Distributed Activity Recognition with Fuzzy-Enabled Wireless Sensor Networks

Mihai Marin-Perianu[1], Clemens Lombriser[2], Oliver Amft[2]
Paul Havinga[1], Gerhard Tröster[2]

[1] Pervasive Systems, University of Twente, The Netherlands
{m.marinperianu, p.j.m.havinga}@utwente.nl
[2] Wearable Computing Lab, ETH Zürich, Switzerland
{lombriser, amft, troester}@ife.ee.ethz.ch

Abstract. Wireless sensor nodes can act as distributed detectors for recognizing activities online, with the final goal of assisting the users in their working environment. We propose an activity recognition architecture based on fuzzy logic, through which multiple nodes collaborate to produce a reliable recognition result from unreliable sensor data. As an extension to the regular fuzzy inference, we incorporate temporal order knowledge of the sequences of operations involved in the activities. The performance evaluation is based on experimental data from a car assembly trial. The system achieves an overall recognition performance of 0.81 recall and 0.79 precision with regular fuzzy inference, and 0.85 recall and 0.85 precision when considering temporal order knowledge. We also present early experiences with implementing the recognition system on sensor nodes. The results show that the algorithms can run online, with execution times in the order of 40ms, for the whole recognition chain, and memory overhead in the order of 1.5kB RAM.

1 Introduction

Sensor miniaturization and advances in wireless communication made real visionary technologies such as wearable computing, Wireless Sensor Networks (WSN) and the Internet of Things. As a result, we witness today a rapidly growing uptake of these technologies in various industrial and business related fields [5]. In particular, providing context-aware assistance to workers in industrial environments [1] has a number of potential benefits: (1) improves the productivity at work by supporting the workers with just-in-time, relevant information, (2) increases the reliability by supervising safety-critical process steps and (3) accelerates the learning curve of new, unskilled workers.

In this paper, we focus on the concrete application of assembling and testing car body parts at a car manufacturing site [16, 3]. The mechanical assembly activities are supervised by a distributed system composed of wireless sensor nodes worn by the workers, embedded into the tools and deployed within the infrastructure. Having the technology integrated in the usual work environment

ensures a simplified, unobtrusive human-machine interaction, which is important to foster user acceptance [6]. The same system can additionally support the training of new employees. The most important technical objective of the system is to recognize the user activities with high accuracy and in real time, in order to provide prompt and exact assistance. We can expect several major difficulties related to: the inaccurate and noisy sensor data, the very limited resources (processing, memory and energy), the high variability of the data (e.g. for sensors placed on different parts of the body) and the unreliable, low bandwidth wireless communication. To overcome these problems, we propose a distributed activity recognition system based on fuzzy-enabled WSN. Fuzzy logic represents nowadays a well-established field with an impressive number of successful applications in the industry and other connected areas [12]. Fuzzy inference systems (FIS) constitute an effective tool for WSN because (1) they can be implemented on limited hardware and are computationally fast, and (2) they can handle unreliable and imprecise information (both numeric and linguistic information), offering a robust solution to decision fusion under uncertainty [14]. In addition, fuzzy logic reduces the development time; it is possible to have a running system based only on the common-sense description of the problem, and then tune it using expert knowledge or automatic training and learning methods.

In our approach we decompose the activity recognition process in three steps: (1) the detection of action events using sensors on the body, tools, and the environment (based on the previous work of Amft et al. [3]), (2) the combination of events into basic operations and (3) the final activity classification. For performance evaluation, we use the experimental data from 12 sensor nodes with 3D accelerometers, obtained during the car assembly trial conducted in [3] (see Fig. 1). Sec. 3 presents the overview of the distributed architecture. Sec. 4 analyzes the properties of the experimental data and derives the difficulties in achieving an accurate recognition. The detailed design of the FIS and the performance results are given in Sec. 5 and 6. As an extension to the normal fuzzy inference, we show how temporal order knowledge can improve the overall accuracy for the activities executed in sequences of operations. In order to study the feasibility of our solution, we implement the recognition algorithms on the Tmote Mini platform. The results from Sec. 7 show that the algorithms can run online even on resource constrained sensor nodes. Sec. 8 discusses several important factors that can affect the performance of fuzzy-enabled WSN. Finally, Sec. 9 concludes the paper.

2 Related Work

Activity recognition is a topic of high interest within the machine vision community. In particular, we can trace back the fundamental idea of dividing the recognition problem into multiple levels of complexity. In the "Inverse Hollywood Problem", Brand [4] uses coupled hidden Markov models (HMM) to visually detect causal events and fit them together into a coherent story of the ongoing action. Similarly, Ivanov and Bobick [7] address the recognition of visual activ-

Fig. 1. A car assembly worker performing several activities. Sensors are both worn by the worker and attached to the tools and car parts.

ities by fusing the outputs of event detectors through a stochastic context-free grammar parsing mechanism. Wren et al. [18] show that even low resolution sensors (motion detectors and ultrasonic sensors) can provide useful contextual information in large buildings.

From a different perspective, distributed event detection has received considerable attention in the field of WSN. The research focuses on fundamental issues of WSN, such as reducing the number of messages needed for stable decisions [11], minimizing the effects of data quantization and message losses [13], and using error correcting codes to counteract data corruption [17]. The sensor nodes are typically assumed to sample at low data rates and to implement simple binary classifiers based on maximum likelihood estimates. However, recent work [15] shows that sensor nodes are able to execute more elaborate tasks, such as the classification of sound data. This result opens promising perspectives for using WSN in complex activity recognition processes.

