A Methodology for Extracting Temporal Properties from Sensor Network Data Streams

Dimitrios Lymberopoulos
Networked Embedded Computing Lab
Microsoft Research
Redmond, WA, USA
dlymper@microsoft.com

Athanasios Bamis, Andreas Savvides
Electrical Engineering
Yale University
New Haven, CT, USA
firstname.lastname@yale.edu

ABSTRACT

The extraction of temporal characteristics from sensor data streams can reveal important properties about the sensed events. Knowledge of temporal characteristics in applications where sensed events tend to periodically repeat, can provide a great deal of information towards identifying patterns, building models and using the timing information to actuate and provide services. In this paper we outline a methodology for extracting the temporal properties, in terms of start time and duration, of sensor data streams that can be used in applications such as human, habitat, environmental and traffic monitoring where sensed events repeat over a time window. Its application is demonstrated on a 30-day dataset collected from one of our assisted living sensor network deployments.

Categories and Subject Descriptors

I.5.3 [**Computing Methodologies**]: Pattern Recognition-Clustering[similarity measures]

General Terms

Algorithms, Experimentation, Human Factors

Keywords

sensor networks, data stream, temporal structure, assisted living

1. INTRODUCTION

Today's sensor network technologies can reliably collect location, time and duration (l, t, d) data streams, but there is currently lack of understanding of how to exploit this information to provide reliable services with respect to the phenomenon being sensed. This is a challenging task because each sensor data stream has different timing behavior that spans over multiple spatio-temporal scales.

Spatial context usually denotes the location of a sensed event; its meaning is closely coupled to a physical place and it is relatively easy to understand. Temporal context, however, frequently depends on two parameters, start-time and duration (t, d), and it is not always straightforward to understand. This is mainly because the occurrence of an event at a specific location, may also have a special meaning with respect to a time-window, that can only be extracted when multiple instances of the time window are considered. Furthermore, since the start time and duration parameters take continuous values, extracting the temporal context of events requires proper time and duration discretization.

This paper describes a methodology for automatically extracting the temporal structure of an arbitrary sensor data stream and demonstrates its importance in building the spatiotemporal model of the sensor data stream. Our methodology allows us to detect temporal patterns in the data with respect to a given time-window by properly extracting time, duration and frequency information across different time windows. The proposed method assumes no information about the sensor data stream or its source and can be applied to a wide range of applications, particularly the ones that describe activity modeling. In our research, we use this methodology to extract daily activity patterns of elders in an assisted living setup [11],[9]. The same solution however could be applied to many other application domains to extract temporal information about visited GPS locations, traffic congestion, workload of data center servers, cell-phone user activity and more. For example, in the personal safety system using GPS phones we developed in [21],[19], this method could be used yo extract the habits of a person from GPS data and apply it towards safety goals.

The main contribution of this work is a four step process for extracting the temporal characteristics of a data stream. First, sensed events are labeled by spatial context into distinct event types. For each event type, we temporally organize different event instances into time windows of fixed duration and express the initial data stream as a sequence of different time-windows. The resulting stream is then relabeled by applying a clustering algorithm that reclassifies (relabels) the events within each type according to temporal context parameters (t, d).

The proposed method is evaluated in the context of an assisted living wireless sensor network deployment that monitored an elder person living alone in his house over a period of 30 days. After applying the algorithm presented in this paper on the movement/activity patterns recorded, we were able to automatically extract the temporal structure of the different activities that the monitored person was engaged into and use this structure to build a detailed spatiotemporal model of the person's daily activity.

MobiSys'09, June 22–25, 2009, Kraków, Poland.

2. MOTIVATION

The main challenge in extracting temporal characteristics lies on the fact that time and duration of a sensed event, are continuous variables that can take any value. To consider them in a model, these quantities need to be appropriately discretized. In doing so however, one needs to consider the fact that temporal characteristics may differ in two ways. Temporal variations within one particular event's time, and temporal variations across different event types. Even worse, these characteristics for a given sensing event might completely change over time in a given sensor network deployment or even across different network deployments. Because of this, extracting a set of discrete time and duration parameters that best describe a sensing event across different event instances is not trivial.

To better illustrate this, let us consider a realistic example drawn from our assisted living testbed. Assume a sensor network is deployed inside a house to continuously monitor an elder person living alone. The goal of the network is to provide information with respect to *when* and *for how long* the monitored person engages into specific activities such as sleeping, cooking, watching TV and visiting the bathroom. In this case, we wish to identify the temporal characteristics of the recorded sleep activity instances, that is start time and duration. One could argue that the answer to this question is trivial since most elder people sleep early during the night (10pm-12am) and wake up early in the morning (7-8am). However, this might not always be the case. Figure 1 shows the time and duration characteristics of all the *Sleep* activity instances of a monitored elder person as recorded in our home sensor network deployment over a period of 30-days. *The frequency at which the Sleep instances appear in a given time and duration window, indicates when and for how long the person goes to sleep.* Figure 1 verifies the initial intuition showing that the person usually sleeps before midnight for a duration of 9 to 10 hours. However, most of the times, the person interrupts his sleep to visit the bathroom either in the middle of the night (time window 0 in Figure 1) or early in the morning (time window 6 in Figure 1). These temporal characteristics of sleep activity provide very important information about the monitored person's habits and are often difficult to identify based on a-priori or "common-sense" knowledge. In addition, this information is specific to the monitored person and to the time window at which he was observed. In general, the temporal characteristics of sleep activity change over time and vary across different people.

From these simple observations we can identify the fundamental properties and challenges of our problem:

1. The time and duration characteristics of routine events are not independent. For example, in Figure 1, *Sleep* activity instances at different time-windows have different duration characteristics. Furthermore, only a small number of time and duration window combinations appear frequently. *As a result, time and duration information have to be jointly considered.*

2. The answer to the question *"when and for how long an event takes place"* might not be unique and the number of answers might not be known. For instance, in Figure 1 there are at least 3 different answers to the question *"when and for how long the person goes to sleep".*

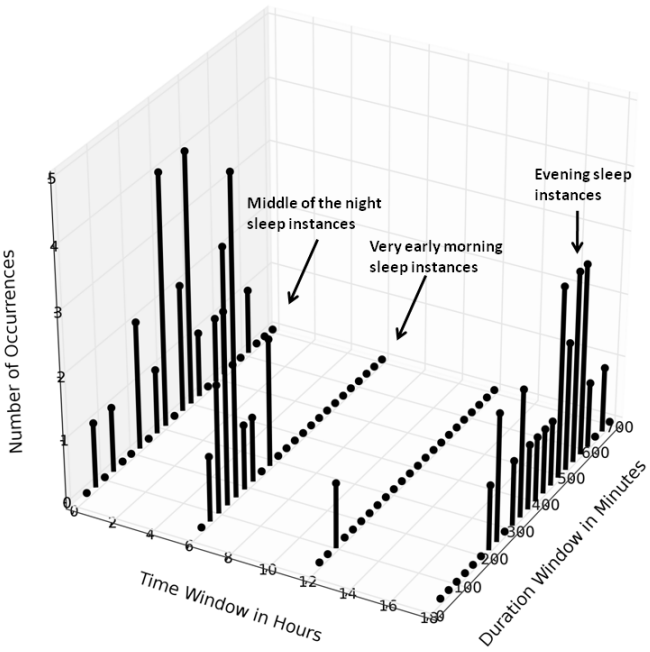


Figure 1: Time and duration characteristics of the Sleep activity. Time is divided into 4 time windows of 6 hours duration each. Duration is divided into 30-minutes windows. The z-axis represents the number of times the Sleep activity appeared in a specific time and duration window.

3. The temporal characteristics of a specific event change over time. For example, in the case of home monitoring applications one can note that temporal attributes of the recorded daily activity cycles(i.e. sleeping cycles like the one shown in Figure 1) are specific to the monitored person and the time period they were collected. Daily activity cycles vary across different people and change over time even for the same person.

4. The way time and duration information is organized in time windows is important. Using time and duration windows of different sizes in Figure 1 could significantly impact the quality of the extracted information. Since our goal is to discover these time and duration characteristics, we should not assume any a-priori time and duration discretization similar to the one shown in Figure 1. Instead, the temporal characteristics of the sensed events should be used to automatically discover the best time and window sizes.

Thus, a data driven approach that is able to automatically discover the temporal properties of the sensed events assuming no a-priori information about the event or its source is needed. The goal of this process is to provide an answer to the question: *"when and for how long does this event type take place?"*. In the example used in this section, sleep activity instances were the actual sensed events. In other application domains, visited GPS locations, traffic congestion, workload of data center servers, and basically any measured quantity over time can take the form of sensed events.

