $\mathcal{A}$ *is a triple* $G = (V_G, I_{\mathcal{A}}, \delta_G)$ *where*

- $V_G = \cup_{i=1,..,k} V_i$ *is a set of observations;*
- $\delta_G : V_G \times V_G \times I_{\mathcal{A}} \to \Omega$ *is a function that maps triples* $(v, v', id(A_i))$ *to the timespan distribution* $\delta_i(v, v')$ *if* $(v, v') \in E_i$, *and* $\emptyset$ *otherwise.*

A temporal multiactivity graph merges a number of stochastic activities. It can be graphically represented by labeling nodes with observations and edges with the ids of activities containing them, along with the corresponding timespan distributions. The temporal multiactivity graph can be computed in time polynomial in the size of A. Furthermore, the temporal multiactivity graph has to be computed only once before building the index. Fig. shows two temporal stochastic activities and the corresponding multiactivity graph.

Temporal Multiactivity Graph Index

Definition (Temporal Multiactivity Graph Index). *Let* $\mathcal{A} = \{A_1, \ldots, A_k\}$ *be a set of stochastic activities, where* $A_i = (V_i, E_i, \delta_i)$, *and let* $G = (V_G, I_{\mathcal{A}}, \delta_G)$ *be the temporal multiactivity graph built over* $\mathcal{A}$. *A* Temporal Multiactivity Graph Index *is a 6-tuple* $I_G = (G, start_G, end_G, max_G, tables_G, completed_G)$, *where*

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

- $start_G : V_G \rightarrow 2^{I_A}$ *is a function that associates with each node* $v \in V_G$*, the set of activity ids for which* v *is a start node;*
- $end_G : V_G \rightarrow 2^{I_A}$ *is a function that associates with each node* $v \in V_G$*, the set of activity ids for which* v *is an end node;*
- $max_G : V_G \times I_A \rightarrow [0,1]$ *is a function that associates with each pair* $(v, id(A_i))$ *the probability* $A_i.p_{max}(v)$*, if* $v \in V_i$*, and 0 otherwise;*

- *For each* $v \in V_G$*,* $tables_G(v)$ *is a set of records of the form* $(current, activityID, t_0, prob, previous, next)$*, where current is a reference to an observation tuple,* $activityID \in I_A$ *is an activity id,* $t_0 \in T$ *is a timestamp,* $prob \in [0,1]$*, previous is a reference and next a set of references to records in* $tables_G$*;*
- $completed_G : I_A \rightarrow 2^{\mathcal{P}}$*, where* $\mathcal{P}$ *is the set of references to records in* $tables_G$*, is a function that associates with each activity identifier* $id(A)$ *a set of references to records in* $tables_G$ *which correspond to completed instances of activity A.*

IV. Tmagic Insertion Algorithm

This section describes an algorithm to insert tuples into the tMAGIC index (Algorithm 1). The algorithm takes as input a temporal multiactivity graph index I_G, a new tuple t_{new} to be added to the index, a probability threshold p_t, and seven boolean flags—f_{MS}, f_{EA}, f_{MP}, f_{MPS}, f_{BP} ,f_{TF} , f_{CTX}—indicating which restrictions and/or pruning strategies must be applied.

Algorithm 1. $insert(t_{new}, I_G, p_t, f_{MS}, f_{EA}, f_{MP}, f_{MPS}, f_{BP}, f_{TF}, f_{CTX})$

Input: New tuple to be inserted t_{new}, temporal multiactivity graph index I_G, probability threshold p_t, Boolean flags f_{MS}, f_{EA}, f_{MP}, f_{MPS}, f_{BP}, f_{TF}, f_{CTX} indicating which restrictions/pruning strategies must be applied.
Output: Updated temporal multiactivity graph index I_G.

