Opportunistic Sensing in Train Safety Systems

Hans Scholten and Pascal Bakker
Pervasive Systems
University of Twente
Enschede, the Netherlands
hans.scholten@utwente.nl, aboe@aboe.nl

***Abstract*—Train safety systems are complex and expensive, and changing them requires huge investments. Changes are evolutionary and small. Current developments, like faster - high speed - trains and a higher train density on the railway network, have initiated research on safety systems that can cope with the new requirements. This paper presents a novel approach for a safety subsystem that checks the composition of a train, based on opportunistic sensing with a wireless sensor network. Opportunistic sensing systems consist of changing constellations sensors that, for a limited amount of time, work together to achieve a common goal. Such constellations are self-organizing and come into being spontaneously. The proposed opportunistic sensing system selects a subset of sensor nodes from a larger set based on a common context. We show that it is possible to use a wireless sensor network to make a distinction between carriages from different trains. The common context is acceleration, which is used to select the subset of carriages that belong to the same train out of all the carriages from several trains in close proximity. Simulations based on a realistic set of sensor data show that the method is valid, but that the algorithm is too complex for implementation on simple wireless sensor nodes. Downscaling the algorithm reduces the number of processor execution cycles as well as memory usage, and makes it suitable for implementation on a wireless sensor node with acceptable loss of precision. Actual implementation on wireless sensor nodes confirms the results obtained with the simulations.**

Keywords-opportunistic sensing; wireless sensor network; context awareness; activity recognition; train safety.

I. Introduction

In this paper, we show how opportunistic sensors, which use a common pattern of movement, are deployed to detect the composition of a train. The problem was introduced in [1] and [2] describing how wireless sensor networks enhance safety in moving linear structures such as trains. The wireless sensor network is a subsystem of a railway safety system to monitor the initial composition of a train and to detect changes in composition once the initial composition has been established. Movement as a discriminating factor for context awareness in wireless sensor networks has been described earlier [3] [4], but not for trains.

A. Railway Safety Systems

Currently, the rail system in Europe consists of multiple different safety systems: the countries still rely mostly on a country-specific system. International European trains have multiple systems onboard to be able to pass borders and enter another country and hence another safety system. Projects in different European countries are working on a uniform system [5], the European Rail Train Management System (ERTMS), under supervision of the European Railway Agency [6]. The eventual goal is the adoption of the system in all participating countries. The draft for the latest version of ERTMS is known as ERTMS level 3. ERTMS level 1 and level 2 still consider track sections instead of trains as sections. Like the old system, a track is divided in fixed length sections. When a train enters a section, called a block, no other train is allowed in. The length of a block is determined on the worst case, considering length, weight and maximal speed of trains. This implies that a small light train takes as much space as a large and heavy train. ERTMS level 3 considers a train as a moving block, keeping a safety zone around the train in accordance with its length, weight and speed.

By representing trains as moving blocks, short and light trains will form smaller blocks than heavier trains, which have a much longer brake path. This allows more trains on the same track when compared to a system that uses static safety zones. Level 1 and level 2 ERTMS systems use a straightforward approach to detect whether a block is occupied. When a train enters or leaves a block, its axles are counted. When the number of axles is equal at both counts, the block is empty. If not, (part of) the train still occupies the block. ERTMS level 1 uses signals located at the side of track for the indication of the availability of the next zone, whereas ERTMS level 2 communicates this information by radio using GSM-Rail (GSM-R). ERTMS level 3, in contrast to levels 1 and 2, does not rely on a trackside signaling system. Instead, it uses an onboard safety system that checks and controls the safety zone of the train. It is imperative that a train can guarantee its integrity, i.e. its composition is known and no carriages are lost. As no trackside backup system is available, a dangerous situation occurs when a train loses a carriage. The next train may crash into the lost carriage if it is not informed in time. A system that accurately monitors in real time the integrity of the composition is a crucial part of ERTMS level 3. Leaving a carriage behind is not as farfetched as it sounds. It happens all the time, for example, in switchyards or at stations. In