The general problem of inferring complex events from simple events is also considered in middleware systems, such as DSWare [8] or Context Zones [10]. Still, there is a gap between the inference process in these systems and the algorithms operating with low-level sensor signals. In particular, it is difficult to evaluate the influence of different levels on the overall recognition performance.

In this paper, we evaluate a distributed recognition system that can run entirely on sensor nodes and classify the activities online. The resource constraints (computation, energy, memory, communication) are extreme and make impractical the usage of state-of-the-art HMM or Bayesian inference methods. Instead, we propose a lightweight fuzzy inference engine that can run on limited hardware and proves stable to inaccurate sensor data.

3 Solution Overview

In this section we start with a brief overview of fuzzy logic, and then present in detail the architecture of the fuzzy-based activity recognition system.

3.1 Overview of Fuzzy Inference Systems

The fuzzy inference process maps crisp inputs to crisp outputs by executing four basic steps. First, the inputs are *fuzzified* through the membership functions. In

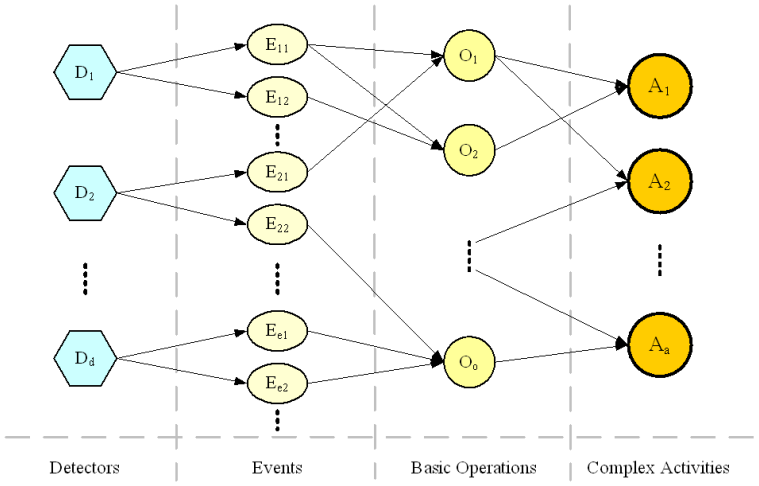


Fig. 2. Distributed activity recognition architecture. The three main steps are: the detection of simple events (see also [3]), the combination of events into basic operations and the final activity classification.

contrast with crisp logic, the membership functions allow an input variable to belong to different fuzzy sets, with partial degrees of confidence (for example, the acceleration can be fuzzified as 0.7 *High* and 0.2 *Medium*). Second, the fuzzified inputs are processed according to the *rule base*, which is a collection of IF-THEN statements with fuzzy propositions as antecedents and consequences. The rule base is derived from expert knowledge or generated automatically from numerical data. Third, the evaluation of the rules is *aggregated* into a fuzzy output. Finally, the fuzzy output is *defuzzified* to a crisp value, which is the final FIS result.

3.2 Distributed Activity Recognition Architecture

The analysis of the car assembly scenario resulted into the following architecture design issues to be considered:

1. Multiple *detectors* are available: sensors worn on the body, attached on the objects and tools, or deployed within the infrastructure. Although heterogeneous, most of these detectors have limited capabilities and low accuracy.
2. Despite their limitations, the detectors can use simple methods to extract online the relevant features of the activities from the sampled data. Online processing is a design choice imposed by the low WSN bandwidth, which makes it unfeasible to transport large amounts of data to a central fusion point. Since the features are computed over a sliding time window and reported asynchronously by different detectors, we refer to them as *events* (see [3] for the detailed algorithms and performance of the event detectors).
3. A single event from one detector gives just an estimation of the real activity going on as it is perceived by the sensor at this location. However, fusing the information from several detectors reporting similar (or correlating) events

within the same timeframe leads to a high confidence in recognizing that the user is performing a certain *basic operation*. For example, the basic operation "Mount the screws" can be inferred from the events signaled by the sensors on the user's arm, on the screwdriver and on the car part being assembled.

4. Sequences of basic operations form *complex activities*, which represent the final output of the recognition system. For example, the complex activity "Mount the brake light" consists of three basic operations: "Pick-up the brake light", "Insert the brake light" and "Mount the screws". We notice from this example that the order of the operations can be an important parameter for distinguishing among different activities (see Sec. 5.2 for details on how the execution order is used to improve the overall recognition performance).

Fig. 2 summarizes the observations above. We denote the set of detectors by $\mathcal{D} = \{D_1, D_2, ..., D_d\}$. Each detector D_i can produce the set of events $\mathcal{E}_i = \{E_{i1}, E_{i2}, ..., E_{ie_i}\}$. Events from various detectors are combined to obtain the basic operations $\mathcal{O} = \{O_1, O_2, ..., O_o\}$. The identification of the basic operations represents the first level of data fusion. The aim is to reduce the number of inputs and consequently the classification complexity. The second level of data fusion performs the combination of multiple observed basic operations into more complex activities $\mathcal{A} = \{A_1, A_2, ..., A_a\}$, where each A_i is composed of a set of operations $\{O_{i_1}, O_{i_2}, ..., O_{i_k}\}$, $1 \leq i \leq a$, $1 \leq k \leq o$.

There are two types of problems that adversely affect the performance of the recognition system. First, the low accuracy of the detectors may result into reporting false events (also referred to as *insertions*) or missing events (also referred to as *deletions*). Second, there are overlaps among different basic operations and also among different complex activities (i.e. the same events are involved in more operations and the same operations are performed during various activities), which can lead eventually to a misclassification due to confusion.