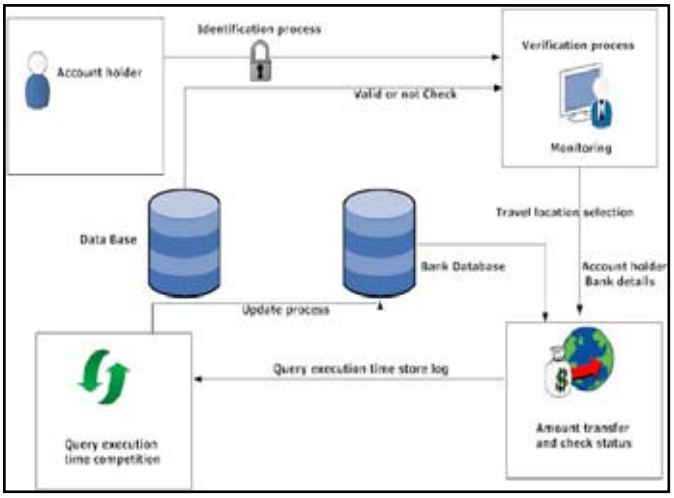
```
1: //Look at start nodes
2: if start_G(t_new.obs) ≠ ∅ then
3:    updatedActivities ← ∅
4:    if f_MS then
5:       for all record r ∈ tables_G(t_new.obs)
         s.t. r.activityID ∈ start_G(t_new.obs) ∧ r.next = ∅ do
6:          r.current ← t_new^↑
7:          updatedActivities ← updatedActivities ∪
            {r.activityID}
8:       end for
9:    end if
10:   for all id ∈ start_G(t_new.obs)\updatedActivities do
11:      add (t_new^↓, id, t_new.ts, 1, ⊥, ∅) to tables_G(t_new.obs)
12:   end for
13: end if
14: //Look at intermediate nodes
15: for all node v ∈ V_G s.t. ∃id ∈ I_A, δ_G(v, t_new.obs, id) ≠ ∅ do
16:    if f_TF then
17:       r_first ← min{r ∈ tables_G(v) | t_new.ts − r.current.ts
          ≤ max_{id∈I_A | δ_G(v,t_new.obs,id)≠∅} max_G^↓(v, t_new.obs, id)}
18:    else
19:       r_first ← tables_G(v).first
20:    end if
21:    for all record r ∈ tables_G(v) s.t. r ≥ r_first ∧ (¬f_EA ∨
       (f_EA ∧ r.next = ∅)) ∧ (¬f_CTX ∨ (f_CTX ∧
       t_new.context ≃ r.current.context)) do
22:       id ← r.activityID
23:       if δ_G(v, t_new.obs, id) ≠ ∅ then
24:          (I, τ) ← δ_G(v, t_new.obs, id)
25:          p ← τ(x, y) where [x, y] ∈ I and x ≤ t_new.ts −
             r.current.ts ≤ y
26:          if (¬f_BP ∧ r.prob · p ≥ p_t) ∨ (f_BP
             ∧r.prob · p · max_G(t_new.obs, id) ≥ p_t) then
27:             r_n ← (t_new^↑, id, r.t_0, r.prob · p, r^↑, ∅)
28:             add r_n to tables_G(t_new.obs)
29:             r.next ← r.next ∪ {r_n^↑}
30:          //Look at end nodes
31:          if id ∈ end_G(t_new.obs) then
32:             if (f_MS ∧ ∃r_c^↑ ∈ completed_G(id), span(r_c)
                ⊆ span(r_n) ∧ r_n.prob ≤ r_c.prob) ∨ (f_MP ∧
                ∃r_c^↑ ∈ completed_G(id), span(r_c) = span(r_n)
                ∧r_n.prob ≤ r_c.prob) ∨ (f_MPS ∧ ∃id_c ∈
                I_A, r_c^↑ ∈ completed_G(id_c), span(r_c) =
                span(r_n) ∧ r_n.prob ≤ r_c.prob) then
33:                discard r_n
34:             else
35:                add r_n^↓ to completed_G(id)
36:                for all id_c ∈ I_A do
37:                   for all r_c^↓ ∈ completed_G(id_c) s.t. (f_MS ∧
                      id_c = id ∧ span(r_n) ⊆ span(r_c) ∧
                      r_c.prob ≤ r_n.prob) ∨ (f_MP ∧ id_c =
                      id ∧ span(r_n) = span(r_c) ∧ r_c.prob ≤
                      r_n.prob) ∨ (f_MPS ∧ span(r_n) =
                      span(r_c) ∧ r_c.prob ≤ r_n.prob) do
38:                      remove r_c^↓ from completed_G(id)
39:                      delete(r_c)
40:                   end for
41:                end for
42:             end if
43:          end if
44:       end if
45:    end if
46: end for
47: end for
```

Lines 2-13 handle the case when the tuple contains an observation that is the start node of an activity in A. If MS must be applied, the current pointers of records in $tables_G(t_{new}$:obs) that do not have a successor are replaced with pointers to the new tuple, in order to minimize the span (Lines 4-9). If a record corresponding to a start node already has a successor, its current stays unchanged and a new record is added to the table for every activity in $start_G$(tnew:obs) for which no record was updated (Lines 10-12), denoting the fact that the new observation may be the start of a new activity occurrence.