Figure 1. Freight carriage connection

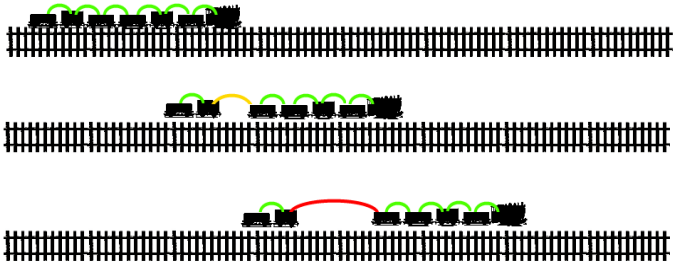


Figure 2. Beacons, single train

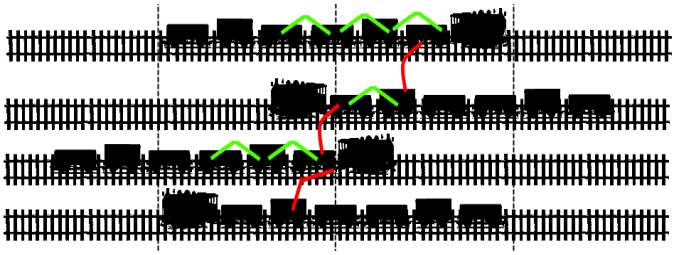


Figure 3. Beacons, multiple trains

any case, the train must know its composition and report changes. Passenger trains are equipped with numerous wired links between wagons, which can be used to monitor its carriages.

Freight trains, however, are normally not equipped with electrical connections between carriages (see Figure 1). When a train rides at night in the Netherlands, the engineer manually puts up a light at the last carriage, since the carriages lack all electrical provisions.

A system for guaranteeing freight train integrity should preferably operate wirelessly. Because carriages are scheduled for maintenance once or twice every year, the new system should be able to run for at least a year without human intervention. The system has to operate in all kinds of weather and should be able to withstand dust, water and other kinds of abuse.

B. Opportunistic Sensing

"Opportunistic sensing is seen as a way to gather information about the physical world in the absence of a stable and permanent networking infrastructure." (Opportunity Workshop at Ubicomp 2010, Copenhagen, Denmark). The absence of a stable and permanent networking infrastructure dictates that collected information is either processed and acted upon inside the network by opportunistic collections or clusters of nodes [7], or the information is preprocessed and stored inside the network until there is an opportunity to forward it outside the network, as is the case in delay tolerant networks [8] - [18].

Opportunistic sensing and opportunistic networking is often associated with human-centric ubiquitous systems, such as in crowd sourcing and participatory sensing applications [19] - [21] or are focusing on human activity recognition [22] - [24].

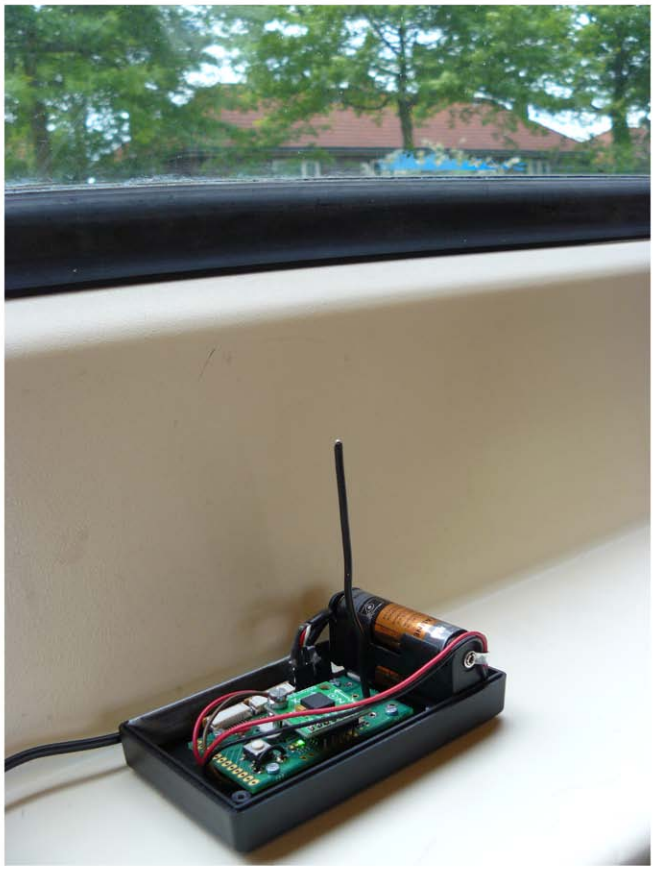
C. Opportunistic Sensing for Train Safety Systems

The proposed method of using opportunistic sensing with movement as discriminating factor seems overkill, where a simple detection system based on radio beacons would be sufficient. Figure 2 illustrates the principle. Once the decoupled carriages at the back of the train are out of radio range, an alarm will be raised. The system works for a single train with no other trains in range. This is not the case at stations or switchyards, or, generally, when a train passes an other on an adjacent track and comes in range of the other train's beacons, as shown in Figure 3. The system would still work if the train had knowledge about its composition. Periodically checking all beacons in range would detect any change in the composition of the train. Because the ID of the beacons in sight are not known a priori, nor the mapping of IDs to carriages, nor carriages to trains, there is no way to discriminate carriages in different trains in close proximity. Additional measures are needed to discriminate between trains.