Fuzzy logic can be used throughout the whole activity recognition chain. The detectors usually identify the events with a certain confidence and cannot distinguish perfectly among similar events. This situation maps well to the notion of membership functions, as explained in Sec. 3.1. Likewise, combining events into basic operations is similar to fuzzy majority voting, where different weights are assigned to the detectors based for example on their placement. Rule-based fuzzy inference is appropriate for the final activity classification, by checking the occurrence of the right operations. An interesting extension is to include the time sequence of the operations in the inference process (see Sec. 5.2).

4 Experimental Data

In order to evaluate the performance of our system, we utilize the experimental data obtained by Amft et al. [3] from a car assembly trial. Throughout the trial, 12 sensors with 3D accelerometers were used as detectors. Their placement is shown in Table 1. The detectors could spot a total number of 49 distinct events. We use the confidences of recognizing the events, ranging between $(0; 1]$, as inputs

Table 1. Placement of the detectors.

Detector	Placement	Detector	Placement
D_1	Right arm	D_7	Trunk door
D_2	Left arm	D_8	Screwdriver 1
D_3	Upper back	D_9	Screwdriver 2
D_4	Front light	D_{10}	Front door
D_5	Brake light	D_{11}	Back door
D_6	Hood	D_{12}	Socket wrench

Table 2. Activity A_1-"Mount the front door".

Basic operation	Events (Detectors)
O_1-Pick-up the front door	$9(D_1)$, $15(D_2)$, $39(D_{10})$
O_2-Attach the front door	$40(D_{10})$
O_3-Mount the screws	$6(D_1)$, $20(D_3)$
O_4-Pick-up the socket wrench	$7(D_1)$, $21(D_3)$, $47(D_{12})$
O_5-Use the socket wrench	$10(D_1)$, $48(D_{12})$
O_6-Return the socket wrench	$49(D_{12})$
O_7-Close the front door	$16(D_2)$

to our fuzzy-based recognition system. As explained in Sec. 3.2, we first combine the events corresponding to a certain basic operation, then perform the complex activity classification. The following example will best explain this process.

Let us consider the complex activity A_1-"Mount the front door". [3] Table 2 lists in the left column the operations involved in this activity, O_1 to O_7. The right column enumerates the events (represented as numbers) and the corresponding detectors (given in brackets). We distinguish the following cases:

- *Operations identified by only one detector/event, such as O_2.* The confidence of executing such a basic operation is given by the associated detector/event confidence.
- *Operations identified by several detectors on the body, such as O_3.* Compared to the previous case, the combined confidence is higher, but still solely dependent on the accuracy of the gesture recognition. For computing the confidence of the basic operation, the average of the event confidences can be used. If multiple detectors are available, we can apply a fuzzy majority measure [9].
- *Operations identified by several detectors on the body and on the tools (e.g. O_4) or the car parts (e.g. O_1).* This is the best case, as we can fuse information from different sources and produce a more reliable detection. The fusion can be done through a simple weighted average or a fuzzy majority measure, if more detectors of the same type are available.

[3] The complete description of the activities can be found in [3].

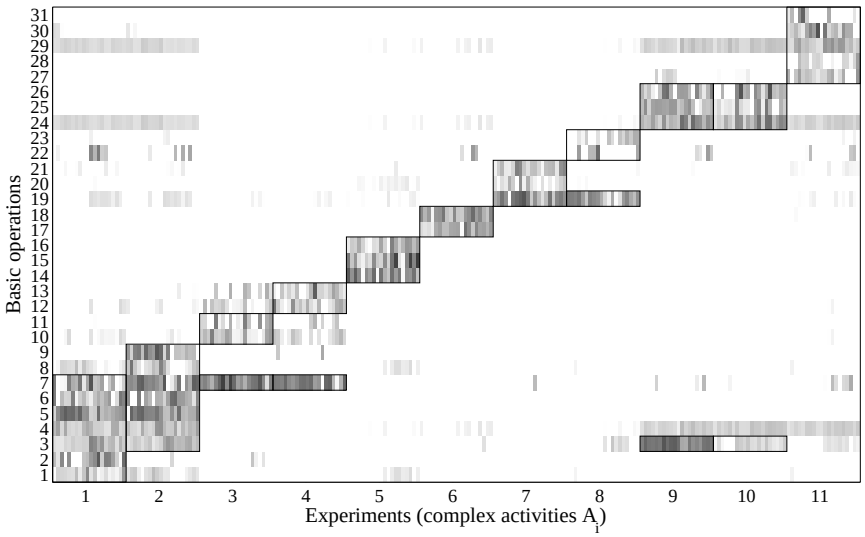


Fig. 3. Basic operations observed by the detectors in the car assembly experiment. The gray spots represent confidence values, where a darker spot means a higher confidence. Operations relevant for the activity are marked by rectangles.

Fig. 3 depicts a grayscale map of the whole car assembly experiment. On the horizontal axis we have 11 complex activities, each of them performed 10 times by each of two subjects. The total recorded data amounts to approximately 5 hours. On the vertical axis we have the basic operations derived from the events reported by the detectors. By using a first level of data fusion, we reduced the number of input variables from 49 events to 31 operations. The gray spots indicate the confidence levels of the basic operations; higher confidences correspond to darker regions. The solid line rectangles mark the real operations involved in each of the 11 activities.

