

# A Framework for Distributed Activity Recognition in Ubiquitous Systems

Petteri Nurmi, Patrik Floréen, Michael Przybiski and Greger Lindén  
Helsinki Institute for Information Technology HIIT, Basic Research Unit  
Department of Computer Science, University of Helsinki  
P.O. Box 64, FI-00014 Helsinki, Finland  
{firstname.lastname}@cs.helsinki.fi

**Abstract**—The recognition of human activity has many important applications that rely on linking observed behaviour with particular actions. However, activity recognition systems are usually build for specific applications, and the used architectures and solutions are often not applicable in other problem domains. In contrast, we propose a generic component-based framework for activity recognition. The use of components allows the architecture to be potentially distributed over the network and is thus suitable for a wide class of environments. We also demonstrate the workings of our framework by using it to recognize walking and travelling by metro from 3D accelerometer data.

**Keywords**—AI architectures, ubiquitous computing, adaptive systems, distributed AI.

## I. INTRODUCTION

Performing automatic recognition of human activity is critical for various compelling applications such as adaptive multimodal interfaces, proactive service provisioning, and visual surveillance. In general these systems use contextual information to adapt their behaviour, but as, according to Dey and Abowd context is *any* information that can be used to characterize the situation of an entity [1], the potential amount of contextual information is huge, and in order to provide practical applications the used information needs to be carefully selected. Currently the most commonly used information sources are identity and location, but recently interest in using also other sources of information, especially user activity, has risen.

The reason why activity information plays such an important role in various applications is that causal models allow linking observed behaviour with the selection of particular actions (see e.g. [2]). For example, in automated visual surveillance the information that

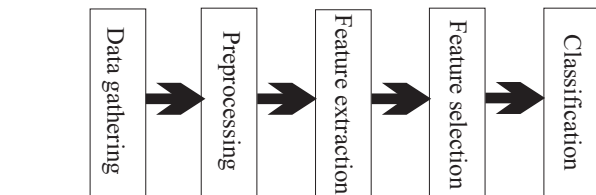


Fig. 1. The phases of the activity recognition process.

somebody is trying to break into a house can be used to automatically trigger a police alarm.

In contemporary systems the recognition of activities and the causal models are typically hardcoded into the system. However, in order to build more generic systems, the possibility to reuse recognition information is required. Furthermore, especially in ubiquitous systems, the devices that perform the activity recognition have limited computational capabilities, and thus large-scale provision of activity information requires the distribution of processing. In this paper we introduce a framework that has been designed to meet these challenges, i.e., allows performing activity recognition in a distributed manner.

Activity recognition in ubiquitous systems is essentially a data analysis task where data arrives in an online manner. The overall process is illustrated in Fig. 1.

In the first phase, *data gathering*, sensors measure characteristics of their environment and communicate the measurements to an entity that further processes the signals. In the second phase, the processing entity performs preprocessing on the data. The various possible preprocessing tasks include, e.g., removing outliers, smoothing the values, replacing missing values, and performing transformations on the data [3], [4].

After preprocessing techniques have been applied to the data, the next step is to derive features from the measurements using statistical and signal processing

techniques. The algorithms that are used in this phase are typically aggregate by nature, i.e., they produce a single value, such as the mean or variance, from a set of measurements. The reason why aggregate features are preferred is that many sensing devices generate huge amounts of data, and in order to reduce communication and computational overhead, the amount of measurements needs to be reduced. If the sensor measurements are produced by multiple data sources, also features that attempt to capture some aspects of similarity between the sources can be derived.

The potential number of features that can be used is numerous. However, it is often the case that only a small subset of them is relevant and to be able to perform the activity recognition in real-time the used features need to be carefully selected. The range of different techniques is large and a thorough discussion about the different methods is out of scope for the paper.

The final phase of the analysis consists of classifying the signals to match a particular activity. Also the different classification techniques are numerous and often the best technique depends on the application. Thus a detailed discussion about this phase is out of scope for the paper.

The organization of the paper is as follows. Section II discusses related work. The actual framework is presented in Section III, whereas Section IV presents an example use case for the proposed framework. Finally, Section V concludes the paper and provides directions for future work.