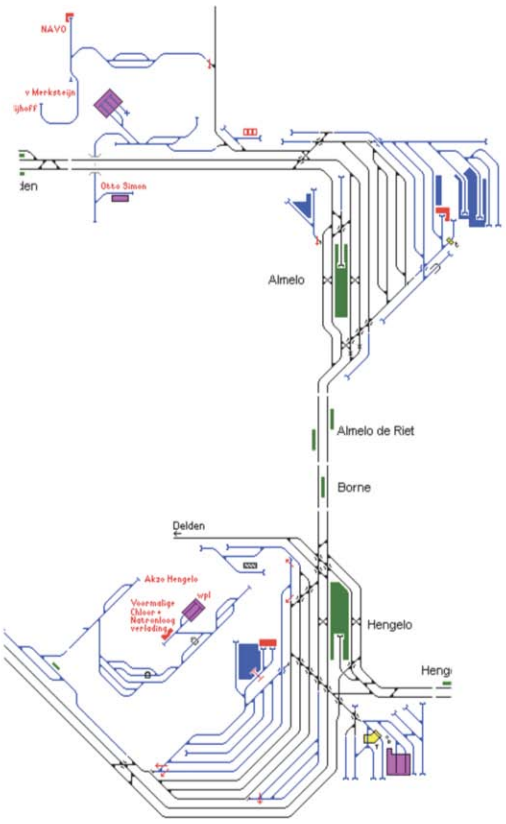
This paper investigates movement, or more precise acceleration, as a way to distinguish carriages in different trains and determine a train's composition. In the remainder of this paper, we will discuss the collection and analysis of the data sets to be used in the simulations, the simulation itself and the implementation on wireless sensor nodes respectively, followed by a discussion of the results.

II. Data Collection and Analysis

As explained in the introduction, movement is used as discriminating feature. However, motion is multi-dimensional and probably too complex to use in wireless sensor nodes in an energy efficient manner. Not only processing power and memory usage might be issues, also running complex algorithms for longer periods of time consumes large amount of energy, negatively influencing the operational time of the system. Because of all these limitations, it might seem advantageous to deploy a centralized solution with simple sensors and a server doing all the processing. Raw data is sent directly to the server, where the correlation of all carriages is calculated. While this saves on processing and



(a) Wireless sensor node



(b) Railway track

Figure 4. Data collection

memory on the sensor node, data sent by radio will be much higher than in the decentralized case, where processing takes place on the node. The increase in power consumption by the radios exceeds the energy savings due to less processing. Still, the sensors would depend on batteries for their energy provisioning. The result would be less operational time.

For the distributed, wireless sensor network version, several measures can be taken to enable implementation on nodes:

- Minimize the time the algorithm executes,
- Simplify the algorithm as much as possible, and
- Simplify the input data for the algorithm.

Under normal conditions, a train's composition will not change while underway and moving. If a change occurs under these circumstances, it will be accidentally. We have seen that a system with radio beacons works to detect these changes, but only under the condition that the composition of the train is known a priori. Such a system would be much simpler and more reliable than any implementation with accelerometers. It is also much more energy efficient than running complex algorithms on a wireless sensor node.

A train run has four distinct phases:

- The train is standing still,
- The train starts moving and accelerates,
- The train has a steady speed, and
- The train decelerates and stops.

During the first phase, standing still, nothing changes and sensor nodes sleep. When the second phase begins, the sensor nodes wake up and the train's composition is determined. Once the composition is known, radio beaconing detects any changes from the initial composition. At the end of phase four, the train has come to a complete standstill, the nodes go to sleep again to preserve energy. The algorithm that runs on the sensor nodes reflects the four phases of a train run: sleep, determine composition, detect changes from initial composition, and sleep.

