Managing Sensor Data Streams in a Smart Home Application

Johan Jansson, Ismo Hakala
University of Jyväskylä / Kokkola University Consortium Chydenius
P.O.Box 567, FI-67701, Kokkola, Finland
Email: firstname.lastname@chydenius.fi

***Abstract*—A challenge in developing an ambient activity recognition system for use in elder care is finding a balance between the sophistication of the system and a cost structure that fits within the budgets of public and private sector healthcare organisations. Much activity recognition research in the context of elder care is based on dense networks of sensors and advanced methods, such as supervised machine learning algorithms. This paper presents the data processing aspects of an activity recognition system based on a simpler, knowledge-based unsupervised approach, designed for a sparse network of sensors. By structuring sensor data management as a streaming system, we provide a simple programming model for the application logic, which facilitates building a fault-tolerant system with the potential for distributed data management within the sensor network. The system, evaluated by a public sector healthcare organisation, constitutes an example of a system that is useful and has a sustainable cost structure.**

***Index Terms*—activity recognition; healthcare; home care; passive infrared sensor; PIR; sensor data; sensor data management; sensor data processing; sensor data streams; smart home; wireless sensor network; WSN.**

TABLE I
HEALTH ISSUES MAPPED TO SYSTEM OUTCOMES

Issue (Klemets et al., 2017)	**Room visit**	**Outside**	**Wandering**	**Room motion**
Raise attention				
Urinary tract infection	bathroom			
Digestion	bathroom			
Depression		X	X	
Wandering behaviour	change*	X	X	
Pain			X	bedroom
Memory impairment	change*	X	X	
Provide insight				
Tiredness and fatigue	bedroom			
Functional ability		X	X	all rooms
Pressure ulcer	X			X
Hypnotic drug effect			X	all rooms
Nutrition	kitchen			

I. INTRODUCTION

The population is ageing in many countries of the world, putting pressure on the healthcare sector (Andueza Robustillo et al., 2015; United Nations, Department of Economic and Social Affairs, Population Division, 2015). To cope with this demographic challenge, different smart home technologies have been proposed as a means of increasing the efficiency of elder care and supporting ageing-in-place (Liu et al., 2016; Rashidi and Mihailidis, 2013). Activity monitoring systems in the homes of elders is a proposed solution with the potential to detect health issues early and assist healthcare professionals in estimating the care needs of seniors (Klemets et al., 2017; Peetoom et al., 2015).

This paper presents the data processing aspects of a smart home system that uses wireless sensor network/IoT-based technology for activity recognition in the homes of elderly people. The system was developed in collaboration with home care nurses to provide activity information that assists them in making decisions about care and nursing resources. During this project, nurses identified seniors who have memory impairment and live alone, as the main target group of the system. Bathroom visits, time spent in the bedroom, wandering behaviour, and time spent outdoors during the night were identified as activities that could help the nurse get a better understanding of how the senior is doing (Klemets et al., 2019). Table I lists issues identified by nurses and related activities that can be recognised by the system. To recognise these activities based on sensor data, a data processing system was developed to refine raw sensor data into information about the activities that can be visualised for nurses in a graphical user interface. The system was deployed for evaluation in 12 different elderly persons' apartments in collaboration with the home care services of the city of Kokkola in Finland.

Wireless sensor networks like the one used in this project inherently provide streams of sensor data that need to be continuously processed, monitored, and responded to based on given conditions. This makes them a good fit for data stream management systems (Golab and Özsu, 2003), where the sensor data can be managed as a continuous data stream, as described by Abadi et al. (2003) and Babcock et al. (2002). Structuring sensor data management as a data stream pipeline brings benefits such as being able to provide a simple programming model for the application logic; thus it is possible to build a more fault-tolerant, distributed, parallelised system by running the same application logic on an advanced streaming framework, as described by Akidau et al. (2013).

The contribution of this paper is a modular system design for a sensor data streaming pipeline in a smart home system designed specifically for the needs of elderly people living

alone. In homes with two or more residents, the other residents may provide nurses with information on daily activities; hence, our focus on single-person homes. The system is described in detail to illustrate the modular design, as well as specified inputs and outputs for each processing stage of the pipeline. This approach facilitates the verification and evaluation of the design, especially the system's correctness and whether it produces information that meets the users' needs. The detailed description also highlights the individual data processing steps' low space and computational complexity, which make it possible to distribute data processing to resource-constrained devices in wireless sensor networks.

II. Background and related work

There are several aspects of activity recognition using sensors that monitor the living environment, as well as processing sensor data streams. Peetoom et al. (2015) present different groups of monitoring technologies and the aim for using those technologies in activity recognition. Rashidi and Mihailidis (2013) summarise aspects of ambient-assisted living tools: technologies, algorithms, applications, design issues, and social and ethical issues. Krishnan and Cook (2014) discuss different approaches to processing streaming data and propose a sliding window-based data processing method. Zhuang et al. (2005) present different sensor data stream engines. In the following section, we examine some aspects that are of interest in this project.

1) Sensor technology: Different kinds of sensor technology are used for active recognition sensors. These can be divided into the following main groups: passive infrared (PIR) motion sensors, wearable sensors, video monitoring, pressure sensors, and sound recognition (Peetoom et al., 2015). Wearable sensors include beacons, accelerometers, gyroscopes, and GPS positioning (Rashidi and Mihailidis, 2013). Interactions with objects, such as dishwashers or phones, can be used for activity recognition (Krishnan and Cook, 2014). The system described in this paper uses a minimal set of PIR sensors and door switches for activity recognition. The system can be extended with other sensors, depending on the needs of the users of the system.

2) Number of sensors: Activity monitoring studies are usually based on dense networks of sensors where a smart home deployment has tens of sensors (Cook, 2012). From an economical and practical standpoint, we chose to deploy a sparse network of sensors in this project, typically with one sensor per room (i.e. 5–6 sensors per apartment in a service home setting). While a dense sensor network can produce enormous amounts of data, there is not necessarily a correlation between the number of sensors and the system's performance. Accuracy can be achieved by carefully positioning a smaller number of sensors (van Kasteren and Kröse, 2007). The system described in this paper is an example of performing activity recognition by deploying a sparse network, carefully positioning the sensors, and evaluating the sensor positions with a test protocol, as described in Section VI.