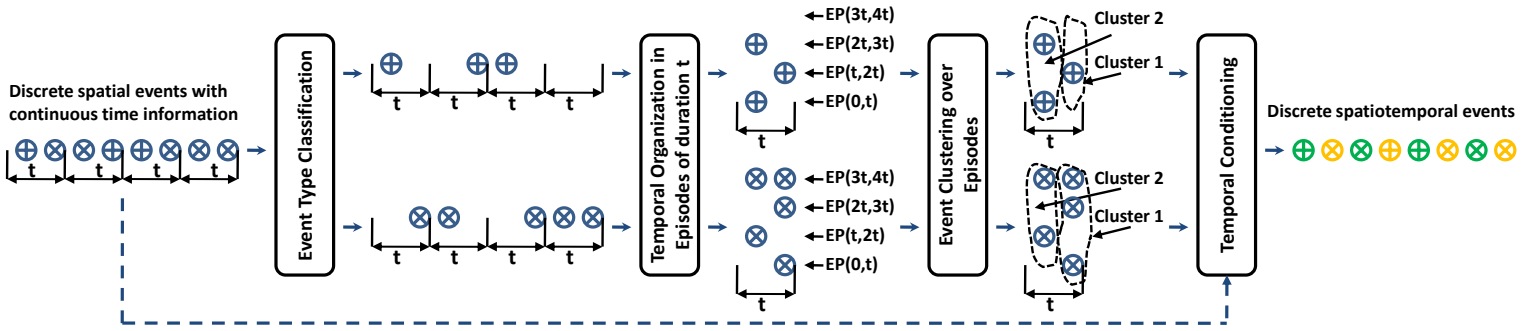


Figure 2: Overview of the proposed methodology for extracting the temporal structure of the sensor data stream. Our approach converts a stream of discrete sensing events with continuous information to a stream of discrete spatiotemporal events without assuming any information about the type or the source of the input sensor data stream.

3. PROPOSED APPROACH OUTLINE

From the previous example, the proposed solution outline is beginning to emerge. Events in the sensor data stream come as different types based on either the type of sensing features recorded or the location of the sensor. This abstracts away the spatial context in the sensor data stream. Temporal context suggests that events of the same type come in different instances, that are correlated across time windows. *To discover these temporal correlations across time windows, we properly mine time, duration and frequency information* in the following way (Figure 2):

1. Encode spatial context by separating events into types that carry information about sensor features and/or locations.

2. For each event type, temporally organize sensed events based on a given time window (i.e day, week, month or whatever else is appropriate for the application). The size of the time window (day,week,month etc.) determines the nature of the temporal characteristics (daily,weekly,monthly etc.) that will be extracted. This step transforms a single sequence of sensed events over time into multiple sensed event sequences over a given time window.

3. The goal now becomes to properly mine time, duration and frequency information across the multiple sensed event sequences. In our approach, the key for extracting the spatiotemporal properties across the different event sequences is frequency. We aim to discover all the event instances with similar start time and duration characteristics that repeatedly appear across the different event sequences generated at the previous step. The more frequently the event appears with a specific start time and duration, the more important these two parameters are for this event type. To classify events with respect to time and duration, we first formulate the problem as a clustering problem of event instances, where each event instance is represented as a one-dimensional line segment uniquely identified by its start time and duration (Section 4). The high-level goal of the clustering algorithm is to create groups/clusters of large numbers of event instances with very similar start time and duration characteristics (Section 5).

4. Each cluster generated by our clustering algorithm represents a pair of frequently repeating start times and durations. Given the discovered start time and duration parameters for each event type, we condition each event instance to produce a sequence of spatiotemporal events. *In that way, we move from a sequence of spatial events with continuous time and duration information to a sequence of discrete spatiotemporal events.*

Given the sequence of discrete spatiotemporal events any data mining algorithm can be used [8, 1, 2, 17, 5, 16, 14, 13, 5, 16, 3, 6, 14, 15] to extract the statistical associations of the different spatiotemporal event types in the sensor data stream. The role of this process is to discover how *different discrete* event types (as opposed to identical events with different continuous time properties) correlate across different time windows. These statistical associations can be used to directly extract the spatiotemporal model of the sensor data stream. This model represents the spatiotemporal structure of the sensed phenomenon and can be used to predict future events or detect unusual event sequences at run-time. Our previous work has already addressed these parts of data model extraction methodology, from time information encoding in sensor data streams [11] to model extraction [8]. In this paper, we focus on the extraction of the temporal structure of the sensor data stream (Figure 2). Our proposed approach is described in the sections that follow.

4. NETWORK MODEL

The events generated by the sensor network might be primitive sensing features identified at the node level ("temperature exceeded a threshold", "Joe is in the kitchen" etc.) or more complex events identified at the network level ("There is a fire in this area of the network", "Joe is enjoying his coffee in front of the TV after dinner", etc.). The output of the network can thus be considered as a collection of different events of interest: $\{E_1, E_2, E_3, \ldots, E_N\}$. For our discussion, we will initially consider only a single event type, say E_1. With only this event type, the output of the sensor network is a collection of event instances over time:

$$\{E_1(t_1, d_1), E_1(t_2, d_2), E_1(t_3, d_3), \ldots, E_1(t_n, d_n)\} \quad (1)$$

Note that every event instance is associated with two basic temporal characteristics: *time* and *duration*. *Time* refers to the absolute time when the event started taking place and *duration* refers to the time duration that this event lasted

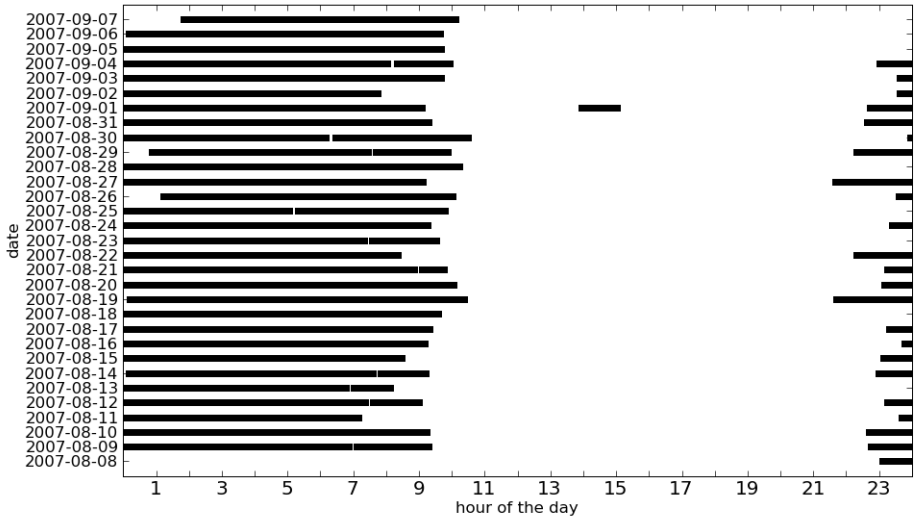


Figure 3: The different *sleep activity* instances of the monitored elder person over a 30-day period. The y-axis shows the date when the activity instance was recorded and the x-axis shows the ground truth time.

(e.g. how long the temperature exceeded the threshold or how long the fire lasted).

We define an episode $EP(T_{start}, T_{end})$ as the collection of all event instances that took place between T_{start} and T_{end}:

$$EP(T_{start}, T_{end}) = \{E_1(t_i, d_i) | T_{start} \leq t_i \leq T_{end} \forall i\} \quad (2)$$

Episodes are nothing more than a temporal grouping of events. In that way, the output of the sensor network can be defined as a sequence of episodes (time windows) with identical durations:

$$< EP(T_1, T_2), EP(T_2, T_3), \ldots, EP(T_{N-1}, T_N) > \quad (3)$$

where $T_2 - T_1 = T_3 - T_2 = \ldots = T_N - T_{N-1}$. The time duration of each episode can be selected according to the needs of the application. For instance, it could be a single day, a week, a month or even a year depending on the application requirements. The only restriction we enforce is that the time durations of all episodes in equation (3) are equal.

To better illustrate event representation and temporal grouping of events in episodes, let us consider again the example of monitoring the sleeping activity of an elder person living alone in his house. Figure 3 shows all the sleep activity instances recorded over the 30-day monitoring period assuming that the duration of each episode is a single day. The x-axis represents the actual time in the context of a single episode, in this case a day. The y axis represents the different episodes. Note that at each episode (day) one or more sleep instances are recorded and denoted as one-dimensional line segments uniquely identified by the start time and duration of the corresponding sleep instances. As a result, our 30-day sensor data stream is represented as a collection of 30 different episodes of a single day duration each. Each episode contains a number of sleep instances which are represented as one-dimensional line segments over time.

Note that in Figure 3, several of the line segments in a single episode are very briefly interrupted. This usually corresponds to other events, such as bathroom visits that are not shown in Figure 3, taking place in the middle of the night or early in the morning. Depending on the context of the monitored person, these interruptions might or might not be important. For elder people, knowing the frequency and duration of bathroom visits that interrupt sleeping activity is important for medical purposes. Therefore, the interrupted sleeping activity instances are considered to be different. In other cases, these sleeping activity instances might be considered a single, continuous sleeping activity. This issue can be automatically handled in our interpretation framework described in [9],[10]. The methodology described in this paper is independent of these decisions and can be transparently applied to any temporally organized collection of line-segments like the one shown in Figure 3.