In the following, we will focus on the most complex second phase, discriminating carriages in different trains and

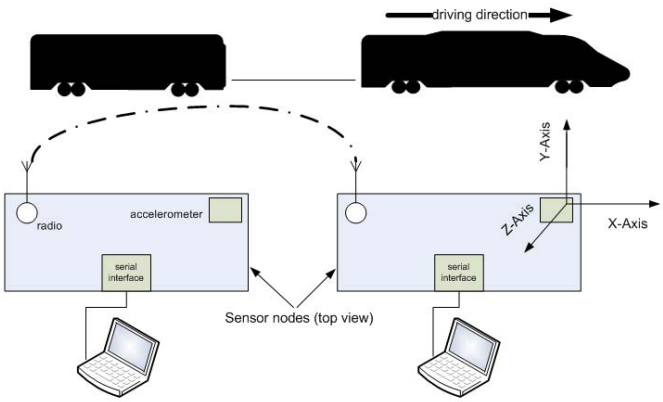


Figure 5. Collecting data sets

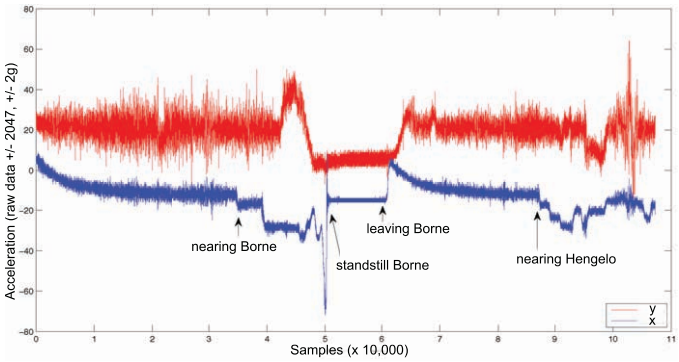


Figure 6. Accelerometer x- and y-axis

determination of a train's composition.

A. Collecting the Data

To check the feasibility of the proposed algorithm a number of simulations with realistic data sets are executed. The data sets consist of sensor data recorded on a track featuring multiple stops and curves (see Figure 4b), resulting in a wide variety of data. Figure 4a shows the wireless sensor node that is used to sample the data, consisting of an Ambient muNode 2.0 [25] provisioned with an STMicroelectronics LIS3LV02DQ accelerometer. The maximum sample rate of this sensor is 640 Hz, but the combination of hardware and software limits the sample rate to 160 Hz. The sensor nodes are aligned with the horizontal x-axis in the driving direction, the y-axis in the horizontal sideways direction and the z-axis in the vertical direction.

The data sets are recorded with two sensor nodes in two distinct carriages in the same train during several runs (see Figure 5). The data sets from sensors in different runs of the same train are used to simulate different trains. The sensor data from each sensor node is recorded real-time with laptops. All data are time stamped by the sensor nodes and the sensor nodes are connected via their radios to synchronize them.

B. Analyzing the Data

Figure 6 depicts raw data with a sampling frequency of 155 Hz of a journey between two stations with one station in between. The train is already moving when the graph begins. The two graphs are shifted up (y-axis) and down (x-axis) for a better overview.

The bottom graph shows the x-axis, the driving direction, and illustrates changes in speed of the train. After the initial acceleration (not shown) the speed settles to a constant value (acceleration is 0). Just before the middle station, the train decelerates in stages and comes to a standstill with a shock. Leaving the middle station the train accelerates to a constant speed.

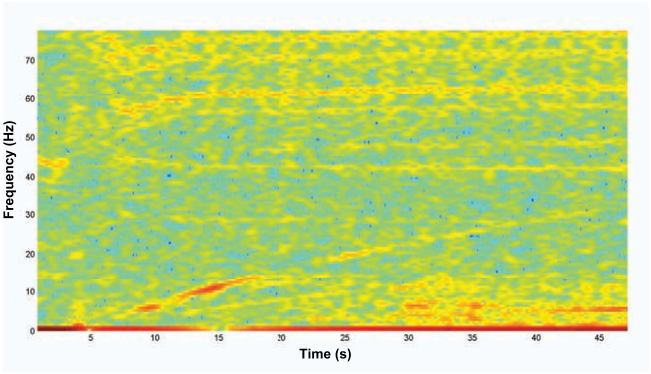
The top graph depicts the sensor's y-axis, movement sideways, and just before reaching the middle and the terminal station the train is crossing a switch changing tracks. In between the graph exposes short sideway movements of the train inside the track. Even when standing still, sideway movement is detected, though with a smaller amplitude than when the train is moving. This is contributed to noise the accelerometer generates. This noise is also present in the x and z direction and must be filtered out before the sensor data can be processed further.

Not shown in the graph is the sensor data of the z-axis. This data did show clearly when a train is moving or not. Further analysis learned that the data did not have enough features to make a distinction between trains when they start moving at the same time. In the remainder of this section we will focus on movement in the other two directions.