3) Algorithms: Activity recognition can be performed using machine learning algorithms, such as naive Bayes classifiers, hidden Markov models, and conditional random fields (Cook, 2012). Rashidi and Mihailidis (2013) mention that supervised algorithms are common in ambient sensor systems, but supervised methods might not scale in real-world deployments. Unsupervised activity discovery methods, such as mining for frequent sensor sequences or discontinuous activity patterns, can be used (Krishnan and Cook, 2014). Krishnan and Cook (2014) present a sliding window-based approach on streams of sensor events for activity recognition. Similar to the system presented in this paper, they use binary (i.e. on/off) passive infrared sensors installed in the houses of volunteers. The algorithms can also be divided into a) data-driven methods based on probabilistic and statistical models and b) knowledge-driven methods that utilise prior domain knowledge and formal logic reasoning (Chen et al., 2012).

This paper presents an unsupervised knowledge-driven approach to activity recognition that identifies discontinuous activity patterns. This approach is well suited for the sparse network of sensors used in this project (Chen et al., 2012).

4) Data processing: Wireless sensor networks, such as those used in this project, provide streams of sensor data that need to be continuously processed, monitored, and responded to based on given conditions. This makes them a good fit for data stream management systems (Golab and Özsu, 2003), where sensor data can be managed as a continuous data stream (Abadi et al., 2003; Babcock et al., 2002).

Three common approaches to processing sensor data streams are explicit segmentation, time-based windowing, and sensor event-based windowing (Krishnan and Cook, 2014). In this project, we use the time-based windowing approach, as described in later sections.

Structuring sensor data management as a streaming system has benefits, such as being able to provide a simple programming model for the application logic; thus it is possible to build a more fault-tolerant, distributed, parallelised system by running the same application logic on an advanced streaming framework, as described by Akidau et al. (2013). This paper describes the sensor data management of this project as an example of a data streaming system.

5) Data stream management in sensor networks: Zhuang et al. (2005) divide WSN data stream engines into (1) workflow-based, such as Aurora (Abadi et al., 2003), Borealis (Abadi et al., 2005), and TelegraphCQ (Chandrasekaran et al., 2003); (2) relation-based, such as STREAM, as described by Motwani et al. (2003), and TinyDB, as described by Madden et al. (2005); and (3) object-based, such as Cougar, as described by Demers et al. (2003). These papers describe frameworks for query processing of sensor data streams, and many of them provide SQL-like query languages. The system presented in this paper is a data stream processing pipeline constructed as a directed acyclic graph of processing steps. This pipeline is built around a single primitive: a data processing step that processes tuples (called sensor samples or data intervals in this paper), outputting zero or more tuples for each

input tuple. This resembles the tuple processing structure of the Aurora model presented by Abadi et al. (2003). Our system does not use a query language, but common query operations, such as map, filter, and time-windows, are implemented in low-level steps of the processing pipeline. Diallo et al. (2012) present a survey of real-time data management in WSNs, where most of the presented solutions are based on simulation results. In contrast, this paper presents a system with actual wireless sensor networks deployed to 12 apartments of elderly persons and used by nurses in their day job.

The contribution of this paper is a **low complexity approach** to performing the data processing for a smart home activity recognition system for elderly people living alone, **deployed to 12 service home, single-person apartments and used by nurses in the provision of home care**. The chosen sensor technology, number of sensors, algorithms, and data processing and streaming pipeline all serve the purpose of building a system with low complexity.

III. Requirements and design challenges

The activity recognition system is based on a wireless sensor network. The use of a wireless sensor network adds the following requirements to the system:

Hardware/physical requirements

1) Sensor selection: To recognise the activities listed by the nurses while maintaining the clients' privacy, the system needs to locate and track the client in the apartment in a non-intrusive way. This can be achieved using PIR motion sensors and front-door mechanical switches. Data from these sensors need to be integrated to acquire information on the location and activity of the client.

2) Energy efficiency: High energy efficiency is a requirement for the sensor network to keep system maintenance costs at a sustainable level. To keep energy efficiency at a sufficient level, the battery-powered sensors need to operate in a duty cycle where the sensor node sleeps most of the time, as described in Section IV.

3) Number of sensor nodes: By limiting the number of sensors, we can keep installation, hardware, and maintenance costs low. The data processing algorithm needs to be designed to work for a relatively small number of sensors per apartment, also taking sensors' duty cycle into account.

Data processing requirements

4) Data clean-up: The raw data produced by the sensor nodes will need to be cleaned up prior to further processing. There might be clock drift between sensor nodes in the designed system, and this drift needs to be taken into account by synchronising time of different sensors. Additionally the motion sensors might be positioned so that multiple sensors sense motion in the same room, or one sensor senses motion in multiple rooms. The designed system handles these room overlaps and cleans the raw data into room-specific data.

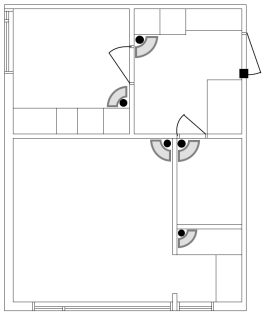


Fig. 1. Example sensor layout in an apartment. Black dots depict PIR sensors, and the grey arcs indicate sensing direction. The black square depicts a door sensor.

5) Data integration: The designed system needs to integrate motion data from different sensors in the apartment to generate information about the client's location. To identify motion patterns involving multiple sensors, data aggregation can not be performed on the sensor node level. Thus, data from individual sensors must be integrated on network level, such as in the network sink, gateway, or server.

6) Missing data: There might be time windows when no data are received from one or more sensors, for example due to wireless interference or a low battery level. This creates a challenge for designing sensor synchronisation and data processing in a robust way that can compensate for occasional missing sensor data. Since the nurses' need for data is on a hourly or daily level, missing a few minutes of data has very small impact on the end result. However, the data processing needs to cope with this situation in a robust way.

7) Activities: As described in Table I, the following kinds of activities were identified to be of importance to the nurses: time spent in different rooms, time spent outside, wandering behaviour, and motion in different rooms.