5. TEMPORAL STRUCTURE DISCOVERY OF SENSOR STREAMS

Our goal is to find the *frequent temporal characteristics* of an event across the different episodes. Depending on the time duration of each episode this corresponds to finding the daily, weekly, monthly or yearly frequent temporal characteristics. The temporal characteristics are (a) the actual time when the event starts and (b) the duration of this event. For instance, finding the daily temporal characteristics of the sleep activity instances shown in Figure 3, corresponds to finding all the *frequent* times during the course of a day that this event will take place with a specific duration. *We call the temporal characteristics of an event frequent when this event takes place around the same time and with almost the same duration over a large number of episodes. Note that absolute time and duration are not necessarily independent.* If the starting times of two event instances are identical but their durations are very different and vice versa, then the two instances might not be related.

Problem Statement: *Given a sequence of episodes where each episode contains a collection of instances $E(t, d)$ of a specific event E, our goal is to identify the pairs of starting time indexes and durations $(t_i, d_i), i = 1, 2, \ldots$ during which the event of interest E occurs repeatedly across different episodes.*

In practice, the problem of finding the frequent temporal characteristics of the recorded event instances is a clustering problem. For instance, identifying when and for how long the monitored elder person sleeps during the course of the day corresponds to identifying the clusters of line segments with similar start time and duration characteristics across different episodes in Figure 3. Ideally, all sleep instances would have identical start time and durations producing a single cluster as output. In the worst case, sleep instances would be uniformly distributed over time producing a large number of clusters equal to the number of event instances. In most of the cases, however, the collection of event instances would look like the one in Figure 3, where several event instances overlap with each other over time in many different ways. Note that in this case it is not trivial to identify what the clusters of the input collection of line segments should be.

Even though several clustering algorithms have already been proposed in the literature in the context of several application domains, the clustering problem considered in this paper differs in two ways. First, instead of clustering points in space we want to cluster line segments in time. *These line segments are defined by two different but closely related temporal attributes: the start time and the duration of a given event instance.*Second, in contrast to other previously proposed approaches in computational geometry and computer vision, the line segments are strictly one-dimensional.
Posing our problem as a clustering problem of one-dimensional line segments raises a series of issues:

1. How do we quantify the similarity of two event instances that are represented as one-dimensional line segments? Note, that the similarity is a function of both temporal characteristics (time and duration) of the two event instances.

2. How do we quantify the quality of a cluster of event instances? In other words, how do we know if an arbitrary cluster of event instances is frequent or not?

3. Given a quality metric for an arbitrary cluster of event instances and a recorded collection of event instances, how can we construct the frequent clusters?

The key to answering these questions lies on examining two important properties of the input line segments:

1. The amount of overlap between line segments. As Figure 3 shows, the higher the overlap between two line segments the more similar the temporal characteristics of these two line segments are.

2. The fraction of the overall number of line segments that are similar. The more similar the line segments are,the more frequently repeating the time and duration characteristics of these line segments are. For instance, in Figure 3 most of the sleep instances start taking place between 10pm and midnight, clearly indicating when the person usually goes to sleep.

In the rest of the paper, we describe in detail how our clustering algorithm exploits these properties of the input line segments to efficiently cluster them. To identify the recurring event instances, we first define a similarity metric and then develop a clustering algorithm that can cluster frequently similar events across different episodes according to the proposed metric.

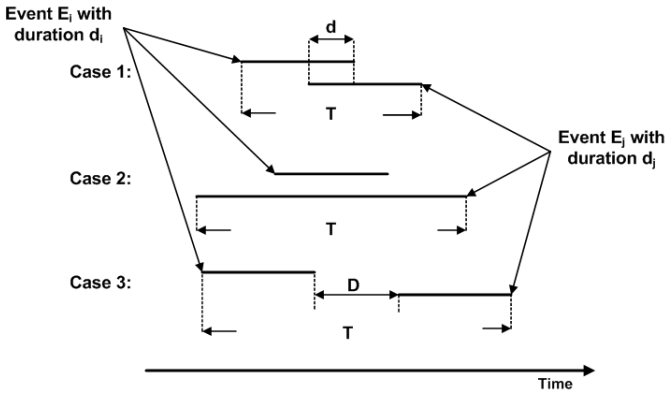


Figure 4: The 3 possible relative positions of two arbitrary event instances. Note that swapping events E_i and E_j results to the exact same pairwise density according to equation 5.

5.1 Pairwise Density of Event Instances

To properly identify recurring event instances, our clustering algorithm first needs a metric for quantifying how similar the temporal characteristics of two arbitrary event instances are. To clarify the properties that such a metric should take into account, consider Figure 4 where all the possible cases of relative positions of two arbitrary event instances (E_i, E_j with start times t_i,t_j and durations d_i and d_j respectively) are shown. In the first case, the two event instances partially overlap. In the second case, one of the event instances is fully overlapped by the other one and in the third case the two event instances are at a distance D in time and they do not overlap at all. Our goal is to define a metric for quantifying the similarity of two arbitrary event instances independently of their relative positions over time (cases 1, 2 or 3).

Given that each event instance is represented as a line segment defined by its starting time and duration, we can quantify how similar two event instances E_i and E_j are as follows:

$$D_{ij} = \frac{T - d_{ij}}{T} \tag{4}$$

where T is the time span of both event instances and d_{ij} is the distance between the event instances defined as:

$$d_{ij} = |t_i - t_j| + |(t_i + d_i) - (t_j + d_j)| \tag{5}$$

where (t_i, d_i) and (t_j, d_j) are the starting time and duration parameters associated to event instances E_i and E_j respectively. The first term in equation (5) captures the difference in terms of the time when these event instances start taking place. The second term captures the difference in terms of the duration between the two event instances with respect to the event instances start times.

Even though equation (5) provides a notion of how close two event instances are, it does not capture accurately the *density* of the two event instances. This is done in equation (4) where the time span of the two event instances is taken into consideration. Figure 4 shows how the time span T is defined in all possible relative positions of two event instances. Equation (4) provides a notion of how close the two event instances are to each other with respect to their time span.

To better illustrate this, let us compute the proposed metric for all the different cases shown in Figure 4:

$$D_{ij}^{case1} = \frac{d}{d_1 + d_2 - d} \tag{6}$$

$$D_{ij}^{case2} = \frac{d_1}{d_2} \tag{7}$$

$$D_{ij}^{case3} = \frac{-D}{D + d_1 + d_2} \tag{8}$$

For the first two cases (equations (6) and (7)), the proposed metric is defined as the overlap of the two event instances divided by their time span. Note, that this is nothing more than the *temporal density* of the two event instances. In both cases, the larger the overlap (given a fixed time span) or the smaller the time span (given a fixed overlap) the higher the D_{ij}. Ideally, D_{ij} becomes equal to 1 when the two line segments are identical (start and end at the same time). In this case the time span of both line segments is equal to the overlap of the two line segments. When there is absolutely no overlap between the two event instances ($d = 0$ in the first case), D_{ij} becomes 0.

In the third case (equation (8)), where there is no overlap between the two event instances, the proposed metric is defined as the distance D between the two event instances (shown in Figure 4) divided by their time span T. Note that now the metric is negative. This is due the fact that d_{ij}, as defined in equation (5), is now larger than the time span T in the third case shown in Figure 4. *The physical explanation behind this is that the proposed metric in equation 4 treats the distance D between the two event instances as a negative overlap between the two event instances. In other words, whenever the two event instances do not overlap, their distance D becomes a virtual negative overlap.* Note, that according to equation (8), the larger the distance D is, the smaller the metric that approaches to but never reaches -1. Conversely, when D is very small and close to 0 the metric becomes equal to 0.

Hence, the metric proposed in equation (4) manages to describe the density of any pair of event instances by assigning values in the range $(-1, 1]$. *The higher the metric value the more dense the pair of event instances is.* We call this the *pairwise density metric.*

To avoid dealing with negative values for the rest of our discussion we will map the metric's range of values from $(-1, 1]$ to $(0, 2]$ by simply using: $1 + D_{ij}$ instead of D_{ij}:

$$D_{ij} = 1 + \frac{T - d_{ij}}{T}, 0 < D_{ij} \leq 2 \tag{9}$$

5.2 Density of Arbitrary Clusters

Given equation (9) as a measure of the similarity/density between two arbitrary event instances, we can now proceed and quantify the quality of a cluster with an arbitrary number of event instances. The main properties that such a metric should take into consideration are the following:

1. The quality of a cluster is high when all of its event instances share similar temporal characteristics as this is defined by equation (9). The more similar the event instances are in terms of their time and duration the higher the quality of the cluster.
2. The proposed metric should be used to compare the quality of any two clusters that might even contain different number of event instances. In other words, the proposed metric should take into consideration the number of event instances a cluster contains. The more event instances are included, the higher the quality of the cluster.