## II. RELATED WORK

### A. Activity Recognition

Most of the work on activity recognition has focused on either identifying single activities in a particular scenario (a classification task), or on analyzing sequences of activities (a time-series analysis task). According to [5], activity recognition has a history that goes back to at least the 1870s, when animal locomotion was studied. In particular, there is a vast amount of literature in the area of computer vision, where the aim is to determine different types of human activity, mostly motion, from video images; see for instance the survey [5]. Usual modern methods applied include variations of neural networks and hidden Markov models.

An intensive area of research is that of smart spaces (smart homes and offices), with projects such as The Adaptive House at the University of Colorado at Boulder [6], Georgia Tech Aware Home [7], Microsoft's Easy Living [8], and MavHome at University of Texas at

Arlington [9]. In these projects, contextual information is gathered from many different kinds of sensors (touch, audio, video etc.), some of which are placed in particular places in the space, and some with are attached to different objects. In [10] and [11], techniques such as Markov models, naive Bayesian networks, and decision trees are used to select which interface to load on a remote control. In [12], different situations in an office environment are recognized, such as a phone conversation or that nobody is in the office. These situations were recognized using a layered hidden Markov model, which used microphone, camera, keyboard, and mouse information. Our framework is not confined to a particular space.

Yet other relevant areas of research are those of wearable and mobile computing. For instance, in [13], accelerometer, galvanic skin response, and temperature sensors were used to detect behaviour like walking, sitting, waving arms, and climbing stairs. The methods employed in this work are Principal Component Analysis, Kohonen Self-Organizing Maps,  $k$ -means clustering, and first order Markov models. In [14], naive Bayesian classifiers are applied to audio data to recognize the moving behaviour and the kind of music a user is listening to. Our framework is not restricted to one particular user.

Of particular interest to us is the Doppelgänger user modelling framework [15]. The user models of Doppelgänger can be used for personalisation and activity recognition. However, the starting point for our framework is the reasoning process itself.

Generally, "standard" pattern recognition techniques [16] have been applied for activity recognition. Recently also particle filtering has become popular, for example [17] and [18] use particle filtering to infer transportation modes, such as in bus or walking.

Ultimately, in a fully ubiquitous computing environment, the activity recognition task is neither confined to a particular space, nor centred on a particular individual, but the different sensors available on people and in the environment interact within the same framework. Furthermore, the applications should operate using any available contextual information.

### B. Architectures and Frameworks

Different approaches have been taken for providing a common architecture for context-aware applications and for the integration of context-reasoning mechanisms. According to Moran and Dourish [19], current research focuses on either different versions of a blackboard based

approach or on widget-based approaches. Usually these approaches are implemented in the form of middleware and its services, or in the form of application frameworks. Another possibility that Moran and Dourish describe is implementation using interacting agents, which are distributed over a network [4].

Examples of middleware approaches include, e.g., the Reconfigurable Context-Sensitive Middleware (RCSM) [20], and the CORTEX middleware [21]. For our purposes the CORTEX approach is more interesting as it introduces special entities, called *sentient objects*, which are responsible for receiving, processing, and providing context-related information. Sentient objects are defined as autonomous objects that are able to sense their environment and act accordingly [21]. The advantage of this approach is the possibility to re-organize them, for instance depending on their primary task.

Also some application frameworks that support context-awareness have been proposed. For example, [22] describes the implementation of a framework that supports management of context information on mobile terminals. The structure of this framework is centred on the blackboard paradigm for communication, which is handled by a context manager. Most components that use this framework, including the applications, act as clients for the context management system on the device itself. Other services can potentially run also in a distributed environment.

Another framework approach is the Context Toolkit [23], which separates acquisition and presentation of context information from the application that requires it, by using so-called widgets. The focus of this work lays in the automatic inference of higher-level context information from lower-level sensor data.

Another system, developed in the Hydrogen project [24], describes a three-layered architecture, designed to overcome some existing problems in context-aware mobile systems. The framework consists of the adaptor, management, and application layers. For the communication between the different layers an XML-based protocol is utilized. The aim of the framework is the provision of an architecture that is lightweight, extensible, robust, and which enables the possibility of adding further meta-information to the system.