8) Detect if multiple persons are present: The system was designed based on the assumption that only one person is present in the apartment, as the apartments participating in this project are single-person.

IV. System architecture

The system described in this paper is used to monitor the activity of home care clients in their homes in a non-intrusive way. In each monitored apartment, there is a wireless sensor network (WSN) deployed to monitor the activity of the resident using motion sensors — one in each room — and reed switches on the apartment's front door. The WSNs send the gathered sensor data to a server. The data are then processed on the server and presented in a web-based data-visualisation user interface for the home care personnel, the resident, and close relatives with the resident's consent. The WSNs are built on the CiNet platform. The CiNet wireless sensor nodes use the Atmel ATZB-24-B0 2.4GHz ZigBit module, which integrates an ATmega1281 microcontroller and an AT86RF230 RF transceiver (Atmel, 2013). The CiNet sensor nodes used in the SmartHome4E project are equipped

with either a Panasonic EKMB1103111 pyroelectric infrared (PIR) motion sensor or a AMS-38 Mechanical Surface Mount Contact reed switch.

In each monitored apartment, the WSN nodes are arranged in a star topology (i.e. all sensor nodes communicate directly with the sink node), using the IEEE 802.15.4 wireless communication standard. The sink node is connected via a serial port to the gateway, a small single-board Raspberry Pi computer running Linux. The gateway uses the internet connection of the apartment to forward the data from the sink to the server over HTTP.

In the monitored apartments, we installed a reed switch sensor node to detect the opening and closing of the apartment's front door. These nodes send the door event data ('open' and 'close') in real-time to the sink node. In each room we installed a motion sensor node. Each motion sensor node runs the following duty cycle: the PIR sensor senses motion continuously and sends interrupts to the microcontroller when it detects motion. The microcontroller stores a one-bit sample every 5 seconds (0 = no motion, 1 = motion), i.e. the microcontroller wakes up only for the first PIR interrupt of each 5 second sample. Every 10 minutes, the motion sensor node sends a bit array $(0, 1, 0, \ldots 0)$ with 10 minutes of data (120 samples) to the sink node.

The motion sensor sample size of 5 seconds is a trade-off between battery life, and the granularity of the collected data. A shorter sample size would provide more detailed data but imply shorter battery life as the sensor duty cycle would increase. Since we are designing a system that needs to have low maintenance cost (i.e., the labour cost of exchanging the sensor's battery is high relative to the hardware cost), and we are able to identify resident activity at this sample size, we chose the sample size to be 5 seconds in this project.

Each motion sensor in the system produces, in theory, 17 568 samples per day. In the deployments where we tested the system, the sensors of a single apartment produced, on average, 79 745 samples per day, and the system as a whole produced an average of 675 228 sensor data samples per day. For example, to inspect one month's activity for an apartment, the system needs to process the equivalent of 2.4 million raw sensor samples. This amount of data puts emphasis on the performance of data processing, as well as the compression/aggregation of data.

V. Data processing algorithm

Below, we describe the data analysis process for a single apartment, occupied by an elderly resident living alone. The data analysis process is described as a pipeline where sensor data is streamed through different data processing stages. Each stage refines the data to a higher abstraction level (see Figure 2). The first two stages—sensor node and sensor network—are typical for WSNs, processing data on the sensor level and integrating different sensors' data into sensor network-level data. The room motion stage performs data processing that is more specific for this application; sensor data is refined to room-specific motion. The location stage checks whether more than one person is in the apartment. If only one person is present, it classifies the person's location. The final stage performs the actual activity recognition by identifying wandering, time spent in different rooms, and time spent outside the apartment.

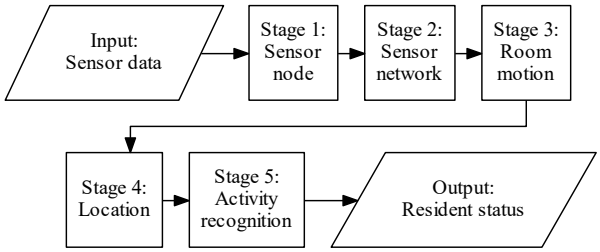


Fig. 2. Data analysis pipeline. Each stage refines the abstraction level of the data.

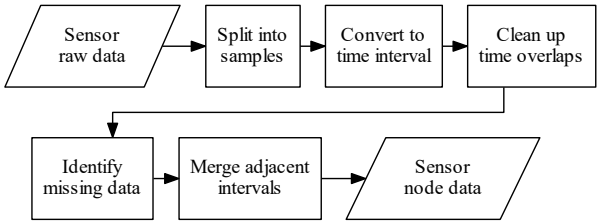


Fig. 3. Stage 1: Sensor node

These stages represent data as a stream of time intervals of variable lengths; each time interval describes a value (e.g., sensor measurement, motion in the room, location) for a part of the time line such that $t_{start} \leq t < t_{end}$. The value does not change within the time interval. The time intervals are ordered by time and, except for the sensor node stage, the time intervals in the data streams do not overlap in time. We define the data structures for the input and output time-intervals of each stage in the following sections and describe the processing steps of each stage.

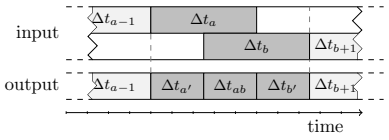
Stage 1: Sensor node

The first stage, illustrated in Figure 3, refines raw sensor data into time intervals to be processed as a data stream. This stage is run separately for each sensor; it consumes raw data samples and produces sensor node time-intervals. The sensor data is cleaned up for time overlaps, merged, and scanned for time periods with missing sensor data. The input and output data structures are described in Table II. This stage is performed on the server in this project, but it would be possible to implement this stage in the sensor network sink or gateway. Below, we describe the processing steps of this stage:

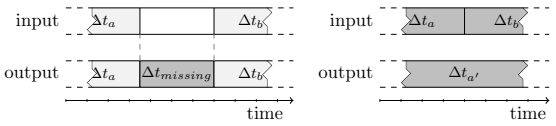
1) Split into samples: When sensor data arrives, the gateway assigns a time stamp to the data. The motion sensor nodes send 120 bits of motion data every 10 minutes. Each bit represents the physical motion sensed during a 5-second sample. We convert this data into 120 individual samples using the gateway's time stamp for the first sample, and