Note, however, that these two properties are not independent. Simply adding event instances into a cluster should not just increase its quality. The reason is that the quality of the cluster should also be affected by the similarity across all the event instances in the cluster. Therefore, blindly adding event instances to a cluster might decrease its quality. In the same sense, keeping the number of event instances in a cluster small to keep the pairwise density of events high should not increase the overall quality of the cluster. The reason is that our goal is not to just find recurring temporal characteristics but to identify *frequent temporal characteristics.*

The following density-based cluster metric captures both of these properties and their tradeoffs:

$$Q_{C_k} = \begin{cases} \frac{\sum_{i<j} D_{ij}}{\binom{n_{C_k}}{2}} \times \frac{n_{C_k}}{n} = \frac{2\sum_{\forall i,j}^{i \neq j} D_{ij}}{n \times (n_{C_k} - 1)} & \text{if } n_{C_k} \geq 2 \\ \frac{1}{n} & \text{if } n_{C_k} = 1 \end{cases} \tag{10}$$

where Q_{C_k} is the quality of cluster C_k, n_{C_k} are the number of event instances in cluster C_k and n is the total number of instances that have been recorded independently of the cluster where they belong. The first term in equation (10) represents the *average pairwise density of the cluster.* For every pair of event instances, the pairwise density is computed according to equation (9). All the pairwise densities are summed and then divided by the total number of pairs of event instances in the cluster giving us the *average pairwise density of the cluster* which can take any value in the range $[0, 1]$. The second term in equation (10) is used to scale the average pairwise density of the cluster with respect to the fraction of the total number of event instances that belong to the cluster and takes values in the range of $[\frac{1}{n}, 1]$. The purpose of the second term is to favor the clusters that contain large numbers of event instances. The reason is that as the number of event instances in a cluster increases, it becomes more difficult to concurrently increase the average pairwise density of the cluster. By multiplying these two terms, we manage to automatically control the tradeoff between the number of event instances in the cluster and the average pairwise density of the cluster.

Given any number of clusters with any number of event instances each, we can quantify their quality using equation (10). *The higher the quality of the cluster, the higher the probability that this cluster is a frequent cluster and therefore the higher the probability that it contains events that share similar frequent temporal characteristics.*

5.3 Automatic Extraction of Frequent Clusters

Being able to quantify the quality of an arbitrary cluster allows us to build an efficient algorithm for automatically finding the frequent clusters of event instances given any input sequence of event episodes similar to equation (3). The main problem in doing this is the fact that we do not know

in advance the total number of clusters. In other words, besides building the frequent clusters, our objective is to also identify the number of clusters. Since we have quantified the quality of an arbitrary cluster, we can concurrently build and identify the number of clusters using the following expression:

$$\frac{1}{N_C + n} \times \sum_{i=1}^{N_C} Q_{C_i} \tag{11}$$

where N_C is the total number of clusters, n is the total number of event instances to be clustered and Q_{C_i} is the quality of cluster C_i. Given a collection of clusters, equation (11) provides an estimate of its quality. In general, the higher the value of expression (11) the better the quality of the collection of clusters. As a result, given two different collection of clusters (with possibly different number of clusters), equation (11) can be used to choose the best one.

Note that even though equation (11) can be seen as the *average cluster density across all the clusters* it does not represent the actual average cluster density. The term $\frac{1}{N_C+n}$ is used instead of $\frac{1}{N_C}$. The reason is that the value of $\frac{1}{N_C}$ varies significantly for different values of N_C and when N_C is small. For instance, consider the case where we have two possible clustering schemes for the collection of events: one with 3 clusters and another one with only 2 clusters. The term $\frac{1}{N_C}$ corresponds to $\frac{1}{3}$ and $\frac{1}{2}$ for the two clustering schemes, which gives us a ratio of 1.5. This means that without even considering the actual densities of the individual clusters, using 2 clusters is 1.5 times better than using 3 clusters. As a result, using the term $\frac{1}{N_C}$ could alter the results of our approach, since it would always favor creating a small number of clusters and therefore ignore the importance of the individual cluster's density being high. To deal with this problem and knowing that $1 \leq N_C \leq n$, we opted to use $\frac{1}{N_C+n}$ instead of $\frac{1}{N_C}$. In that way, the first term in equation (11) is no longer changing significantly for different values of N_C and when N_C is small. In the rest of the paper we will refer to equation (11) as the *average cluster density across all the clusters*.

Having introduced the notion of *average cluster density across all the clusters* we can proceed and design the efficient algorithm shown in Figure 5 for identifying the frequent clusters of the sensed events based on their temporal characteristics. Initially every event instance becomes a cluster. Therefore, assuming we have n event instances, initially we have n clusters each with a quality equal to $\frac{1}{n}$. At the next step, we compute the quality of all the clusters that could be produced by merging any pair of current clusters and we quantify the effect of every such merging on the average cluster density (equation (11)). The pair of current clusters that if combined gives us the maximum average cluster density is merged and the overall number of clusters is reduced by one. This process continues iteratively until there is no pair of clusters that can be merged in order to increase the average cluster density (equation (11)). In this case, the algorithm terminates and the current collection of clusters along with their qualities becomes the output of the algorithm. This iterative process of creating clusters is similar to the hierarchical agglomerative clustering approach [22]. The major differences with agglomerative clustering are two. First, we quantify the "distance" between two clusters as the average pairwise density of all the elements of both clusters. The

// n: total number of event instances
// N_C: current number of clusters
// Q_{C_i}: quality of cluster C_i
// C_{ij}: the cluster produced after merging C_i and C_j
// Q_{global}: current average cluster density
// Q_{matrix}: $N_C \times N_C$ matrix where the entry $Q_{matrix}(i,j)$ is equal to the change of Q_{global} when merging clusters C_i and C_j

// Initialization
$N_C = n$, $Q_{C_i} = \frac{1}{n}$, $i = 1, \ldots, N_C$

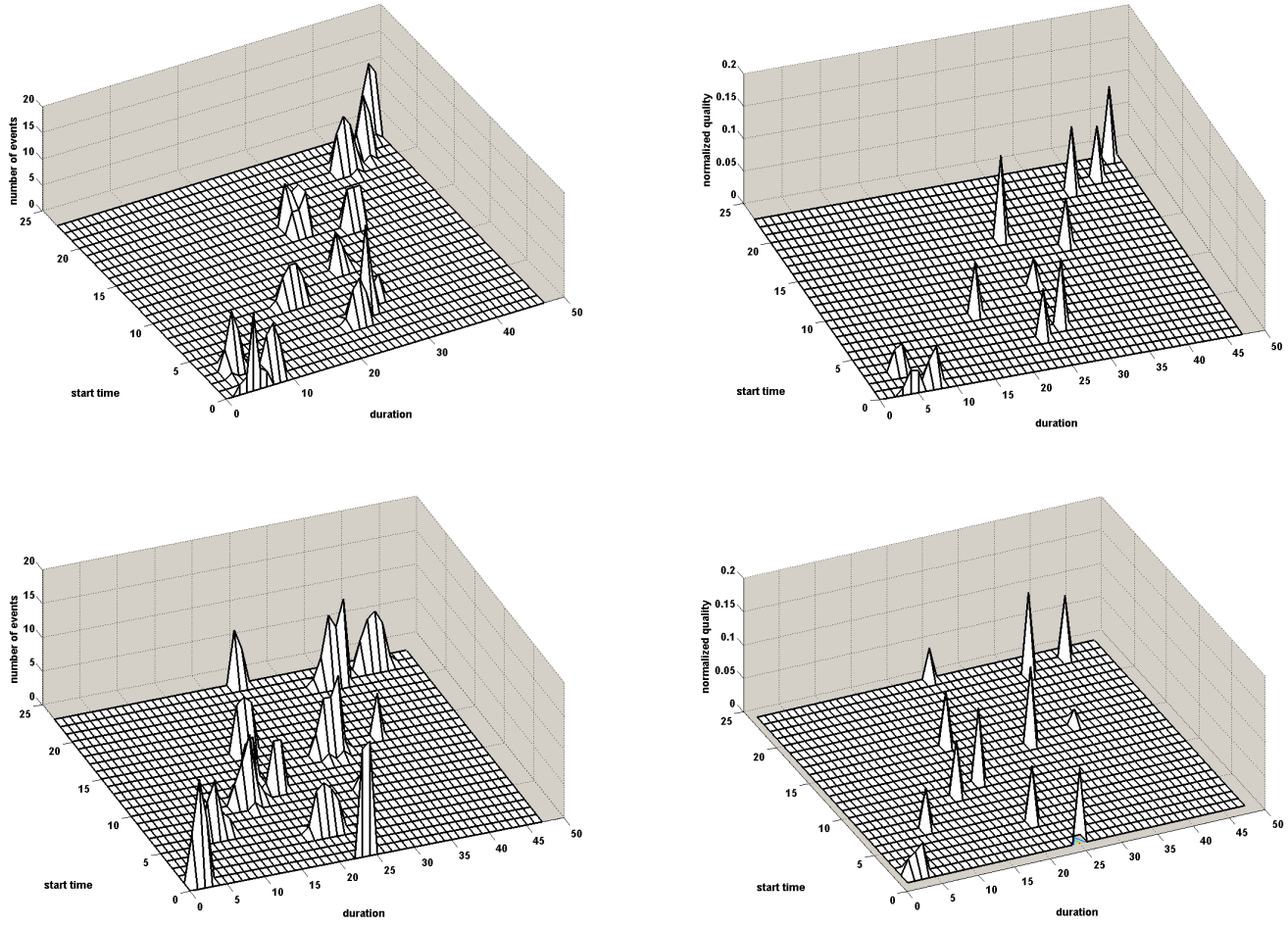
while(1)
{
  for i=1:1:(N_C-1)
  {
    for j=(i+1):1:N_C
    {
      $C_{ij} = C_i \cup C_j$
      $Q_{matrix}(i,j) = Q_{global} - \frac{1}{(N_C-1)+n} \times (Q_{C_1} + \ldots + Q_{C_{i-1}} + Q_{C_{ij}} + Q_{C_{j+1}} + \ldots + Q_{C_{N_C}})$
    }
  }
  if $Q_{matrix}(i,j) < 0$, $\forall i,j$ {**quit**}
  else
  {
    if $Q_{matrix}(i,j) = \max_{i,j} Q_{matrix}(i,j)$ $\{C_i \cup C_j\}$
    $N_C = N_C - 1$;
    update(Q_{global})
    if $N_C == 1$ { **quit** }
  }
}
// Output: A collection of N_C clusters with qualities $Q_{C_1}, \ldots, Q_{C_{N_C}}$