### III. FRAMEWORK SPECIFICATION

The goal of a framework is to provide a basic structure for interactions between components, and to define interfaces for common functionalities, so that different components can be reused and applications can be easily

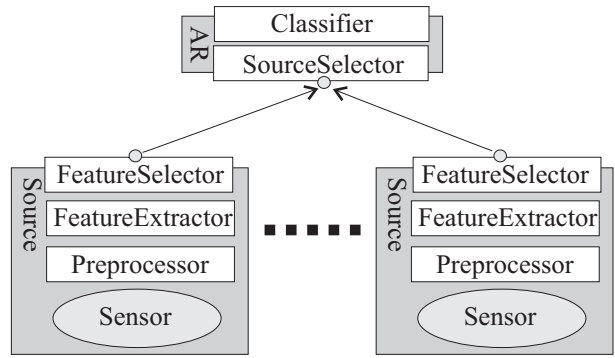


Fig. 2. High-level view of the framework.

created on top of the framework. When designing a framework for activity recognition in ubiquitous systems, additional requirements arising from the environment need to be taken into account. Namely, mobile and hand-held devices often have very limited resources, such as computational capabilities, memory, or energy. Also the network connectivity of such devices is usually limited. Furthermore, the data that arrives from several, rather heterogeneous sources needs to be analysed, processed and provided to applications and services in an online manner.

In this section we present a framework that has been designed to support the tasks required in the activity recognition process and that takes these additional environmental constraints into account. A high-level overview of the proposed framework is given in Fig. 2. Details about privacy, security, and data representation have been omitted and instead the focus is on parts that perform data analysis tasks.

The framework uses a component-based approach. However, instead of having separate components for each phase of the activity recognition process, functionalities have been grouped together so that the framework consists of two kinds of components: *Sources* and *Activity Recognizers (AR)*. The reason for this is that a fully component-based approach would cause too much communication overhead and may result in too complex a system, not feasible for mobile devices.

The Source components are responsible for the first four phases of the activity recognition task and have been designed to be used close to the actual sensing mechanisms. Thus the sources can reside on a mobile, on a handheld, or even on a microelectromechanical sensing (MEMS) device. On the other hand, the Activity Recognizer components have been designed so that they can potentially reside elsewhere in the network. Thus the

environmental constraints require minimizing communication overhead between sources and activity recognizers, and the computational overhead of the sources.

The internal structure of a source is further divided into several modules. Each module implements a single phase of the overall activity recognition process, and thus the tasks of the modules of a source are context gathering, preprocessing, feature extraction, and feature selection. Context gathering is implemented by *Sensor* modules which are essentially wrappers for integrating third-party sensing mechanisms into the framework. In other words, each sensor encapsulates measurements from a single physical source, such as GPS, cell-id, or accelerometer data.

On the other hand, the preprocessing, feature extraction, and feature selection tasks are implemented using container modules that allow attaching and detaching different algorithms dynamically to the framework. The containers are also responsible for calling the algorithms when new data is available (publish/subscribe), or when inferences are needed (request/reply). The *Preprocessor* and *Feature Extractor* containers allow attaching an arbitrary number of algorithms, whereas the *Feature Selector* allows attaching only a single algorithm to the framework (see also Section V).

Each container specifies the interface that the algorithms must implement. However, containers of different sources can use different interfaces, and thus the framework is applicable to different representations of data and to different classes of methods. The advantage of this approach is that individual algorithms become reusable reasoning components that ease the implementation of context-aware applications by abstracting the reasoning tasks from the application design.

The Activity Recognizer components are responsible for performing the final phase of the activity recognition process, namely, deciding in which activities the user (or other entity) is involved. The reason why this task is separated from other tasks is that classification often requires more computational power than the earlier phases. This holds especially, when learning or updating the classifier is needed. Currently the learning of the classifiers is assumed to be performed offline on a server, but later we intend to add support for learning the classifiers. However, the computational constraints of learning have already been taken into account in the design phase. Support for classification is provided through a *Classifier* container, which, as the other containers, allows classification algorithms that satisfy a specific interface to be attached and detached to the framework.