We observe that the fuzzy-based classification method has to cope with the following difficulties:

- Activities with significant overlapping definitions, such as A_1 and A_2, which have the operations O_3 to O_7 in common.
- Activities such as A_3 and A_4, which have theoretically only O_7 in common, but in practice record a large amount of false detections (insertions) causing overlaps.
- Constantly high-confidence insertions, for example O_{29} during A_1 and A_2.
- Missed detections (deletions), such as O_{22} during A_8 or O_{31} during A_{11}. The continuity of the deletions in these examples suggests that they are caused by packet losses at the associated detectors.

5 Fuzzy-based Activity Recognition

In this section we present in detail the design the fuzzy-based recognition system. We first analyze several alternatives for the main FIS components, then extend the inference for sequences of operations ordered in time.

5.1 Fuzzy System Components

The experimental data described in Sec. 4 is used as input to the fuzzy inference for the final activity classification. There are three important components of the FIS: the membership functions, the rule base and the defuzzifier. Making the right choices for these components is the most difficult task in fuzzy system design. However, the difficulties can be leveraged if (1) there is expert knowledge about the process to be modeled and/or (2) there is a collection of input/output data that can be used for learning and training. The car assembly scenario matches both conditions. Therefore, we base the FIS design on the following list of facts derived from the actual process:

1. The number of output classes is known, equal to the number of complex activities (11).
2. The mapping of inputs (basic operations) to outputs is known from the definition of each activity.
3. The input data is noisy and contains insertions (see Sec. 4) that generate confusion among output classes.
4. The erroneous wireless communication causes deletions in the input data (see Sec. 4), which translate into non-activation of the fuzzy rules.

Fact 2 allows us to define the rule base. Each complex activity is selected by a rule taking the inputs corresponding to the activity definition. For example, activity A_7 generates the rule:

$$\begin{aligned} \text{IF} \quad & O_{19} \text{ is } High \text{ AND } O_{20} \text{ is } High \text{ AND } O_{21} \text{ is } High \\ \text{THEN} \quad & A_7 \text{ is } High \end{aligned} \tag{1}$$

The next step is to choose the membership functions for the fuzzy sets. For computational simplicity, we use trapezoidal membership functions. The first problem is to choose the upper threshold of the membership functions. For each input O_i, we compute the threshold as the mean value $m(O_i)$ over the training set. Consequently, confidences higher than $m(O_i)$ will be fuzzified to 1. The second problem arises from Fact 4: deletions (i.e. zero value inputs) determine the non-activation of the appropriate rules when using the usual *max-min* or *sum-product* inference. To alleviate this effect, we lower the zero threshold of the membership functions, so that zero input values are fuzzified to strictly positive numbers. The downside of this approach is that the level of confusion among output classes increases. We will see however in Sec. 6 that this has no major impact on the overall performance.

The last component to discuss is the defuzzifier. The standard centroid-based defuzzification is the most time-consuming operation on a sensor node, taking

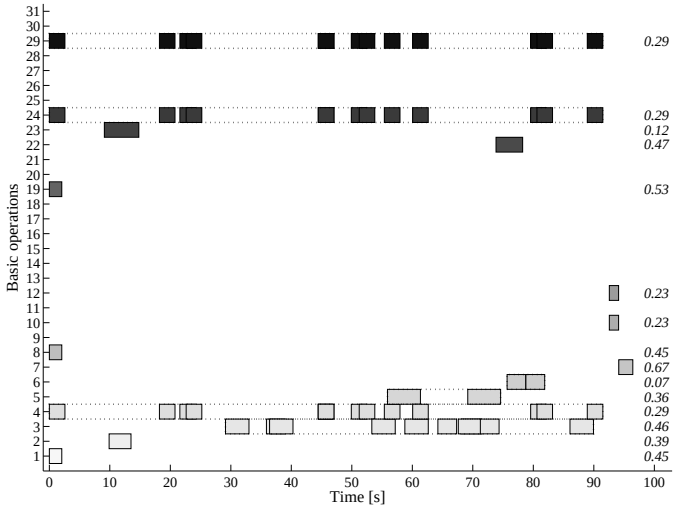


Fig. 4. Basic operations reported during activity $A_1 = \{O_1, \ldots, O_7\}$. The operations in the upper part are mistakenly inserted by the detectors. Multiple occurrences of operations are combined (dotted lines) to analyze the correct order in time.

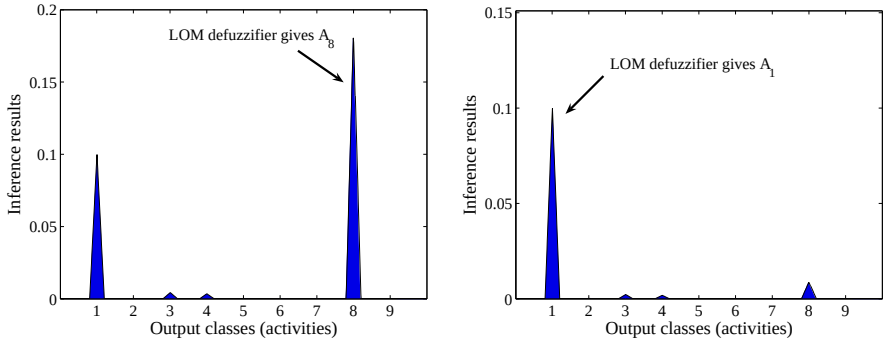
up to 95% of the total inference time [9]. For the activity classification problem, however, Fact 3 suggests that the *largest-of-maximum* (LOM) defuzzification method is more appropriate because it favours the most probable activity to be selected at the output. In addition, as shown in Sec. 7, LOM defuzzification has a much faster execution time than the centroid method on sensor nodes.