Figure 5: The proposed algorithm for discovering the frequent temporal characteristics of a collection of event instances.

higher this density the closer the two clusters are. Second, the goal of the iterations is to maximize equation 11. It is equation 11 that determines which clusters to be merged and when the algorithm actually terminates.

Every produced cluster can be seen as a specific class of event instances with unique temporal characteristics. The average start time and duration attributes along with their standard deviations can uniquely represent the temporal properties of each cluster. Note, however, that these temporal characteristics do not necessarily represent *frequent temporal characteristics. This is defined by the individual quality of each cluster. The higher the quality of a cluster the more frequent the temporal properties of this cluster are.* In other words, if the input data does not have a frequent temporal structure then this could be easily identified in the output of our algorithm by the quality metric that is associated to each cluster. This property is extremely important since the data the algorithm operates on is real, noisy data that might contain only a few, or even worse, non-periodic temporal properties. *In both cases, the proposed algorithm does not only produce an output; it also provides a confidence of how good this output is given the specific input data.*

As it can be seen in Figure 5, every pair of clusters has to be considered for merging at each step. This corresponds to examining n^2 possible cluster mergings. In addition, only



(a) (b)

Figure 6: (a)Distribution of input line segments based on their start time and duration characteristics. The z axis corresponds to the number of line segments. (b) Distribution of the normalized quality of the produced clusters. The z axis corresponds to normalized quality. In all plots, the x-axis represents start time window and the y-axis duration window. Each start time window corresponds to 60 minutes and each duration window corresponds to 30 minutes.

two clusters can be merged into a single new cluster at each step. Therefore, in the worst case, we end up with a single cluster after $n-1$ steps, resulting to an overall time complexity of $\Theta(n^3)$. This complexity can be further reduced to $\Theta(n^2 \log n)$ by using dynamic programming techniques.

6. EVALUATION

We evaluate the proposed clustering algorithm in two phases. First, since it is prohibitively expensive to have ground truth data in an assisted living setup, we verify the properties of our algorithm using a synthetic dataset that allows controlled inputs with predictable outputs. The goal of these experiments is to validate the properties of our approach and show its ability to accurately discover the temporal structure of any input data stream. Second, we demonstrate the proposed temporal event classification methodology on a data trace collected by a multimodal wireless sensor network deployed in the house of an elder person living alone for a period of 30 days. Our approach is used to extract the daily temporal structure of the different activities performed by the elder person. To demonstrate the value of the information extracted by our algorithm, we use the discovered temporal information to build the fine-grained spatiotemporal activity model of the monitored person.

6.1 Verifying the Properties of the Clustering Algorithm

The goal of the proposed algorithm is to discover the frequent temporal characteristics of the input event instances assuming no prior information about the input events. To demonstrate this functionality we artificially generated several input datasets on which we run the proposed algorithm. The purpose of these experiments was twofold. First, it allowed us to evaluate the performance of our algorithm on randomly generated data for which no prior information is available. Second, it gave us enough control over the input to create well-formed scenarios for which the output of the algorithm can be predicted. Being able to predict what the

output of the algorithm should be, allows us to comment on the performance of the proposed algorithm.

The input datasets were generated using a mixture of gaussian, binomial and random probability distributions. Each dataset was generated over 24 start time windows of 60 minutes duration each and 48 duration windows of 30 minutes duration each. Figure 6(a) shows the distribution of line segments over time in two representative input datasets. The flat areas correspond to combinations of start time and durations for which little or no event instances take place across episodes. These areas represent infrequent event instances. The peaks correspond to the combination of time and duration parameters for which a large number of line segments take place across episodes. The higher the peaks the more the line segments. These areas correspond to frequent event instances. As a result, the goal of our algorithm is to automatically discover the combination of start time and duration parameters that correspond to the peaks in Figures 6(a).

Figures 6(b) show the output generated by the proposed algorithm. For each generated cluster, we compute the average start time and duration of all the event instances in this cluster. Then, we plot the normalized quality of each cluster at the time and duration windows that corresponds to cluster's average start time and duration. By simply comparing Figures 6(a) and 6(b), it becomes clear that the normalize quality peaks in Figures 6(b) perfectly match the frequency peaks in Figures 6(a). This shows that the cluster quality metrics described in Section 5 can successfully capture the frequently repeating temporal characteristics of the input event instances. A frequency peak in Figures 6(a) is always corresponding to an analogous normalized cluster quality peak in Figures 6(b).

More information about the properties of the cluster quality metrics used can be extracted by comparing the relative height of the peaks in Figures 6(a) with the relative height of the corresponding peaks in Figures 6(b). Note that in certain cases there are obvious differences between Figures 6(a) and 6(b) in the sense that tall peaks in Figures 6(a) might correspond to significantly shorter peaks in Figures 6(b). The reason is that the value of the cluster quality metric defined in Section 5 does not only depend on the frequency of event instances. It also depends on how similar the frequently repeating event instances are in terms of their start time and duration characteristics. When these characteristics vary significantly across the event instances in the cluster, then the value of the cluster quality is reduced even if a large number of event instances is included in the cluster.

From these observations, it becomes apparent that the output of the algorithm (Figures 6(b)) manages to accurately capture two very important properties of the input: the frequency of repeating line segments and their similarity in terms of start time and duration across episodes. Both of these properties are properly quantified in the form of the normalized quality of the produced clusters. Note that the normalized quality used is a relative metric across the different clusters discovered. We use such a metric because there is not an absolute/ideal output that our algorithm should provide independently of its input. The role of the output is to match the distribution of the input event instances in terms of frequency and start time and duration similarity. When a uniform distribution of event instances is given across episodes, a uniform output distribution of normalized cluster quality is also expected. When more complex distributions are given (Figures 6(a)), then similar complex distributions of the normalized cluster quality are expected as output (Figures 6(b)).

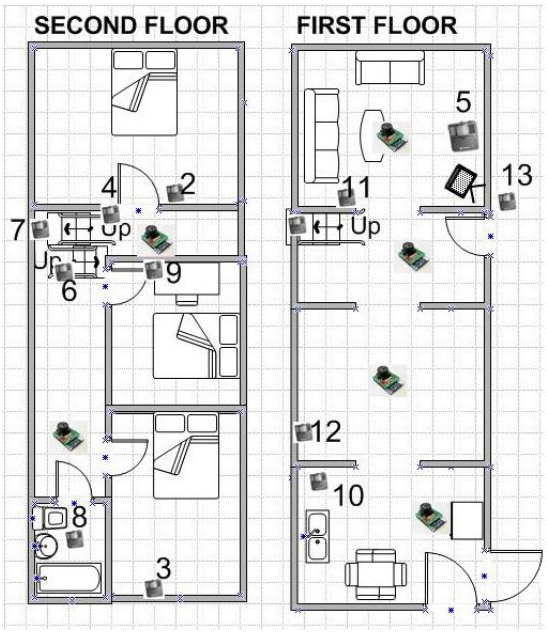


Figure 7: Placement of the camera and motion iMote2 sensor nodes on the deployment site.

6.2 Home Monitoring Testbed and Dataset

In this section we demonstrate the application of the proposed algorithm on a real dataset acquired by one of our home sensor network deployments shown in Figure 8. The network was used to continuously monitor an elder person living alone in his house over a period of 30 days. A combination of low-power camera and motion sensors coupled to the iMote2 sensor nodes were used to timestamp the different rooms and locations that the monitored person visited over time during the 30-day period. Figure 7 shows a 7-day window of the recorded data.