Data exchange between sources and activity recognizers can be performed using either the publish-subscribe model, or the request-reply model. However, in both cases an activity recognizer can receive data from arbitrary many sources. In order to reduce unnecessary communication, the activity recognizer has an additional module, *Source Selector*, which allows using similar techniques as in the feature selection phase to select the most relevant sources. Thus the goal of the source selection is to automatically prioritize sources and to reconfigure subscriptions / registrations between the different components so that only the most relevant data is exchanged between the components.

#### IV. USE CASE: INFERRING ACTIVITIES FROM 3D ACCELEROMETER DATA

In order to illustrate the workings of the framework, we consider the tasks of inferring when a person is walking and when a person is in a metro<sup>1</sup> from 3D accelerometer data, which also includes other activities of the user.

Initially we gathered a set of test data using a sensor box equipped with a 3D accelerometer. The gathering was done by a single test subject according to a specific scenario which included, e.g., walking, travelling with bus, travelling with metro, and going up and down escalators. The sensor box was attached to the hip of the test subject. Overall, the test scenario lasted slightly over half an hour and produced nearly 220 000 samples. Instead of wrapping the actual accelerometer sensor within the framework, we used a software sensor, whose readings were produced by reading an ASCII file that contained the original measurements.

The sampling rate of the accelerometer was 100 Hz, and thus classifying the original measurements in real time was not feasible. For this reason, we configured the source so that each time data were requested by an activity recognizer, the measurements of the last second, i.e. 100 three-dimensional points, were provided. For the feature extraction we used initially typical signal processing and time series related features such as auto-correlation, mean, variance etc. Overall we tested seven different features.

For feature selection we used an online algorithm that is introduced in [25]. The algorithm left us with only two features: variance and absolute magnitude of acceleration. These features are plotted in Fig. 3 and Fig.

<sup>1</sup>Partially underground urban railway system, referred to using different words in different countries: subway, underground etc. In Helsinki it is called metro.

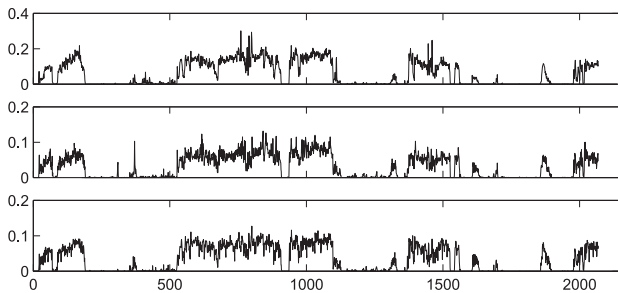


Fig. 3. From top to bottom: the variance of x, y and z dimension of acceleration. Values on the x-axis show the elapsed time in seconds from the beginning of the recording.

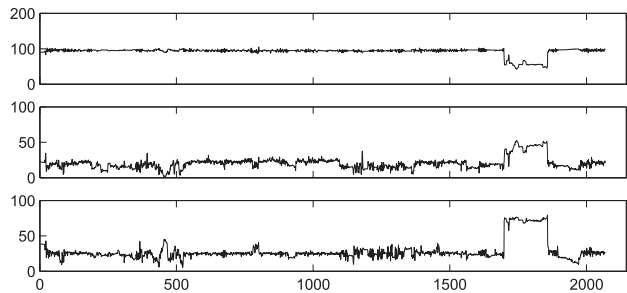


Fig. 4. From top to bottom: absolute magnitude of x, y and z dimension of acceleration. Values on the x-axis show the elapsed time in seconds from the beginning of the recording.

4. Assuming a configuration where the source resides on the terminal device and the activity recognizer on a server, the framework has effectively reduced the communications to 48 bytes (6 doubles) / second.

On top of the accelerometer source we implemented activity recognizers for inferring whether a user is walking or travelling in a metro. As we had only a single source, no source selection algorithms were needed. Classifier design was performed manually by first selecting a subset of the samples and deriving an initial classifier using this set of data. Next another subset of the samples was used to calibrate the classifier and, finally, the entire data set was used to evaluate the performance of the classifier (and thus of the overall activity recognition). The subsets that were used for classifier design and calibration contained each approximately 300 aggregated samples from the overall of nearly 2200 aggregated samples. These subsets included approximately a quarter of the overall data available from the particular activity (walking or in metro) and additionally data from other activities (in bus etc.). At least 50% of the data was from other activities.