5.2 Temporal Order of Basic Operations

In this section, we extend the recognition system for the case of sequences of ordered operations by incorporating time knowledge in the inference process. To illustrate the usefulness of this extension, we return to the example of activity A_1 from Table 2.

Fig. 4 shows one experimental data set collected when performing A_1 during the car assembly trial. The basic operations detected are represented as gray-tone rectangles. The position of the rectangles on the vertical axis gives the operation ID. The width of the rectangles gives the extent of time during which the operations were reported by the corresponding detectors. On the right side of the figure, we list the average confidence of each operation. We remember from Table 2 that activity A_1 consists of operations $O_1, O_2, ..., O_7$, in this order. Analyzing Fig. 4, we can make the following observations:

- The required operations $O_1, O_2, ..., O_7$ are all recognized. However, O_6 has very low confidence (0.07).
- There are recurring insertions, for example O_{29}, with confidence 0.29 on the average. Nevertheless, this does not generate a classification error because



(a) Ignoring time (b) Considering operation sequence

Fig. 5. (a) The insertions observed in Fig. 4 determine the incorrect result A_8 instead of A_1. (b) Using the penalty function T_A for operations appearing in the wrong time order, A_1 is correctly identified.

O_{29} is involved only in A_{11}, together with $O_{27}, O_{28}, O_{30}, O_{31}$, and none of these operations occurs during the experiment.

– There are also insertions of O_{19}, O_{22}, O_{23}, which form the activity A_8-"Mount hood rod". O_{19} and O_{22} have high confidences (0.47 and 0.53). Even though O_{23} has a lower confidence (0.12), this is fuzzified to a high value because O_{23} occurs with low confidence throughout the whole trial (so also in the training set). As shown in Fig. 5 (a), the result of the LOM defuzzification in this case is wrong; the FIS classifies the activity as A_8 instead of A_1.

Time knowledge can help overcome these problems. Let us analyze again Fig. 4. First, we build the unified time interval when a certain operation was recognized (represented as dotted line rectangles). Then, we define the strict temporal order relation $\prec$ between two operations O_i and O_j with their time intervals $T_i = [a_i; b_i]$ and $T_j = [a_j; b_j]$, respectively, as:

$$O_i \prec O_j \iff b_i < a_j \tag{2}$$

The $\prec$ relation is used to identify the operations appearing in wrong order, i.e. the cases when O_i should be executed *after* O_j according to the activity description, but $O_i \prec O_j$ in the experimental data. However, when the unified time intervals T_i and T_j overlap, the operations are *not* considered in the wrong order, so $\prec$ is a strict relation. For example, in Fig. 4, operations $O_1, O_2, ..., O_7$ appear in the right order according to the definition of A_1. In contrast, $O_{19} \prec O_{23} \prec O_{22}$, while the correct sequence for A_8 would be O_{22}, O_{19}, O_{23}. If we filter out such incorrectly ordered insertions, we can prevent errors as in Fig. 5 (a). For this purpose, we need to add to the initial FIS an input variable for each activity class; these variables characterize how well the sequences of operations fit the activity descriptions.

Without the loss of generality, let us consider activity A defined as $A = \{O_1, O_2, ..., O_k\}$ and the maximal sequence of operations from experimental data

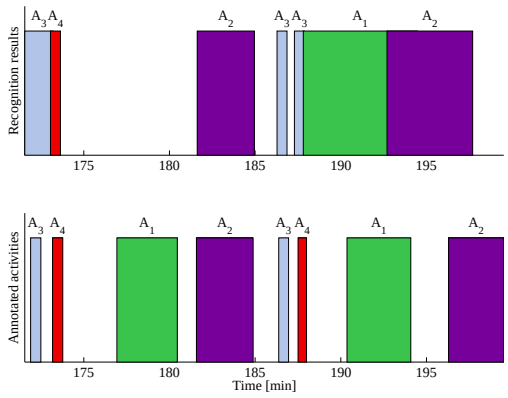


Fig. 6. Snapshot of the recognition results. The upper graph depicts the recognized activities and the lower graph shows the ground truth.

$S = \{O_{i_1}, O_{i_2}, ..., O_{i_l}\}$, with $S \subseteq A$. [4] We define the number of inversions in S with respect to A as:

$$inv_A(S) = \|\{(i_a; i_b) \mid a < b \text{ and } O_{i_b} \prec O_{i_a}\}\| \tag{3}$$

and the number of deletions:

$$del_A(S) = k - l \tag{4}$$

where $\|.\|$ is the cardinality measure.

In other words, $inv_A(S)$ measures the number of pairs of operations in the wrong order and $del_A(S)$ the number of deletions with respect to A. We use the measure:

$$T_A = \frac{inv_A(S) + del_A(S)}{k(k+1)/2} \tag{5}$$

as an input variable to the FIS for each activity A, where S is the maximal sequence included in A from the experimental data. By including both the inversions and deletions, T_A acts as a penalty function for matching the sequence of detected operations to the right activity description. For the example in Fig. 4 and activities A_1 and A_8, we have $T_1 = (0+0)/28 = 0$ and $T_8 = (2+0)/6 = 0.33$, respectively, so T_8 will induce a higher penalty.