Lines 15-29 look at the tables associated with the nodes that precede t_{new}.obs in the temporal multiactivity graph and check whether the new tuple can be linked to existing partially completed occurrences. For each predecessor v of t_{new}.obs, Lines 16-20 determine where the algorithm should start scanning $tables_G(v)$. Note that records are added to the index as new observations are received. Therefore, records r in each index table $tables_G(v)$ are ordered by r.current.t_s, i.e., the time at which the corresponding observation was received. Given two records r1, r2 £ $tables_G(v)$, we use r1 ≤r2 to denote the fact that r1 precedes r2 in $tables_G(v)$, i.e.,r1. current.ts≤r2:current:ts.In the unrestricted case, the whole table is scanned $tables_G(v)$,:first being the first record in $tables_G(v)$,. If TF

is being applied (see Section 4.2),only the "most recent" records in tablesGðvÞ are considered. Additionally, if the EA restriction is to be applied, the algorithm requires each record to have at most one successor; each record is linked to the first observation that is a valid successor (Line 21, where the context restriction is applied as well, if needed). On Line 25, timespan distributions are used to determine the probability that observation tnew:obs is the successor of r:current:obs for the activity definition in A identified by id =r:activityID, given the amount of time elapsed between r:current:ts and tnew:ts. We also enforce the probability threshold (Line 26) by checking whether the partial occurrence still has a probability above the threshold. Alternatively, we detect whether the partial occurrence can still have a probability above the threshold on completion: we refer to this feature of the algorithm as Best Path (BP) pruning strategy, as it allows us to prune away solutions based on the best possible path to an end node. If all these conditions are met, a new record rn is added to the table associated with t_{new}.obs and r:next is
updated to point to rn (Lines 27-29). Note that rn inherits t0 from its predecessor; hence, the start/end times can be quickly retrieved by looking directly at the last tMAGIC record for a completed occurrence.
Lines 31-43 check whether tnew:obs is an end node for some activity. If yes, the algorithm checks if any of MS, MP, or MPS must be applied, and whether conditions for inclusion of the newly completed occurrence in the index under such restrictions are violated2 (Line 32). If so, record rn is discarded (Line 33); otherwise a pointer to rn is added to completedG, saying that a new occurrence has been completed (Line 35). In addition, if any of MS, MP, or MPS must be applied, previously completed occurrences rendered invalid after the addition of the new occurrence are removed from completedG and their records are removed from the index (Lines 36-41), using a function delete which recursively deletes records following the chain of previous pointers.

V. Architecture



Account Identification

Account Identification process for user creates an account process. In this process is used to access only the valid user. Service provider can easily identify if any problem created by unwanted user's .And this method get current user using system IP also for verification process. In this module get account holder information. Account identification is secure process is main objective.

Action Monitor

Action monitor modules working process is monitoring account status of user. If users search travel source and destination process which location search to user to monitoring admin process .In this module monitoring user search processlog has been stored in database table. This stored method user searching process has been stored .Monitoring process is analyze the running time of user access current status update and view.

Log Mechanism

Logging mechanism is mainly used to monitoring all process. In this modules storing and verification process of user log's details. Two types of logging mechanism stored in this process. User In and out mechanism, if user accesses any page and clicks any event can be stored. In this method is very useful and strong evidence for user and owner .Log mechanism stored individual user log create in SQL database table.

Query Execution

Query execution process work in account holder bank details checking process log has stored. In this module check bank process only verification and checking account holder full details. Verification process is check query execution, time calculation and store database. In this module detect unauthorized person enter or not .Ifhacker enters in our page detect and store event in database.

Action comparison

Action comparison process is compare tobank process. Bank side set timestamp to increase and how many time try this process calculate in action comparisonmodule. If any different to normal active on that account holder access log to verify process. After store all Query execution log store in single table .Then effective view in user process in graph generation. If any time differ in any account holder on that user error has been also generate in graph.

VI. Experimental Results

This section describes experiments on both synthetic and real data to evaluate index creation time, memory usage and running time for the two problems described in the paper. We first ran experiments on synthetic data in order to evaluate several aspects of the framework in detail. We then used a third party real-world data set to validate the results obtained on synthetic data, and highlight the differences between the two scenarios. For the first set of experiments, we used synthetic activity definitions and data sets generated using two separate tools: one for generating random activity definitions, and another for simulating a set of activities and generating observation streams. We generated 32 activity definitions, and data sets of 5 million tuples each. The second set of experiments used a third party proprietary real-world travel data set including events such as hotel check-ins and check-outs, flight bookings, departures, arrivals, etc. The data set includes over 10 million records collected over a period of 2 years, and includes names of the individuals involved.