For measuring the performance of the classifiers we used classification accuracy and discriminative accuracy, i.e., how well it recognizes the particular activity and how well it separates the particular activity from other activities, respectively. As the labelling of the data was not exact, but only descriptions made by an additional person observing the test subject, precise evaluations could not be made. However, based on the description of activities, both accuracies seemed to be around 95%.

The classifiers that were used to recognize the activities were simple linear classifiers. For recognizing walking, only the variance was needed as periods with high variance in the measurements tend to represent

walking. However, for recognizing the metro travel, we needed to use both variance and magnitude, of which the absolute magnitude was more significant. The classifier for walking used weight values 0.35 for all dimensions of variance and weight 0.0 elsewhere. The bias term was set to  $-0.01$ . An embedding of the data points is illustrated in Fig. 5.

For recognizing that the user is in a metro, the weights of the absolute magnitudes were initially set to 0.10. In the calibration phase we slightly adjusted these (max  $+/- 0.015$ ) to achieve a better overall accuracy. The weights of the variances were set to  $-100.0$  and the bias term was set to  $+0.03$ .

## V. CONCLUSIONS AND FUTURE WORK

In this paper we introduced a framework for distributed activity recognition in ubiquitous environments. In addition, we illustrated the workings of the framework by considering the task of inferring activities from 3D accelerometer data.

Concerning our future work on the framework, we will first of all use the existing framework and implement more feature extraction and inference algorithms that can be used to recognize other activities. Secondly, our goal is to build on top of the activity recognizers an aggregator framework, which allows to fuse multiple classification results, and thus to identify sequences of activities. Furthermore, as the framework is still in a prototype status, more effort will be put on technical details. For example, additional support for prediction of context values and using multiple feature (and source) selection algorithms at the same time are part of future implementations. Also, our goal is to later extend the levels of reasoning so that the activity recognizers can also be used as sources for other activities.

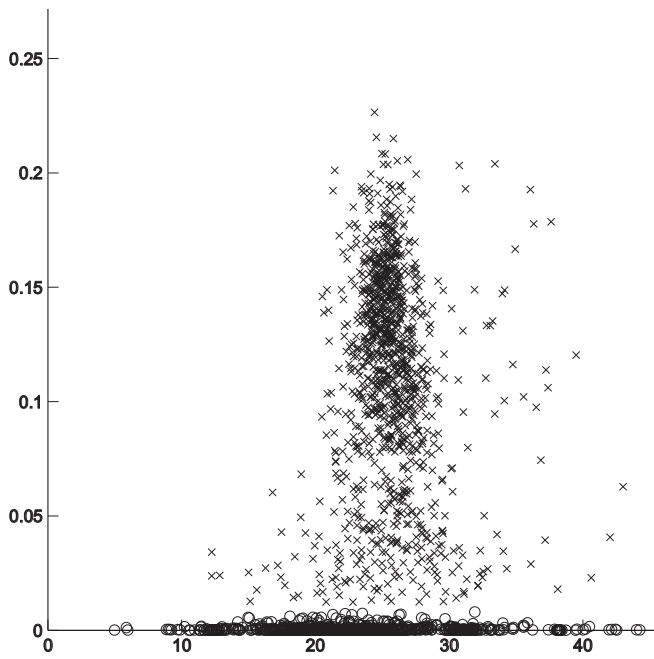


Fig. 5. A two-dimensional embedding of the data points. Crosses represent walking and circles represent other activities. The values on the x-axis represent the mean absolute magnitude of acceleration and the values on the y-axis represent the mean variance of the measurements.