TABLE II
SENSOR NODE STAGE DATA REPRESENTATION

	Input: sensor sample		Output: sensor node time interval	
Motion sensor	sensorID timestamp motion	string datetime bit	sensorID start end motion	string datetime datetime true/false
Door sensor	sensorID timestamp door event	string datetime open/close	sensorID start end door state	string datetime datetime open/close
Missing data			sensorID start end missing	string datetime datetime true



(a) Clean up time overlaps



(b) Identify missing data (c) Merge adjacent intervals

Fig. 4. Sensor data clean-up steps

increment 5 seconds per element. These samples are then processed in the next step as individual samples, ordered by time stamp. Door sensor nodes send 'open' and 'close' event data. Incoming door sensor data are converted to a single sample and processed further in the next step. The motion and door sample data structures are described in the input column of Table II.

2) Convert to time interval: The sensor samples need to be converted into time intervals, which is the data structure used for the remainder of the data stream. Each motion sample is converted to an interval that starts at the sample's time stamp and ends 5 second later. The stream of door samples is converted to time intervals such that each interval spans the time between two door samples. The first door sample initiates the interval and sets the door state for the interval, and the next door sample's time stamp sets the end of the interval and also defines the start and door state of the following interval. The motion and door interval data structures are described in the output column of Table II.

3) Clean up time overlaps: The gateway's timestamps for motion data arrays will not always be exactly 10 minutes apart due to possible clock drift between the sensor node and gateway. We have observed time shifts of $0-1$ seconds. This means that the last motion sample from one motion data array, Δt_a, might overlap with the first sample of the next motion data array, Δt_b. These overlaps need to be cleaned up so that there is one and only one sensor value for each point in time. We do this by splitting up overlapping time-intervals Δt_a and

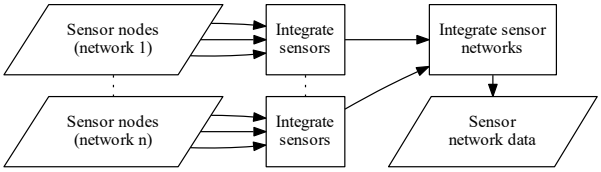


Fig. 5. Stage 2: Sensor network

Δt_b into time intervals $\Delta t_{a'}$, Δt_{ab}, and $\Delta t_{b'}$ as described in Figure 4a. Interval Δt_{ab} will be shorter than the sample size (i.e., 5 seconds), and it will be marked as having motion if Δt_a or Δt_b indicates motion. This can be described with the following equations:

$$\Delta t_{a'} = \Delta t_a \setminus \Delta t_b$$
$$\Delta t_{ab} = \Delta t_a \cap \Delta t_b$$
$$\Delta t_{b'} = \Delta t_b \setminus \Delta t_a$$

4) Identify missing sensor data: Due to clock drift, there might also be short time periods without any data between two consecutive motion data arrays. There are also other potential reasons for time periods without any available data for a sensor (e.g., low battery level or radio interference). In these cases, we insert a time interval for missing data, as illustrated in Figure 4b. These missing intervals are used to notify system maintainers that there are potential problems in the sensor network. In subsequent stages of the activity recognition algorithm, the missing intervals are processed as if the motion sensor senses no motion.

5) Merge adjacent intervals: If a time interval Δt_a ends where time interval Δt_b starts, and both time intervals have identical door or motion values, then we can compress data by merging these adjacent time-intervals to a single time interval. This is effective because motion and door sensors produce binary data and the sensed data will likely be unchanged for long periods of time (e.g., time intervals between door openings can be many hours). This process is effectively performing a run-length encoding of the sensor time-intervals, illustrated in Figure 4c. This can be described by the following equation: $\Delta t_{a'} = \Delta t_a \cup \Delta t_b$

After performing these steps, the each sensor's raw data have been transformed into a data stream of disjointed time-intervals on the sensor node level.

Stage 2: Integrate sensors into sensor network

The second stage, illustrated in Figure 5, reads sensor node intervals from all the sensors of an apartment. Data streams from different sensors are integrated into a single data stream for the apartment, consisting of sensor network intervals. The data representation for input and output time-intervals is listed in Table III. This stage contains the following steps:

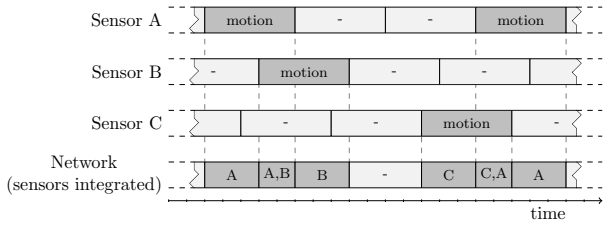


Fig. 6. Integrated intervals from all sensors to sensor network intervals

TABLE III
SENSOR NETWORK TIME-INTERVAL DATA REPRESENTATION

Input: sensor intervals		**Output: network interval**	
sensorID	string	networkID	string
start	datetime	start	datetime
end	datetime	end	datetime
Value (only one of these):		hasMotion	[ordered list of sensor ids]
- motion	true/false	isOpen	[list of door sensor ids]
- door	open/close		
- missing	true		

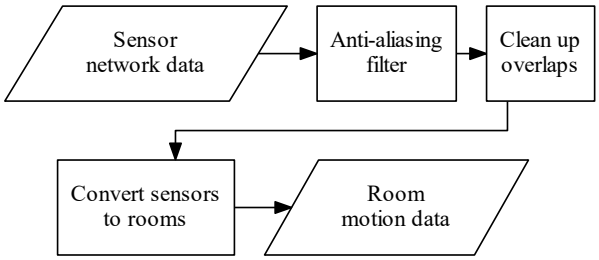


Fig. 7. Stage 3: Room motion

1) Integrate sensors: The time intervals of individual sensor nodes have motion or door data, which can have the values *true/false* or *open/close*, respectively. The time intervals where *missing* is *true* are converted so that *motion* is set to *false*. The input data streams from all sensor nodes are ordered by time. The start and end timestamps of all sensor time intervals are marked on a common time line, and this time line is then processed into sensor network time intervals so that a new sensor network time interval is created for each point in time when there are sensor value changes. This algorithm is described in Figure 6.