The sequence of timestamped locations, recorded by our sensor network deployment, was first manually filtered to remove any false positives/negatives sensor readings and then mapped to simple primitive activities that the monitored person engages during the course of a day. In this section we will present the results of running the proposed algorithm on three of the most representative and intuitive activities, namely the "Sleep", "Hangout" and "Out" activities. "Sleep" activity corresponds to the person actually sleeping in his bedroom. "Hangout" activity corresponds to the person spending time in the living room (mainly watching TV) and "Out" activity corresponds to the person going out of the house for any possible reason. The detections were made using the *sensory grammars* framework described in [11].

Figures 9,10,11 show the output of the proposed algorithm when run on all the recorded event instances of the sleep, hangout and out activities. The different clusters on each plot are displayed with different colors. In the case of Figure 9, the different clusters produced provide the temporal signature of the sleep activity instances. The cluster with the most activity instances (black line segments) represents

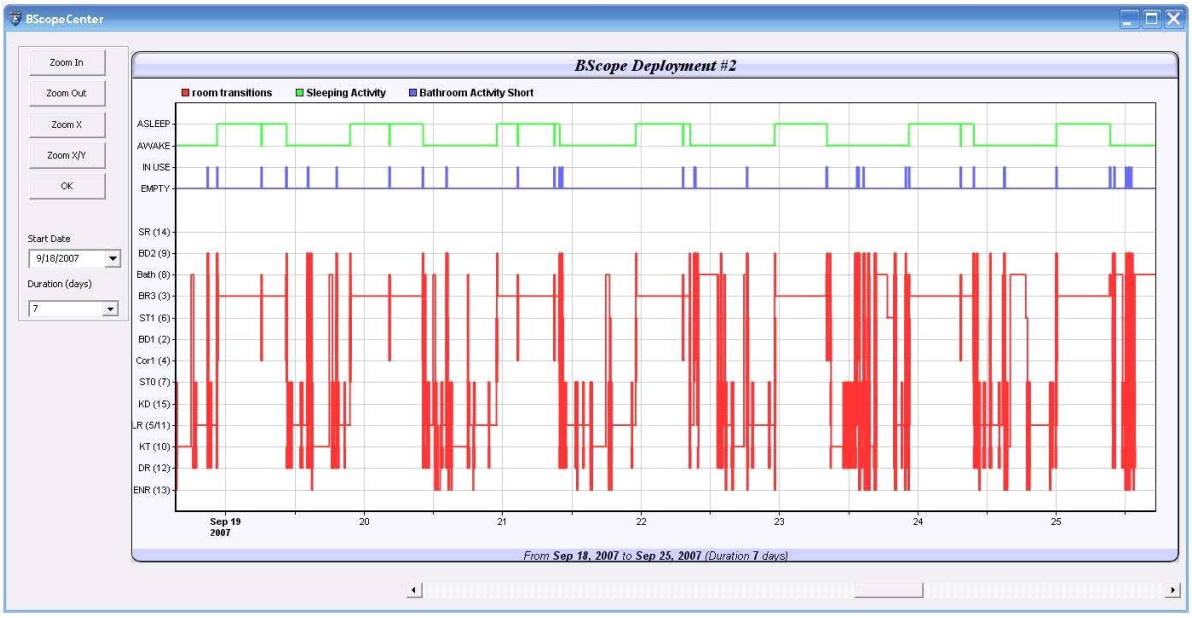


Figure 8: Room transitions and detection of sleeping activity and bathroom usage (using rule based triggers) of the monitored person for a period of one week.

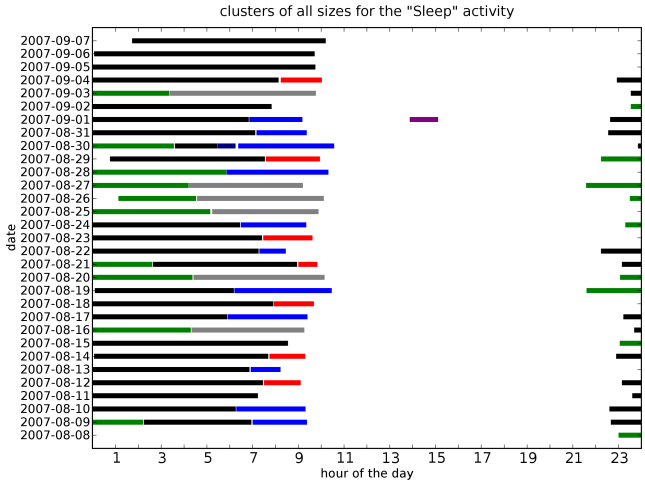


Figure 9: The different *sleep activity* instances of the monitored elder person. The y-axis shows the date when the activity instance was recorded and the x-axis shows the ground truth time. The different colored lines represent the different clusters.

the most frequent sleeping pattern of the monitored person. This corresponds to a continuous sleep from late night up to early morning. At that time the sleep is usually interrupted by an early morning bathroom visit and then the person goes back to sleep for approximately 2 hours (blue and red line segments). However, this does not fully describe the sleeping pattern of the monitored person. It turns out that in more than 33% of the sleep activity instances, the person will wake up in the middle of the night to visit the bathroom. This can be seen from the cluster containing all the green line segments in Figure 9. This bathroom visit is always followed by another sleep activity instance until 9-10am, right before the person wakes up (gray line segments). These different clusters represent the unique temporal characteristics of the monitored person's daily sleep activity pattern.

In exactly the same way, our algorithm can be used to extract the temporal structure of the hangout and out activities. In the case of hangout activity (Figure 10) there are three main time windows. The first but least frequent window takes place right after the person wakes up (red line segments). The remaining two frequent windows take place from approximately 10am to 3pm (black line segments) and from 6pm to approximately 11pm (blue line segments). Usually, between 3pm and 6pm the monitored person gets out of the house and this can be clearly seen in Figure 11.

By simply inspecting the clusters produced by our algorithm in Figures 9,10,11, the temporal structure of all three activities arises. This temporal structure can be clearly seen in Figure 12. For every non trivial cluster (i.e. clusters that contain more than one element) we compute the average start time and duration along with their standard deviations and we plot them with respect to the fraction of line segments that belong to the cluster (Figure 12(a)) and with respect to the normalized quality of the cluster (Figure 12(b)). Using the plots shown in (Figure 12(a)) we can extract the list of start time and durations along with their deviations that a specific activity takes place over the course of a day. How frequent these temporal characteristics are, depends on the fraction of the total number of activity instances that contribute to the calculation of these characteristics. The higher the number of activity instances in a cluster, the more frequent the cluster's temporal characteristics are. For instance, in the case of sleep activity in Figure 12(a), we can easily see that the most frequent sleep activity instances are the ones that start around 11pm and last until around 7am where the monitored person will visit the bathroom. In the same Figure it can also be easily seen that the most frequent out activity instances start around

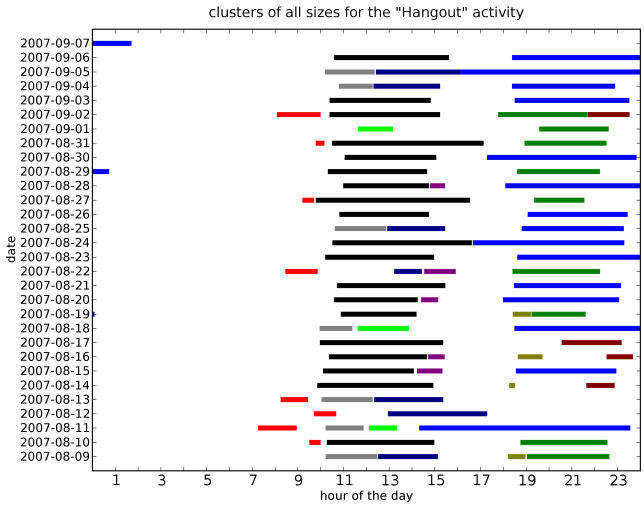


Figure 10: The different *hangout activity* instances of the monitored elder person. The y-axis shows the date when the activity instance was recorded and the x-axis shows the ground truth time. The different colored lines represent the different clusters.

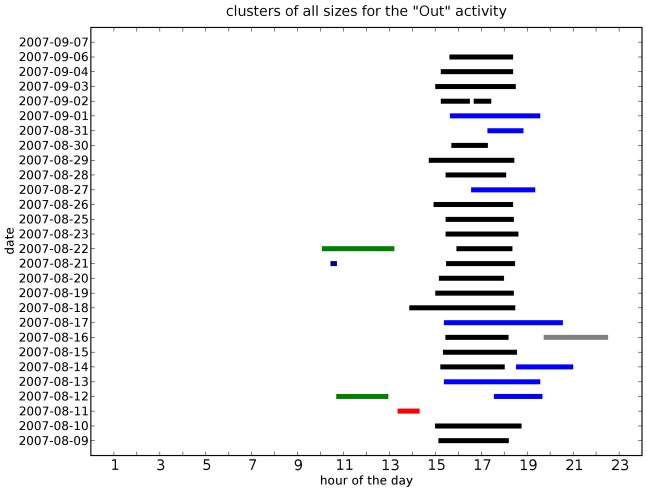


Figure 11: The different *out activity* instances of the monitored elder person. The y-axis shows the date when the activity instance was recorded and the x-axis shows the ground truth time. The different colored lines represent the different clusters.