## REFERENCES

- [1] A. Dey and G. Abowd, "Towards a better understanding of context and context-awareness," Georgia Institute of Technology, Tech. Rep. GIT-GVU-99-22, 1999.
- [2] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The Lumière project: Bayesian user modeling for inferring the goals and needs of software users," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1998, pp. 256 – 265.
- [3] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. MIT Press, 2001.
- [4] P. Nurmi, M. Przybilski, G. Lindén, and P. Floréen, "An architecture for distributed agent-based data preprocessing," in *Proceedings of the Workshop on Autonomous Intelligent Systems: Agents and Data Mining (AIS-ADM)*, ser. Lecture Notes in Computer Science, V. Gorodetsky, J.Liu, and V. Skormin, Eds., vol. 3505. Springer-Verlag, 2005, to appear.
- [5] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231–268, 2001.
- [6] "The Adaptive House," <http://www.cs.colorado.edu/~7E MOZER/house/>, Feb. 2005.
- [7] "The Aware Home," <http://www.cc.gatech.edu/fce/ahri/>, Feb. 2005.
- [8] "Easy Living," <http://research.microsoft.com/easyliving/>, Feb. 2005.
- [9] "MavHome, Managing an Adaptive Versatile Home," <http://mavhome.uta.edu/>, Feb. 2005.
- [10] N. Desai, K. Kaowthumrong, J. Lebsack, N. Shah, and R. Han, "Automated selection of remote control user interfaces in pervasive smart spaces," in *Proceedings of HCIC Workshop on Pervasive Smart Spaces*, 2002.
- [11] C. L. Isbell, Jr, O. Omojukon, and J. S. Pierce, "From devices to tasks: automatic task prediction for personalized appliance control," *Personal Ubiquitous Computing*, vol. 8, pp. 146–153, 2004.
- [12] N. Oliver, A. Garg, and E. Horvitz, "Layered representations for recognizing office activity," in *Proceedings of the Fourth IEEE International Conference on Multimodal Interaction (ICMI)*, October 2002, pp. 3 – 8.
- [13] A. Krause, D. P. Siewiorek, A. Smailgaic, and J. Farrington, "Unsupervised dynamic identification of physiological and activity context in wearable computing," in *Proceedings of the Seventh IEEE International Symposium on Wearable Computers (ISWC'03)*, Oct 2003.
- [14] P. Korpipää, M. Koskinen, J. Peltola, S.-M. Mäkelä, and T. Seppänen, "Bayesian approach to sensor-based context awareness," *Personal Ubiquitous Computing*, vol. 7, no. 2, pp. 113–124, 2003.
- [15] J. Orwant, "Heterogeneous learning in the Doppelgänger user modeling system," *User Modeling and User-Adapted Interaction*, vol. 4, no. 2, pp. 107–130, 1995.
- [16] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 2nd ed. Academic Press, 2004.
- [17] D. J. Patterson, L. Liao, D. Fox, and H. A. Kautz, "Inferring high-level behaviour from low-level sensors," in *Ubicomp*, 2003, pp. 73 – 89.
- [18] L. Liao, D. Fox, and H. A. Kautz, "Learning and inferring transportation routines," in *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*, 2004, pp. 348 – 353.
- [19] T. P. Moran and P. Dourish, "Introduction to special issue on context-aware computing," *Human-Computer Interaction (HCI)*, vol. 16, no. 2-3, pp. 87 – 96, 2001.
- [20] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S. K. Gupta, "Re-configurable context-sensitive middleware for pervasive computing," *Pervasive Computing*, vol. 1, no. 3, pp. 33–40, Jul-Sep. 2002.
- [21] H. A. Duran-Limon, G. S. Blair, A. Friday, P. Grace, G. Samartzidis, T. Sivaharan, and M. Wu, "Context-aware middleware for pervasive and ad hoc environments," 2003.
- [22] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keränen, and E.-J. Malm, "Managing context information in mobile devices," *Pervasive Computing*, vol. 2, no. 3, pp. 42 – 51, Jul. - Sep. 2003.
- [23] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Human-Computer Interaction (HCI)*, vol. 16, no. 2,3&4, pp. 97 – 166, 2001.
- [24] T. Hofer, W. Schwinger, M. Pichler, and G. Leonhartsberger, "Context-awareness on mobile devices, the Hydrogen approach," in *Proceedings of the 36th International Hawaii International Conference on System Sciences (HICSS'02)*, 2002.
- [25] P. Nurmi and P. Floréen, "Online feature selection for contextual time series data," Presented in *PASCAL workshop on subspace, latent structure and feature selection techniques: statistical and optimisation perspectives workshop*, Feb. 2005, <http://www.cs.helsinki.fi/u/ptnurmi/papers/slsfs05.pdf>.