2) Integrate sensor networks: In larger or multi-floor apartments, multiple multiple sensor networks might be deployed in each apartment. In this case, we would need to integrate the data from these sensor networks into a single stream, similar to how individual sensors were integrated into a network in the previous step. This multi-network integration step is, however, skipped in this project since each apartment only has a single sensor network.

Stage 3: Room motion

In the third stage, described in Figure 7, sensor network data are read and transformed into room motion data for the apartment.

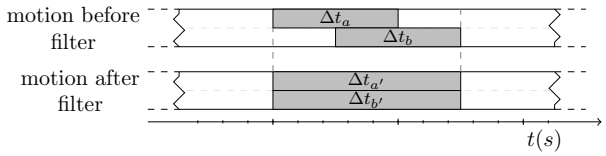


Fig. 8. Anti-aliasing filtering of motion data. The overlapping motion intervals are extended to avoid aliasing.

TABLE IV
ROOM MOTION DATA REPRESENTATION

Input: network interval		**Output: room interval**	
networkID	string	apartmentID	string
start	datetime	start	datetime
end	datetime	end	datetime
hasMotion	[ordered list of sensor ids]	hasMotion	[ordered list of room ids]
isOpen	[list of sensor ids]	isOpen	[list of door ids]

1) Anti-aliasing filter: If a sensor sample shows motion, the sensed motion could have happened anytime during the 5-second sample. This can lead to aliasing when considering multiple sensors; the motion time-intervals might have a different order than the actual order of motion observed in the apartment. We can avoid aliasing by extending the temporally overlapping motion time-intervals by up to one sample length. The anti-aliasing filter is illustrated in Figure 8; motion samples from sensors a and b overlap in time. Since we do not know in which order the motion actually occurred within these samples, we extend the time intervals to avoid aliasing.

2) Clean-up sensor overlaps: Two or more sensors might sense motion partly in the same physical area. Information about overlapping sensors is configured into the system manually at time of installation. Only one sensor is the primary sensor in each room, and other sensors detecting motion in the same area are secondary sensors of that area. When the primary sensor and one or more secondary sensors sense motion simultaneously, all secondary sensors are ignored. This is done by transforming the sensor network data stream so that secondary sensors are removed from the $hasMotion$ list if their primary sensor is present in $hasMotion$.

3) Convert sensor motion to room motion: After processing the sensor network data, the final step of this processing stage converts the sensor data to room data. The data representation of the room motion data is described in Table IV. Each sensor is located in a certain room r, so we replace the sensor identifiers with room identifiers in the $hasMotion$ list and replace sensor identifiers with door identifiers in the $isOpen$ list. By having the room list $hasMotion$ ordered by start of motion, we can use the order in later stages to infer the room sequence when the resident is moving from one room to another.

Stage 4: Location classification

The fourth stage, described in Figure 9, transforms the room motion data into data indicating the resident's location. Each

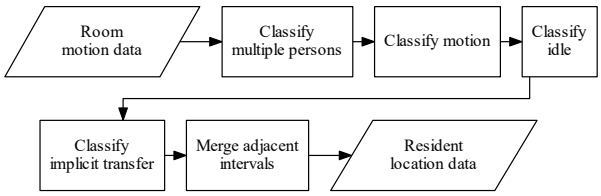


Fig. 9. Stage 4: Resident location classification

TABLE V
CLASSIFY MOTION INTO LOCATION

$hasMotion(\Delta t)$,	$isOpen(\Delta t)$	$\Rightarrow Location(\Delta t)$
any value	not empty	Outdoors
empty	empty	Unknown
$[r_i]$	empty	In room r_i
$[r_i, \ldots, r_k]$	empty	Transfer $[r_i, \ldots, r_k]$

time interval is classified as one of the following location classes:

- In room: the person is staying in a specific room;
- Transfer: the person is moving between rooms;
- Outdoors: the person is outside the apartment;
- Unknown: the location could not be determined.

We classify the location in the following steps.

1) Classify motion: The room motion data stream is transformed into a resident location data stream. In this step, we classify time intervals with an open door to 'Outdoors'. Time intervals with room motion are classified to either 'In room' or 'Transfer', as listed in Table V. In this context, 'Transfer' means that the person is moving between rooms in the apartment. Time intervals with no motion are initialised to 'Unknown' and then classified in subsequent steps. The resident location data representation is defined in Table VI.

2) Classify idle time-intervals: The idle time-intervals (location: 'Unknown') are classified based on the time intervals before and after the idle time-intervals. In certain cases, such as the door/hall transfers, we need to make special considerations for the classification. But in general, we can follow the model listed in Table VII.

3) Classify implicit transfer: Until this point, the location 'Transfer' covers time intervals where there are motion in multiple rooms. In cases where 'In room r_i' is succeeded by 'In room r_j', we insert an implicit transfer time interval of one sample size.

TABLE VI
RESIDENT LOCATION DATA REPRESENTATION

Input: room interval		**Output: location interval**	
apartmentID	string	apartmentID	string
start	datetime	start	datetime
end	datetime	end	datetime
hasMotion	[ordered list of room ids]	location	'unknown'/'inroom'/ 'transfer'/'outdoors'/
		rooms	[ordered list of ids]
isOpen	[list of door ids]	*value of* rooms *is based on location:* *'unknown'* ⇒ *empty list* *'inroom'* ⇒ *single room id* *'transfer'* ⇒ *ordered list of room ids* *'outdoors'* ⇒ *door id*	