The rule base is also extended to take the new inputs into account, so Rule 1 (see Sec. 5.1) becomes:

$$\begin{aligned} \text{IF} \quad & O_{19} \text{ is } High \text{ AND } O_{20} \text{ is } High \text{ AND } O_{21} \text{ is } High \text{ AND } T_7 \text{ is } Low \\ \text{THEN} \quad & A_7 \text{ is } High \end{aligned} \tag{6}$$

The result of the fuzzy inference taking into account the new T_A variables is depicted in Fig. 5 (b). As we can see, the LOM defuzzification yields the correct classification result, A_1.

[4] Insertions of operations not related to A are not taken into account in the input variable for A (T_A), but in the input variables for the other activities, according to their definitions.

6 Results

As explained in Sec. 4, the car assembly trial consisted of 11 complex activities performed 10 times by each of the 2 subjects, so a total of 220 experiments. In this section, we present the performance results of both the normal FIS and the temporal order extension. As a basis for comparison, we refer to the previous work of Amft et al. [3], where the same car assembly trial was analyzed with two methods: *edit distance* and *event histogram*. The activity recognition is performed by processing the continuous flow of events from the detectors through an adaptive search window, sized according to the training data (see [3] for details). Fig. 6 gives a snapshot of the continuous recognition process, where the upper graph depicts the activities reported by the fuzzy inference and the lower graph represents the annotated data (ground truth). We notice that activity A_1 is not recognized at time 179 and A_4 is confused with A_3 at time 188. The remaining activities are correctly reported, within the approximate time intervals given by the search window.

In order to characterize thoroughly the recognition performance, we use the metrics *recall* and *precision*. Fig. 7 shows the recall and precision of the fuzzy-based system, using four-fold cross validation for the selection of training and validation data (the training of the membership functions follows the heuristic method described in Sec. 5.1). The normal fuzzy inference achieves an overall recall and precision of 0.81 and 0.79. The temporal order extension improves the recognition performance to 0.85 recall and 0.85 precision. Fig. 8(a) compares these results to the edit distance (0.77 recall and 0.26 precision) and event histogram (0.77 recall and 0.79 precision) methods from [3]. We can make the following observations:

1. Activities A_6 and A_7 display a recall and precision close to 1, while A_4 and A_{10} have a relatively low recognition performance. This behavior can be explained by analyzing again Fig. 3. We notice the clear separation in the input data for A_6 and A_7, which leads to the high recognition performance. Likewise, we notice the similitude between the input data for A_3 - A_4, which generates the confusion in the recognition results.
2. The normal FIS has the worst performance in the case of A_1 and A_2, which are the most complex activities, each comprising 7 basic operations. The temporal order extension improves considerably the recognition performance of these activities, increasing the recall and precision by 15% and 25% on average. Similarly, the recall and precision of A_{11} (composed of 5 operations) are increased by 10% and 9%, respectively. This shows that analyzing the temporal order is particularly successful in the case of complex sequences of operations.
3. For less complex activities, such as A_8 (composed of only 3 operations), the performance of the temporal order extension may drop compared to the regular fuzzy inference. This is caused by the penalty for deletions introduced via the $del_A(S)$ factor (see Eq. 4), which becomes considerable in the case of a small k (number of operations).

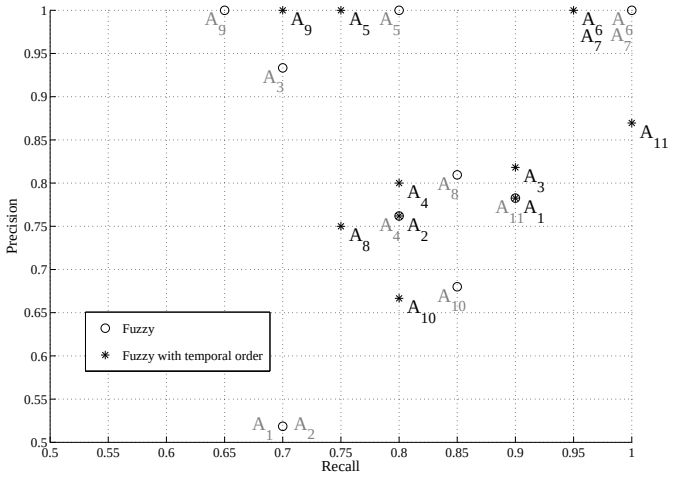


Fig. 7. Recognition performance of the normal FIS and the temporal order extension.

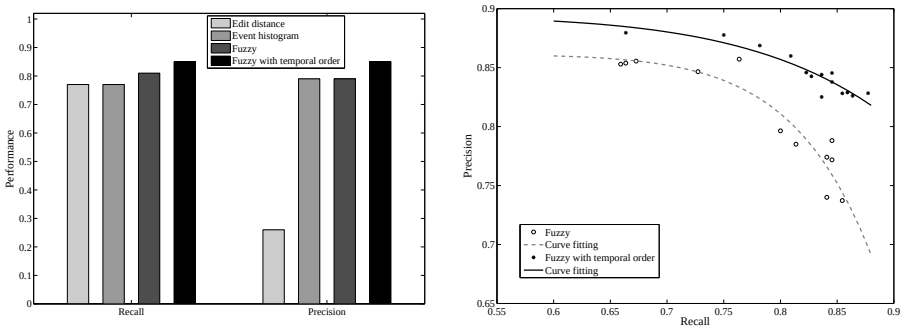
4. The normal fuzzy method outperforms the event distance in terms of precision and has similar performance as the event histogram, with a slight improvement in the recall. The temporal order extension further increases the recognition performance with 4-6% on average. Therefore, the event histogram and normal fuzzy methods are appropriate for activities composed of few basic operations, while the temporal order extension is recommended for more complex activities.