VII. Conclusions

This paper studies the problem of automatically and efficiently detecting activities in very large observation databases collected by systems such as web servers, banks, and security installations. We proposed temporal stochastic automata to model activities of interest and defined a data structure, called a temporal multiactivity graph, to merge multiple activity graphs together and enable concurrent monitoring of multiple activities. We introduced the temporal multiactivity graph index to index very large numbers of temporal observations from interleaved activities. We have designed algorithms to build the tMAGIC index and we have

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

shown how the tMAGIC index can be leveraged to develop algorithms to efficiently solve the Evidence and Identification problems. Finally, we have introduced complexity reducing restrictions and pruning strategies to efficiently solve these problems. Our experiments have shown that tMAGIC consumes reasonable amounts of memory and can quickly solve both the Evidence and Identification problems in both synthetic and a real-world data set.

References

[1] *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.*

[2] *M. Albanese, V. Moscato, A. Picariello, V.S. Subrahmanian, and O. Udrea, "Detecting Stochastically Scheduled Activities in Video," Proc. 20th Int'l Joint Conf. Artifical Intelligence (IJCAI '07), pp. 1802- 1807, Jan. 2007.*

[3] *M. Albanese, A. Pugliese, V.S. Subrahmanian, and O. Udrea, "MAGIC: A multiactivity Graph Index for Activity Detection," Proc. IEEE Int'l Conf. Information Reuse and Integration (IRI '07), pp. 267-278, Aug. 2007.*

[4] *A. Arasu, S. Babu, and J. Widom, "The CQL Continuous Query Language: Semantic Foundations and Query Execution," Int'l J. Very Large Data Bases, vol. 15, pp. 121-142, June 2006.*

[5] *J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram, "Human Activity Recognition Using Multidimensional Indexing," IEEE Trans. Pattern Analysis Machine Intelligence, vol. 24, no. 8, pp. 1091-1104, Aug. 2002.*

[6] *S. Chandrasekaran, O. Cooper, A. Deshpande, M.J. Franklin, J.M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M.A. Shah, "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," Proc. Conf. Innovative Data Systems Research (CIDR '03), 2003.*

[7] *T.V. Duong, H.H. Bui, D.Q. Phung, and S. Venkatesh, "Activity Recognition and Abnormality Detection with the Switching Hidden Semi-Markov Model," Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '05), 2005.*

[8] *L. Golab and M.T. O ¨ zsu, "Issues in Data Stream Management," ACM SIGMOD Record, vol. 32, pp. 5-14, June 2003.*

[9] *R. Hamid, Y. Huang, and I. Essa, "ARGMode Activity Recognition Using Graphical Models." Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '03), 2003.*

[10] *B. Kerkez, "Learning Plan Libraries for Case-Based Plan Recognition," Proc. Midwest Artificial Intelligence and Cognitive Science Conf. (MAICS '02), Apr. 2002.*

[11] *S. Lu¨ hr, H.H. Bui, S. Venkatesh, and G.A.W. West, "Recognition of Human Activity through Hierarchical Stochastic Learning," Proc. IEEE First Int'l Conf. Pervasive Computing and Comm. (PerCom '03), pp. 416-422, Mar. 2003.*

[12] *F. Mo¨ rchen, "Unsupervised Pattern Mining from Symbolic Temporal Data," SIGKDD Explorations Newsletter, vol. 9, no. 1, pp. 41-55, June 2007.*

[13] *F. Reiss, K. Stockinger, K. Wu, A. Shoshani, and J.M. Hellerstein, "Enabling Real-Time Querying of Live and Historical Stream Data," Proc. 19th Int'l Conf. Scientific and Statistical Database Management (SSDBM '07), 2007.*

[14] *K. Seymore, A. McCallum, and R. Rosenfeld, "Learning Hidden Markov Model Structure for Information Extraction," Proc. Workshop Machine Learning for Information Extraction (AAAI '99), 1999.*

[15] *V.T. Vu, F. Bre´mond, and M. Thonnat, "Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition," Proc. 18th Int'l Joint Conf. Artificial Intelligence (IJCAI '03), pp. 1295-1302, Aug. 2003.*