TABLE VII
CLASSIFICATION OF 'UNKNOWN' LOCATION, BASED ON TIME INTERVALS BEFORE AND AFTER

$Location$ (Δt_{before})	$Location$ (Δt_{after})	$\Rightarrow Location(\Delta t)$
Outdoors	Outdoors	Outdoors
Outdoors	In room r	In room r
In room r	Outdoors	In room r
Outdoors	Transfer $[r, \ldots]$	In room r
Transfer $[\ldots, r]$	Outdoors	In room r
In room r	In room r	In room r
In room r	Transfer $[\ldots]$	In room r
Transfer $[\ldots]$	In room r	In room r
In room r_i	In room r_j	Split Δt in half: In room r_i for first half In room r_j for second half
Transfer: $[r_i, \ldots]$	Transfer: $[r_j, \ldots]$	Depending on length of Δt Transfer $[r_i, \ldots, r_j, \ldots]$ if $\Delta t < 1minute$
$[\ldots]$	$[\ldots]$	Unknown, if $\Delta t \geq 1minute$

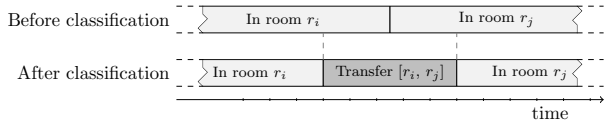


Fig. 10. Classifying implicit transfer: insertion of a transfer interval between 'In room r_i', and 'In room r_j' when the resident changes room. The inserted transfer interval is one sample length (i.e., 5 seconds) long.

We identify these implicit transfers using a sliding time window $W(\Delta t)$ with the size of one sample (i.e., 5 seconds). For each Δt where $W(\Delta t)$ contains 'In room' for multiple rooms $r_i, \ldots, r_k$: classify $Location(\Delta t) = Transfer[r_i, \ldots, r_k]$, and keep the order of the rooms according to the order within $W(\Delta t)$. This process is illustrated in Figure 10.

4) Merge adjacent time-intervals: Since the location classification happens in multiple steps, there may be adjacent time-intervals with the same location state. In order to correctly classify activities and duration of room visits, we need to merge these adjacent resident location time-intervals when the location does not change. This is done in a similar way as described in Figure 4c.

Stage 5: Activity recognition

The final stage, described in Figure 11, classifies resident activities based on the resident's location data. In this context, an activity describes what the resident is doing, such as the resident wandering around the apartment, being in the bedroom during the night, or being outside the apartment.

1) Identify wandering: There is no precise definition of wandering, but it can be described as a cognitively impaired person moving around aimlessly (Lai and Arthur, 2003). In the context of this project, we expect a wandering resident to walk around the apartment and change rooms frequently. To identify this behaviour, we analyse the *transfer* time-intervals

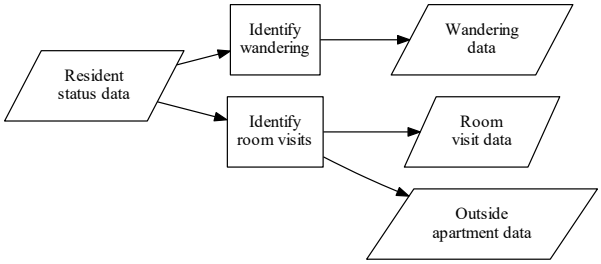


Fig. 11. Stage 5: Activity recognition

TABLE VIII
ACTIVITY DATA REPRESENTATION

Name	Type	Activity classification
apartmentID	string	
start	datetime	
end	datetime	
Value (only one of these):		
wandering	true/false	*wandering*
room	room id	*room visit*
outside	true/false	*outside the apartment*

since the resident is expected to transfer between rooms while wandering. We do not focus on *motion* within a single room, since there are many different physical activities, in addition to wandering, that could trigger *motion* measurements in a room. We need to identify the following indications of wandering:

- Frequent transfers between rooms indicate wandering behaviour;
- Short time-intervals in rooms between room transfers indicate wandering behaviour;
- Low frequency transfers between rooms do not indicate wandering behaviour.

To achieve this, we define the following wandering indicator based on a sliding time window $W(\Delta t)$:

$$wander(\Delta t) = \frac{\text{total time of Transfer during } W(\Delta t)}{\text{window size } \Delta t}$$

We can now define wandering as the time intervals where the wandering indicator exceeds a certain threshold:

$$Wandering(\Delta t) = \begin{cases} true, & wander(\Delta t) \geq threshold \\ false, & wander(\Delta t) < threshold \end{cases}$$

By running the sliding time window on the resident location data stream we produce wandering time-intervals for the activity data stream; see the data representation in Table VIII. For the wandering indicator, we use the following heuristics: window size Δt = 1 minute, threshold = 0.5.

2) Identify room visits: For each time interval Δt_i where $Location(\Delta t_i) =$ 'In room r':

- Classify Δt_i as a room visit in room r;
- If r is a door, classify Δt_i as time outside the apartment instead of room visit.

Data for room visits and time spent outside the apartment are represented as described in Table VIII.

TABLE IX
DATA PROCESSING STAGE OUTPUTS USED FOR SYSTEM VERIFICATION

Data processing stage	System check
Stage 1: Sensor node	Sensors & network operational
Stage 2: Sensor network	Synchronisation between sensors
Stage 3: Room motion	Room overlaps & antialiasing
Stage 4: Location	Location and transfer classification
Stage 5: Activity recognition	Wandering & outside classification

VI. EVALUATION

The system design is evaluated in this paper through the implementation and deployment of the system design described above. To verify the data processing system's functionality and correctness, we set up a test protocol of predefined and timed motion-patterns:

1) Moving around in one room at a time;
2) Standing still in one room at a time;
3) Moving in and between rooms in a predefined pattern;
4) Going outside and returning to the apartment.

This test protocol, lasting 15–30 minutes depending on the apartment layout, is performed and documented by the tester. The outputs of the data processing stages during the test protocol are then checked against the motion patterns documented in the test protocol to verify the system operates as expected; see Table IX.

By performing the test protocol for each apartment, when deploying the system, we could verify that the data processing performed as intended. The algorithm was able to identify the activities as performed by the test person as the sensors' data matched the test person's motion patterns. To evaluate the relevance of the data produced by the system, we consider the user needs in Table I and compare them with the outcomes of the data processing stages. As we can see in Table X, the user needs can be directly mapped to the data produced by data processing stages 3–5.