As a final result, we present the trade-off between the overall recall and precision in Fig. 8(b). The trade-off is given by an activation threshold set on the FIS output: if the output is lower than the threshold, we consider the situation as "nothing happening" (the NULL class), otherwise we report the classified activity A_1-A_{11}. In this way, the recognition performance can be adapted according to what is more important from the application perspective: to have a low rate of false positives (good precision) or to minimize the false negatives (good recall).

7 Prototyping

In order to analyze the feasibility of our approach, we implemented the detectors algorithms and the fuzzy inference on a resource-constrained WSN platform. We chose the Tmote Mini sensor node platform (see Fig. 9 (a)) for its compact, industry-standard miniSDIO form factor. The Tmote Mini integrates an MSP430 microcontroller and a TI/Chipcon CC2420 low-power radio in a single one square inch package.

The implementation of the detectors covers two main tasks: (1) to spot relevant sensor data segments in a continuous data flow and (2) to classify those segments as one of the detector's events. To this end, the detectors implement a feature similarity search (FSS) algorithm [2, 3]. In the first phase, the data is



(a) (b)

Fig. 8. (a) Comparison of the overall recall and precision achieved by four recognition methods: edit distance, event histogram [3], normal fuzzy and the temporal order extension. (b) Tuning the performance of the fuzzy-based methods.

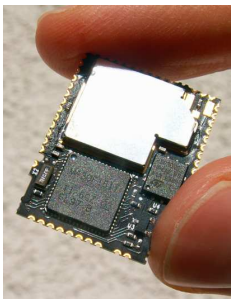
segmented into windows of a detector-dependent size with a step size of 250ms. In the second phase, the algorithm extracts a detector-dependent number of features from the data segments, and forms the feature vector. The similarity (measured by the Euclidean distance) of this feature vector to a set learned from user-annotated data determines the classification decision.

Let us choose detector D_{10} (front door) for example. D_{10} implements 10 features (length, beginning, end, total difference from mean on 5 partial windows, variance) from acceleration data sampled at 50Hz using an ADXL330 3-axis accelerometer. The complexity of the FSS algorithm scales with the window length (200 samples), the number of features and the number of classes to be distinguished. The total execution time measured on the sensor node is 28.8ms, more precisely 26.8ms for the feature computation phase and 2.0 ms for the classification phase. The algorithm requires 1.3kB of RAM and 1.9kB of FLASH.

The implementation of the fuzzy inference covers the main components described in Sec. 5:

- *Fuzzification*. For computational simplicity, we use trapezoidal membership functions. The fuzzification is computational oriented due to the limited RAM available. To reduce the computational overhead, the maximum fuzzified value is scaled to a power of 2.
- *Inference*. The rules base is stored in a compact, binary form. We use *sum-product* inference as it proves more robust to deletions than max-min method.
- *Defuzzification*. The result of the rule evaluation is defuzzified using the *largest-of-maximum* (LOM) method.

Fig. 9 (b) presents the performance parameters of our implementation, for both the normal FIS and the temporal order extension. We can conclude that a total execution time of less than 12ms is feasible on the MSP430 microcontroller. Moreover, the defuzzification time is almost negligible compared to fuzzification and inference. This result highlights the considerable benefit of using LOM instead of centroid defuzzification (for comparison, see [9]). The rightmost column



FIS type	Inputs	$\mathbf{T}_{fuzz}$	$\mathbf{T}_{inf}$	$\mathbf{T}_{defuzz}$	$\mathbf{T}_{total}$	RAM
Normal	31	4.21	4.45	0.07	8.73	288
Time order	42	5.88	5.97	0.07	11.92	344

(a) (b)

Fig. 9. (a) Tmote Mini sensor prototype platform. (b) FIS implementation details: execution time and memory footprint.

of Fig. 9 (b) lists the amount of RAM required by the FIS, out of 10kB available. The code memory footprint amounts to 490 bytes, out of 48kB available. These figures confirm that a very lightweight implementation of the FIS is feasible.

A final important aspect is the numerical accuracy of the FIS running on the node. In order to quantify the impact of integer computation over the fuzzy inference, we input the experimental data sequentially to the sensor node via the serial port and collect the classification results. The overall error is 1.11%, which is a value within tolerance limits considering the impracticality of floating point computation on the node.

8 Discussion

The results presented so far give an evaluation of the activity recognition system for a particular experiment and the associated data set. In what follows, we provide a more general discussion of the relevant performance factors, with respect to the particularities of fuzzy-enabled WSN.

Distributed recognition architecture. A distributed architecture is required when implementing activity recognition on resource-constrained devices because (1) a powerful central node may not be available and (2) the WSN does not have the capacity of transporting the raw data to a central fusion point. However, as we showed in the previous sections, it is essential that the distributed recognition system considers the whole chain from sensor data to activity classification, with respect to both performance evaluation and implementation.

Recognition performance. Fuzzy logic proves to be robust to both unreliable sensor data and wireless link failures, which cause insertions and deletions. The analysis of the temporal order improves the recognition performance, especially for the activities with many operations executed in a clear sequence. However, the performance remains limited for the activities that are either very similar or have large overlapping areas in the input data. Two additional factors can further affect the overall performance. The first concerns filtering the

NULL class, which makes a trade-off between precision and recall. The second is related to the segmentation problem: the recognition system running online cannot know the exact extent of the activities, so only the events within a certain time segment are detected. Consequently, the FIS may operate with incomplete inputs and yield an incorrect result.