3pm and last for approximately 3 hours every day when the person goes to work. A more detailed list of the discovered start time and duration pairs along with their normalized qualities for different activities can be seen in Table 1.

Figure 12(b) shows that the proposed cluster quality metric shown in equation 10 can successfully capture both the frequency of a cluster as well as its average pairwise density. In Figure 12(b) the y-axis represents the normalized cluster quality instead of the fraction of activity instances contained in the cluster (Figure 12(a)). It turns out that in all of the three different activities studied, the ordering of the clusters, in terms of how frequent these clusters are, might change; especially in the case of the less frequent clusters. This is due to the fact that the cluster quality metric in equation 10 takes into account the number of cluster members as well as their average pairwise density. In other words, when a cluster contains a lot of activity instances but its average pairwise density is low, then its frequency will not necessarily be high. In that way, the proposed algorithm does not just generate frequent clusters; it generates *dense frequent clusters*.

6.2.1 *Using the Discovered Temporal Characteristics*

The frequent temporal characteristics identified by our algorithm, like the ones shown in Figure 12, can now be used to properly encode time information in the input event instances. For instance, according to Figure 9 we know that the monitored person engages into sleep activities during the night (around 3-4am), in the morning (after 7am) and of course late evening (around 10-11pm). In the same sense, from Figure 11 we know that the person gets out of the house in the afternoon around 3-4pm and comes back around 6pm and from Figure 10 we know that right before he gets out of the house and right after he comes back to the house, the monitored person spends his time in the living room watching TV. This time information can be used to condition every single event instance based on its start time and duration properties. In that way, we can move from spatial events with continuous information such as "sleeping" to spatiotemporal events such as "sleeping during the night", "sleeping in the morning", "getting out of the house in the afternoon" and "watching TV in the morning" where the time definition of "night","morning" and "afternoon" is determined for every event type ("sleep", "hangout" etc.) by the clusters discovered by the proposed algorithm.

The value of automatically discretizing time information into spatiotemporal events comes from the fact that the spatiotemporal events can now be easily processed by most available data mining algorithms [8, 1, 2, 17, 5, 16, 14, 13, 5, 16, 3, 6, 14, 15] to discover correlations across different event

Activity Cluster	Average Start Time	Average Duration (minutes)	Norm. Quality
Early Sleep	$11:00:00PM$	301	0.17
Regular Sleep	$11:54:37PM$	458	0.39
Early Morning Sleep	$06:34:38AM$	172	0.19
Morning Sleep	$07:55:09AM$	104	0.12
Night Sleep	$04:21:30AM$	322	0.1
Regular Out	$03:16:57PM$	174	0.64
Late Out	$04:36:26PM$	191	0.21
Morning Hangout	$10:26:03AM$	284	0.23
*Afternoon Hangout*1	$05:50:52PM$	85	0.18
*Afternoon Hangout*2	$06:50:53PM$	36	0.11
*Evening Hangout*2	$09:36:00PM$	103	0.05

Table 1: A subset of the discovered list of start time and durations for different activities.

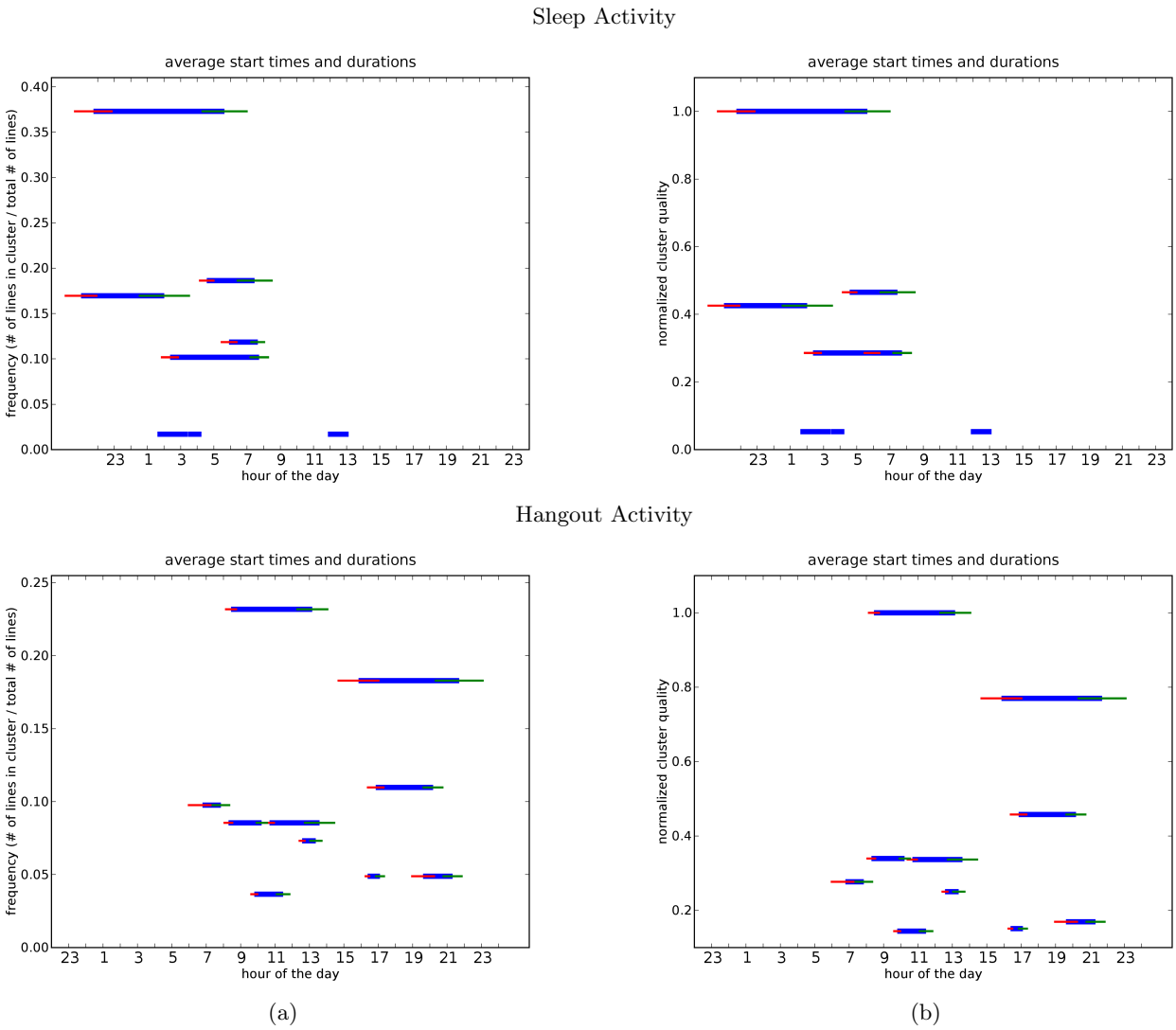


Figure 12: The average start time and duration characteristics along with their standard deviations for the different clusters detected in the case of two different activities. (a) The y-axis shows the percentage of line segments included in the cluster. (b) The y-axis shows the normalized cluster quality. In all plots the x-axis shows the ground truth time.

types in the sensor data stream. Discovering the correlations across different event types allows us to build the complete spatiotemporal model of the sensor data stream. For instance, Figure 13(a) shows the spatiotemporal model (generated using the methodology presented in [8]) of the 30-day sensor stream recorded in our home sensor network deployment. Each state in this model represents a spatiotemporal activity and thus encode both space and time information. The links between different states represent the correlation of different activities in the sensor data stream as described in [8]. To better demonstrate the importance of the discovered time information in the model construction, consider Figure 13(b) where the exact same model generated in exactly the same way but without taking into account the activity temporal characteristics is shown. In this case, each activity becomes a state in the model. By comparing the two models in Figures 13(a),(b), it is clear that the amount of information provided by the model where time information is ignored (or unknown) is minimal. What differentiates the quality of information captured by the two models is the temporal structure information, provided by our algorithm, that allows for more fine-grained spatiotemporal modeling of the sensor data stream.