Collecting feedback from the target group of the system—elderly people with memory impairment caused by, for example, Alzheimer's disease—present its own challenges, depending on the level of cognitive impairment. Hence, the usefulness of this system design was evaluated by meeting with and collecting feedback from the nurses who used this system. A case study of this deployed system by Klemets et al. (2019) shows that the activity data helped nurses better understand their clients' daily rhythms and make well-informed decisions. In two cases, routine night visits by home care nurses could be cancelled as a consequence of deploying this system. The activity data helped the nurses make the decision that one client needed to be moved to a care home, and there were three clients for whom the activity data helped nurses establish that they could continue living in their apartment instead of moving to a care home. The cost savings of cancelled care home applications and cancelled night-time home care visits cover the hardware and maintenance costs of the smart home activity recognition system by a wide margin. Helping nurses identify clients in need of being moved from independent living to

TABLE X
USER REQUIREMENTS MAPPED TO DATA PROCESSING STAGES

Issue (Klemets et al., 2017)	**Data processing stage**
Raise attention	
Urinary tract infection	Stage 5: Room visit (bathroom)
Digestion	Stage 5: Room visit (bathroom)
Depression	Stage 5: Outside, Wandering
Wandering behaviour	Stage 5: Room visits, Outside, Wandering
Pain	Stages 3 & 5: Room motion (bedroom), Wandering
Memory impairment	Stage 5: Room visits, Outside, Wandering
Provide insight	
Tiredness and fatigue	Stage 5: Room visit (bedroom)
Functional ability	Stages 3 & 5: Room motion (all rooms), Outside, Wandering
Pressure ulcer	Stages 3 & 5: Room motion, Room visit
Hypnotic drug effect	Stages 3 & 5: Room motion (all rooms), Wandering
Nutrition	Stage 5: Room visit (kitchen)

a care home improves the cost efficiency of both home care as well as care home services, as the elderly care resources are better targeted (Weissert et al., 2003). The usefulness of the system is also confirmed by the fact that the home care services are extending the deployments of this system as part of a follow-up project.

VII. DISCUSSION

The contribution of this paper is a sensor data streaming pipeline for a **low complexity** smart home activity recognition system, designed for elderly persons living alone. The system was designed in collaboration with home care nurses, with the focus on meeting the user's needs. The focus on user needs drive the emphasis on low complexity for the whole system, it has benefits such as low costs (i.e., hardware, power consumption, and maintenance) and facilitates a robust and modular system design. The system uses a wireless sensor network with motion and door sensors to monitor activity in the homes of elderly people.

The individual steps of the data processing pipeline process data as a stream and can be implemented using fixed-point arithmetic. This results in low computational complexity and low space complexity within the capacity of the microcontroller used in the WSN nodes. The detailed description of the data processing stages in Section V illustrates the level of complexity in each stage of the data processing algorithm. The modular system design, with documented data streams as interfaces between data processing stages, makes it easy to reason about the system and evaluate the correctness of the system.

The system was deployed in service home apartments for elderly people, providing nurses with information about their clients' daily pattern of activity. The evaluation, deployment, and collected user feedback from the real-world deployment confirm the usefulness of this system (Klemets et al., 2019). The low complexity of the data processing, combined with a sparse sensor network, keep the system costs at a sustainable level, suitable for healthcare organisations. This is also confirmed by an extended follow-up project with deployment in up to 40 apartments, funded in part by a public-sector healthcare organisation.

There are examples in the literature of smart home systems with dense sensor networks (Cook, 2012), and supervised machine learning algorithms for activity recognition are prevalent (Rashidi and Mihailidis, 2013). The system described in this paper is an example of a low-complexity design of a smart home system with a sparse network of sensors and an unsupervised knowledge-driven activity recognition approach. In this project, we primarily used PIR motion sensors, which are commonly used in this type of research (Peetoom et al., 2015), although many other type of sensors have been used by other researchers (Krishnan and Cook, 2014; Rashidi and Mihailidis, 2013). In the context where this system was deployed—helping nurses understand how seniors with memory impairment cope living alone in an apartment—the activities recognised by the system were selected based on discussions with the nurses using the system. Based on user feedback from the deployment, the scope of activity recognition could have been reduced from what we have presented in this paper. The pattern of room visits and going outside the apartment were the main activities that the nurses found useful during this project. Thus, there is still room to reduce the complexity of the system.

The current design is based on the assumption that only one person lives in the apartment. We are relying on the user to know or find out when there have been multiple persons in the apartment and disregard the activity classification for these periods of time. As the system has been deployed in a service home setting, and the nurses have mainly used the system to monitor how the resident copes by herself/himself during the night, we were able to cope with this shortcoming. In future work, we intend to detect when multiple persons are present in the apartment and disable the single-person activity recognition for these time intervals, informing users that there are multiple persons present. To detect when multiple persons are present, we propose using a door sensor with two infrared beams that sense people entering or exiting through the apartment door. Based on the door sensor data and the apartment's motion data, the system will classify each time interval between apartment entry/exit events as 'empty apartment', 'single person', 'multiple persons'. Another limitation of the system is that it handles missing sensor data packets as if the sensor was not sensing any motion. This approach increases the system's robustness, enabling data processing when there are temporary sensor or network malfunctions. The loss of a single packet means 10 minutes of data loss, which is not critical for the nurses using this system. However, this method will give odd results when there is a longer sensor outage. This issue has been mitigated by using a monitoring system to notify IT personnel of data loss, low battery levels,

poor wireless connectivity, and other network-related issues. We propose an improvement to visualise explicitly, in the UI, when sensor data packets are missing. This would reduce the risk of the user misinterpreting data during a sensor or network failure.

In addition to the proposed improvements listed above, we propose the following in future work:

Anomaly detection: By comparing activity patterns with earlier activity data, the system can detect anomalies and inform care personnel about changes in activity patterns and, thus, a potential change in the client health status.

Detect sensor overlaps automatically: Instead of manually configuring information about multiple sensors overlapping the same area, it could be possible to detect the overlapping areas automatically by, for example, including it in the apartment installation test protocol. In addition, an alternative approach to handle overlapping areas would be to define these areas as virtual rooms, and take them into account in the data processing algorithm.