Energy consumption. There are two main factors that influence the energy consumption. The first factor is the power dissipated on the actual sensors (e.g. accelerometers), which is independent of the activity recognition system. The second important factor is the wireless communication. The distributed recognition architecture keeps the network overhead to a minimum, by communicating only the spotted events. The number of messages sent by a detector node D_i is proportional to: (1) the amount of events D_i can detect, i.e. $\|\mathcal{E}_i\|$, (2) the frequency of occurring of each event E_{ij} and (3) the rate of insertions. The number of messages received by a node depends on its role in the data fusion process, which can be: (1) simple detector, (2) combining events into basic operations or (3) final activity classifier. It follows that the latter two roles should be assigned to nodes that have fewer (or none) events to detect.

Memory and computational overhead. The implementation on sensor nodes shows that the memory requirements are very low and both the event detection algorithms the fuzzy inference achieve execution time of approximately 40ms. Although the major concern in WSN is to keep the communication overhead to a minimum, fast execution times are equally important in our case, because three demanding tasks compete for CPU utilization: the sensor sampling, the medium access protocol and the activity recognition.

9 Conclusions

Wireless sensor nodes worn on the body and placed on the tools can recognize the activities of the users and assist them at work. We presented a distributed architecture that uses fuzzy logic for reliably detecting and classifying the user activities online. For performance evaluation we used the experimental data from 12 sensor nodes equipped with 3D accelerometers, obtained during a car assembly trial within an industrial setting. The fuzzy system achieved an overall recall and precision of 0.81 and 0.79. An interesting extension was to include temporal order knowledge about the sequences of operations into the fuzzy inference, which improved the recognition performance to 0.85 recall and 0.85 precision. In order to analyze the feasibility of our approach, we implemented both the detection algorithms and the fuzzy logic engine on the Tmote Mini sensor platform. The results showed that our method can run online, with execution times of approximately 40ms, memory overhead in the order of 1.5kB and an overal accuracy loss of 1.11%.

Acknowledgements. This paper describes work undertaken in the context of the SENSEI project, "Integrating the Physical with the Digital World of the Network of the Future" (www.sensei-project.eu), contract number: 215923.

References

1. WearIT@work Project. http://www.wearitatwork.com.
2. O. Amft, H. Junker, and G. Tröster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *International Symposium on Wearable Computers (ISWC)*, pages 160–163, 2005.
3. O. Amft, C. Lombriser, T. Stiefmeier, and G. Tröster. Recognition of user activity sequences using distributed event detection. In *European Conference on Smart Sensing and Context (EuroSSC)*, pages 126–141, 2007.
4. Matthew Brand. The "inverse hollywood problem": From video to scripts and storyboards via causal analysis. In *AAAI/IAAI*, pages 132–137, 1997.
5. M. Marin-Perianu et al. Decentralized enterprise systems: A multi-platform wireless sensor networks approach. *IEEE Wireless Communications*, 14(6):57–66, 2007.
6. F. Gemperle, C. Kasabach, J. Stivoric, M. Bauer, and R. Martin. Design for wearability. In *International Symposium on Wearable Computers (ISWC)*, pages 116–123, 1998.
7. Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):852–872, 2000.
8. S. Li, Y. Lin, S. H. Son, J. A. Stankovic, and Y. Wei. Event detection services using data service middleware in distributed sensor networks. *Telecommun Syst*, 26(2):351–368, December 2004.
9. M. Marin-Perianu and P. J. M.Havinga. D-FLER: A distributed fuzzy logic engine for rule-based wireless sensor networks. In *International Symposium on Ubiquitous Computing Systems (UCS)*, pages 86–101, 2007.
10. V. Osmani, S. Balasubramaniam, and D. Botvich. Self-organising object networks using context zones for distributed activity recognition. In *International Conference on Body Area Networks (BodyNets)*, 2007.
11. J. B. Predd, S. R. Kulkarni, and H. V. Poor. Distributed learning in wireless sensor networks. *IEEE Signal Processing Magazine*, 23(4):56–69, July 2006.
12. T. J. Ross. *Fuzzy Logic with Engineering Applications*. Wiley, 2004.
13. V. Saligrama, M. Alanyali, and O. Savas. Distributed detection in sensor networks with packet losses and finite capacity links. *IEEE T Signal Proces*, 54(11):4118–4132, November 2006.
14. V. N. S. Samarasooriya and P. K. Varshney. A fuzzy modeling approach to decision fusion under uncertainty. *Fuzzy Sets and Systems*, 114(1):59–69, 2000.
15. M. Stäger, P. Lukowicz, and G. Tröster. Power and accuracy trade-offs in sound-based context recognition systems. *Pervasive and Mobile Computing*, 3(3):300–327, June 2007.
16. T. Stiefmeier, C. Lombriser, D. Roggen, H. Junker, G. Ogris, and G. Tröster. Event-Based Activity Tracking in Work Environments. In *International Forum on Applied Wearable Computing (IFAWC)*, March 2006.
17. T. Wang, Y. Han, P. Varshney, and P. Chen. Distributed fault-tolerant classification in wireless sensor networks. *IEEE J Sel Area Comm*, 23(4):724–734, April 2005.
18. Christopher R. Wren, David C. Minnen, and Srinivasa G. Rao. Similarity-based analysis for large networks of ultra-low resolution sensors. *Pattern Recogn.*, 39(10):1918–1931, 2006.