7. RELATED WORK

The different clustering algorithms that have been proposed in the literature can be roughly classified in two different categories [22], namely the *partitioning* and hierarchical algorithms. In the case of *partitioning* algorithms the goal is to organize a collection of events (points in space) into a set of k clusters. Initially, a set of random k clusters is created. Then, iteratively the algorithm refines those k clusters by optimizing an objective function. At each step, every cluster in the partitioning algorithms is represented as single point in space. In the case of $k-means$ algorithms [12] each cluster is represented by its centroid and in the case of

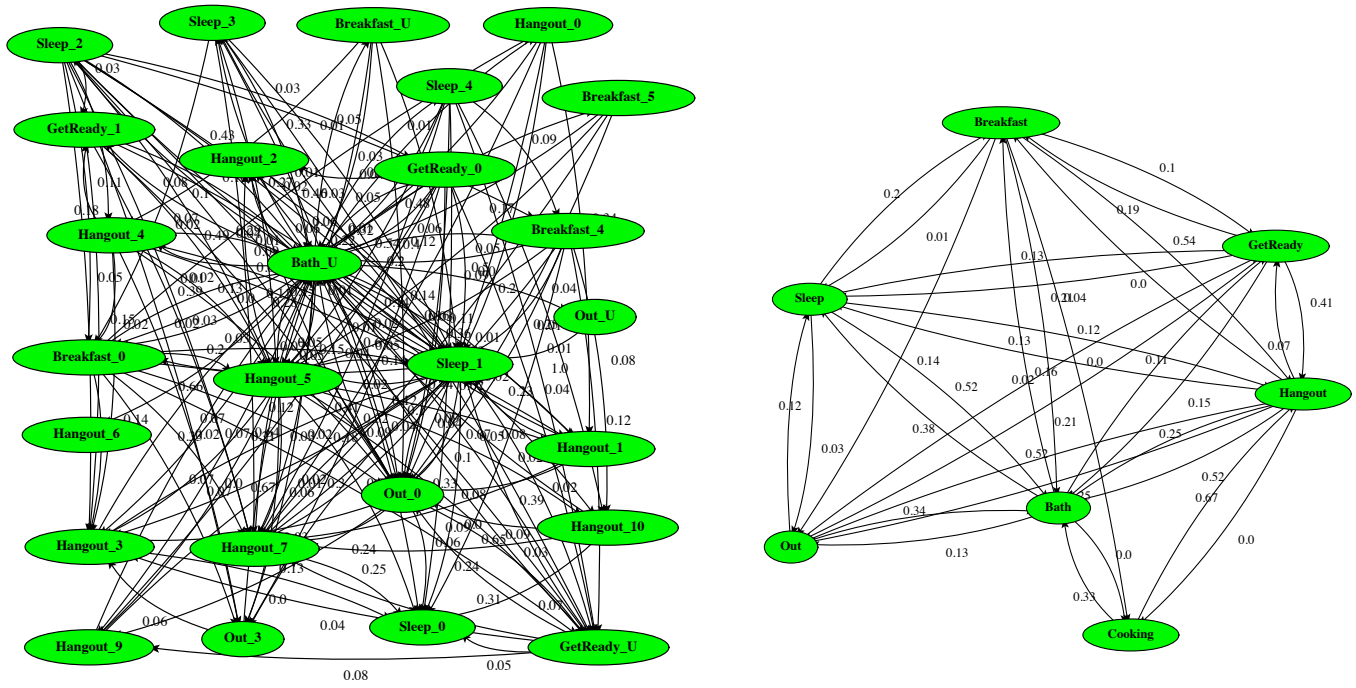


Figure 13: The daily activity model extracted from a recorded data trace of 30 days when (a) both spatial and temporal information is used and (b) only spatial information is used.

$k-medoid$ algorithms [20] each cluster is represented by the event that belongs to that cluster and that is closer to the cluster's centroid. Similar to this type of algorithms is also the expectation maximization (EM) algorithm [4, 7] that has been extensively used in the machine learning and artificial intelligence domains. Even though partitioning algorithms have been widely used in several applications, they are not a good fit for the problem studied in this paper. First, every event is considered as a single point in space, thus ignoring the event's temporal properties. Second, the summarization of a cluster as a single point does not allow capturing all the detailed properties of the cluster. As it was described in Section 5, we can get a more detailed representation of a cluster by considering all of its elements and not just a single one.

The work that is closest to the method proposed in this paper is a hierarchical clustering technique, widely used in the context of Natural Language Processing, called agglomerative clustering [22]. Agglomerative clustering was designed for clustering a collection of vectors instead of a collection of points in space. Initially every vector is represented as a cluster. Then iteratively, at each step the two clusters that are the closest to each other are merged. This process continues either until a single cluster is formed or until there are no two clusters with a distance lower than a predefined threshold distance D_{min} that allows their merging. In the case of the *group-average* agglomerative clustering (GAAC) algorithm, the average distance of all the pairs of vectors that belong in any of the two clusters is used as the distance between the two clusters. Our approach is similar to GAAC in terms of the cluster building process but *it differs in two fundamental ways*:

1. Instead of quantifying the distance between two clusters as the average pairwise distance of all pairs of cluster elements, we introduce the notion of *average pairwise density.*

2. Instead of requiring a threshold distance D_{min} as a stopping criterion, we introduce the notion of a global density metric we wish to optimize. At each step, the clusters that are merged are the ones that maximize the overall average cluster density.

The uniqueness of our approach lies on the way we represent sensor events and on the metrics introduced. These metrics take advantage of the line-segment sensor event representation to better exploit the density of the recorded events across multiple instances and to better extract their temporal properties.

8. CONCLUSIONS

We have presented a data driven algorithm for automatically extracting the temporal characteristics, in terms of start time and duration, across different event instances and for any given event in a sensor data stream. Our approach makes no assumption about the temporal properties to be discovered and could be applied to practically any sensor data stream. The only requirement is a lightweight preprocessing stage for representing the raw data stream as a sequence of spatiotemporal triplets of the form $\{l, t, d\}$.

In this paper, we demonstrated its application on a 30-day dataset collected from one of our home sensor network deployments monitoring an elder person living alone in his house. Currently, our system works reliably when a single

person is monitored. When multiple people are monitored at the same time, a way to separate and assign different data streams to each individual is required. Our recent work has been focusing on this direction [18].

In the immediate future, our plan is to use this in conjunction with the probabilistic grammar framework from [11],[9] to create responsive systems in the home that could actuate and provide services to elders living alone. With these two blocks together, we plan to study the reliability of the system and try to apply it towards providing real-time services.

Our goal is to also apply the developed infrastructure on more diverse mobile datasets outside the concept of assisted living. As an illustration, we are in the process of applying the proposed methodology to GPS measurements from buses to automatically discover and enforce bus schedules as well as to detect where, when and for how long bus drivers stop for breaks.

9. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

[3] J. Coble, D. J. Cook, and L. B. Holder. Structure discovery in sequentially-connected data streams. *International Journal on Artificial Intelligence Tools*, 2006.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[5] E. Heierman, M. Youngblood, and D. J. Cook. Mining temporal sequences to discover interesting patterns. In *KDD Workshop on Mining Temporal and Sequential Data*, 2004.

[6] I. Jonyer, L. B. Holder, and D. J. Cook. Mdl-based context-free graph grammar induction and applications. *International Journal on Artificial Intelligence Tools*, 2004.

[7] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171(5-6):311–331, 2007.

[8] D. Lymberopoulos, A. Bamis, and A. Savvides. Extracting spatiotemporal human activity patterns in assisted living using a home sensor network. In *Proceedings of the International Conference on PErvasive Technologies Related to Assistive Environments (PETRA)*, 2008.

[9] D. Lymberopoulos, A. Ogale, A. Savvides, and Y. Aloimonos. A sensory grammar for inferring behaviors in sensor networks. In *Proceedings of Information Processing in Sensor Networks (IPSN)*, April 2006.

[10] D. Lymberopoulos, T. Teixeira, and A. Savvides. Detecting patterns for assisted living using sensor networks. In *Proceedings of SensorComm*, October 2007.

[11] D. Lymberopoulos, T. Teixeira, and A. Savvides. Macroscopic human behavior interpretation using distributed imagers and other sensors. In *to appear in Proceedings of IEEE*, 2008.

[12] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[13] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[14] J. R. Nayak and D. J. Cook. Approximate association rule mining. In *Proceedings of FLAIRS Conference*, 2001.

[15] J. C. G. Ramirez, D. J. Cook, L. L. Peterson, and D. M. Peterson. An event set approach to sequence discovery in medical data. *Intell. Data Anal.*, 2000.

[16] K. Romer. Distributed mining of spatio-temporal event patterns in sensor networks. In *EAWMS / DCOSS*, 2006.

[17] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

[18] T. Teixeira and A. Savvides. Lightweight people counting and localizing in indoor spaces using camera sensor nodes. In *ACM/IEEE International Conference on Distributed Smart Cameras*, September 2007.

[19] The BScope project. http://bscope.eng.yale.edu.

[20] H. D. Vinod. Integer programming and the theory of grouping. *Journal of the American Statistical Association*, 64(326):506–519, June 1969.

[21] A. Yu, A. Bamis, D. Lymberopoulos, T. Teixeira, and A. Savvides, editors. *Proceedings of the International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense08)*. Raleigh, North Carolina, Nov 2008.

[22] Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical clustering algorithms for document datasets. *Data Min. Knowl. Discov.*, 10(2):141–168, 2005.