Distribute data processing across WSN: We propose distributing the data processing pipeline across the constrained devices of the WSN. The specified input and output data streams between data processing stages suggest that the inter-stage data streams can be transported over the network, and the low complexity requirements per stage suggest that data processing can be performed on the constrained devices of the WSN. This would move data processing closer to the edge of the network, which could improve efficiency, as well as data security and privacy (Shi et al., 2016).

Scalability: In the current setting, the typical monitored home is a one bedroom apartment, which is monitored by fewer than 10 sensors. Due to the low computational and space complexity of the system, the data transfer speed is the limiting factor when it comes to scalability. To scale up the system to a larger apartment with a greater number of sensors per apartment, we can extend the number of gateways to multiple gateways per apartment. Distributing the data processing across the WSN, as mentioned above, improves scalability in terms of deploying the system to a larger number of apartments.

VIII. Conclusion

In this paper, we presented a data processing pipeline for a stream of sensor data in a motion sensor-based smart home system. The data processing pipeline is split into five processing stages, where each stage's interface is specified as input and output streams and their data structures. Each stage refines the sensor data to a higher abstraction level: stage 1 processes raw sensor data and cleans it up; stage 2 integrates data from multiple sensors into network level data; stage 3 refines sensor network data into motion per room; stage 4 classifies the location of the person; stage 5 performs activity recognition. The detailed description of the data processing pipeline illustrates the modularity of the design and facilitates the evaluation of the system, as well as the feasibility of distributing the data processing into the constrained devices of the wireless sensor network. The correctness and suitability for meeting the needs of users of this data processing pipeline was evaluated with a server-based implementation, processing the data generated by WSNs used for motion and door sensors installed in 12 elderly persons' apartments. A protocol was used to test the sensor network in the apartments. In future work, we intend to add the detection of multiple persons in the apartment and visualisation of data loss. We also propose anomaly detection, improved sensor overlap handling, and distributing the data processing across the wireless sensor network.

References

Abadi, D. J., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J.-H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., et al. (2005). The design of the borealis stream processing engine. In Cidr, volume 5, pages 277–289.

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., and Zdonik, S. (2003). Aurora: A new model and architecture for data stream management. The VLDB Journal, 12(2):120–139.

Akidau, T., Balikov, A., Bekiroğlu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., and Whittle, S. (2013). Millwheel: fault-tolerant stream processing at internet scale. Proceedings of the VLDB Endowment, 6(11):1033–1044.

Andueza Robustillo, S., Corsini, V., Juchno, P., and Marcu, M. (2015). Demography report. Publications Office of the European Union.

Atmel (2013). ZigBit 2.4GHz Wireless Modules ATZB-24-A2/B0 Datasheet. Atmel Corporation. 8226C-AVR-07/2013.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '02, pages 1–16, New York, NY, USA. ACM.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., et al. (2003). Telegraphcq: Continuous dataflow processing for an uncertain world. In Cidr, volume 2, page 4.

Chen, L., Nugent, C. D., and Wang, H. (2012). A knowledge-driven approach to activity recognition in smart homes. IEEE Transactions on Knowledge and Data Engineering, 24(6):961–974.

Cook, D. J. (2012). Learning setting-generalized activity models for smart spaces. IEEE intelligent systems, 27(1):32–38.

Demers, A., Gehrke, J., Rajaraman, R., Trigoni, N., and Yao, Y. (2003). The cougar project: a work-in-progress report. ACM Sigmod Record, 32(4):53–59.

Diallo, O., Rodrigues, J. J., and Sene, M. (2012). Real-time data management on wireless sensor networks: A

survey. Journal of Network and Computer Applications, 35(3):1013–1021.

Golab, L. and Özsu, M. T. (2003). Issues in data stream management. SIGMOD Rec., 32(2):5–14.

Klemets, J., Määttälä, J., and Hakala, I. (2019). Integration of an in-home monitoring system into home care nurses' workflow: A case study. International journal of medical informatics, 123:29–36.

Klemets, J., Määttälä, J., Jansson, J., and Hakala, I. (2017). Nurses' perspectives on in-home monitoring of elderlies's motion pattern. Studies in Health Technology and Informatics; 235.

Krishnan, N. C. and Cook, D. J. (2014). Activity recognition on streaming sensor data. Pervasive and mobile computing, 10:138–154.

Lai, C. K. and Arthur, D. G. (2003). Wandering behaviour in people with dementia. Journal of advanced nursing, 44(2):173–182.

Liu, L., Stroulia, E., Nikolaidis, I., Miguel-Cruz, A., and Rincon, A. R. (2016). Smart homes and home health monitoring technologies for older adults: A systematic review. International journal of medical informatics, 91:44–59.

Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2005). Tinydb: an acquisitional query processing system for sensor networks. ACM Transactions on database systems (TODS), 30(1):122–173.

Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G., Olston, C., Rosenstein, J., and Varma, R. (2003). Query processing, resource management, and approximation in a data stream management system. In Proceedings of the 2003 CIDR Conference. CIDR.

Peetoom, K. K., Lexis, M. A., Joore, M., Dirksen, C. D., and De Witte, L. P. (2015). Literature review on monitoring technologies and their outcomes in independently living elderly people. Disability and Rehabilitation: Assistive Technology, 10(4):271–294.

Rashidi, P. and Mihailidis, A. (2013). A survey on ambient-assisted living tools for older adults. IEEE journal of biomedical and health informatics, 17(3):579–590.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5):637–646.

United Nations, Department of Economic and Social Affairs, Population Division (2015). Technical report.

van Kasteren, T. and Kröse, B. (2007). Bayesian activity recognition in residence for elders. In editor, editor, 3rd IET International Conference on Intelligent Environments (IE 07), pages 209–212. IET.

Weissert, W., Chernew, M., and Hirth, R. (2003). Titrating versus targeting home care services to frail elderly clients: An application of agency theory and cost-benefit analysis to home care policy. Journal of aging and health, 15(1):99–123.

Zhuang, L. Q., Zhang, J. B., Zhang, D. H., and Zhao, Y. Z. (2005). Data management for wireless sensor networks: research issues and challenges. In 2005 International Conference on Control and Automation, volume 1, pages 208–213. IEEE.