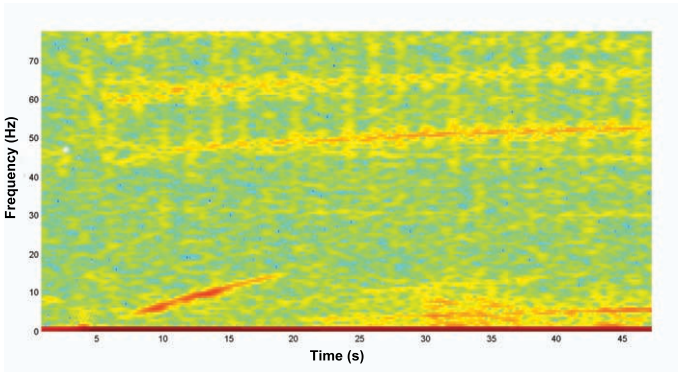
Figures 7 and 8 show frequency spectrum diagrams for sampled data from the x- and y-axes of the accelerometers in carriages over a time period from 0 till 47 seconds. Color is used to depict the intensity of frequencies in the sampled data. Going from low to high intensities, the colors blue, green, yellow and red are used.

Figure 7 shows the frequency spectrum for the x-axis data of two carriages in the same train starting to move. Only frequencies lower than the sampling frequency/2 are considered: 0 to 80 Hz. The spectra seem similar in the low frequencies, but quite different in the high frequencies. The latter can be contributed to the already mentioned noise of the accelerometers. The spectra of two carriages in two different trains on the same section of the railway track (not shown) differ in all frequencies, but the difference in the lower frequencies is smaller than in the higher ones. Closer inspection of the spectra and a preliminary correlation of sensor data in different frequency bands indicates that the most usefull discriminating features are present in a frequency band of 0 to 2Hz. Before the sensor data are further processes a high-off filter is applied with a cut-off frequency of 2 Hz.

Figure 8 shows the spectrum for movement in the y

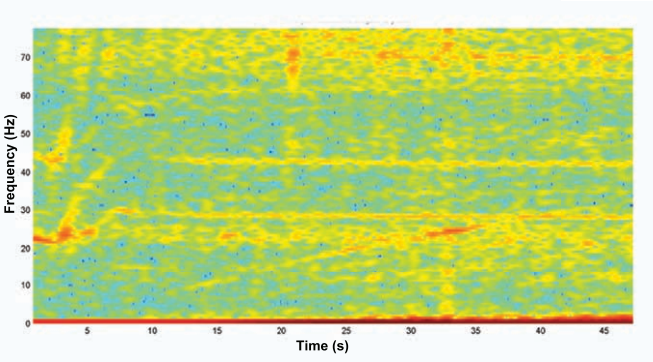


(a) Spectrum carriage 1, train 1, x-axis

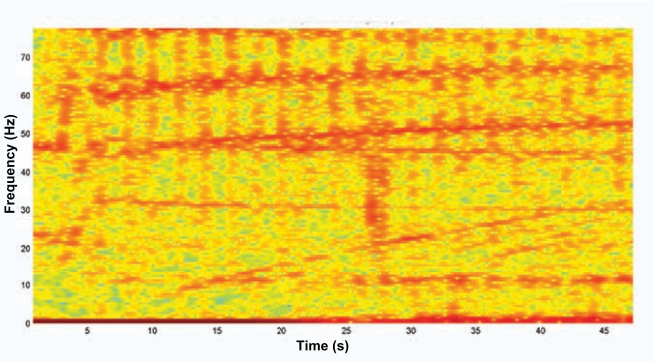


(b) Spectrum carriage 2, train 1, x-axis

Figure 7. Frequency spectrum x-axis of two carriages in the same train



(a) Spectrum carriage 1, train 1, y-axis



(b) Spectrum carriage 2, train 1, y-axis

Figure 8. Frequency spectrum y-axis of two carriages in the same train

direction of two carriages in the same train. Though there are similarities in both spectra, they are much less than for the x direction. A complication is that the spectrum of a carriage in a different train (not shown) is similar to those shown.

Analysis of the collected movement samples shows that from all data in the three axis x, y and z, only the sensor data of the x-axis, which is the direction the train rides, are usefull. The data are highly polluted by noise, visible in the spectrum in the higher frequencies. The noise is generated by high frequency movements of the train and by the accelerometer and must be filtered before the data can be used. We also found that all discriminating features are in a frequency band of 0 to 2Hz, the relatively slow movements of the train.

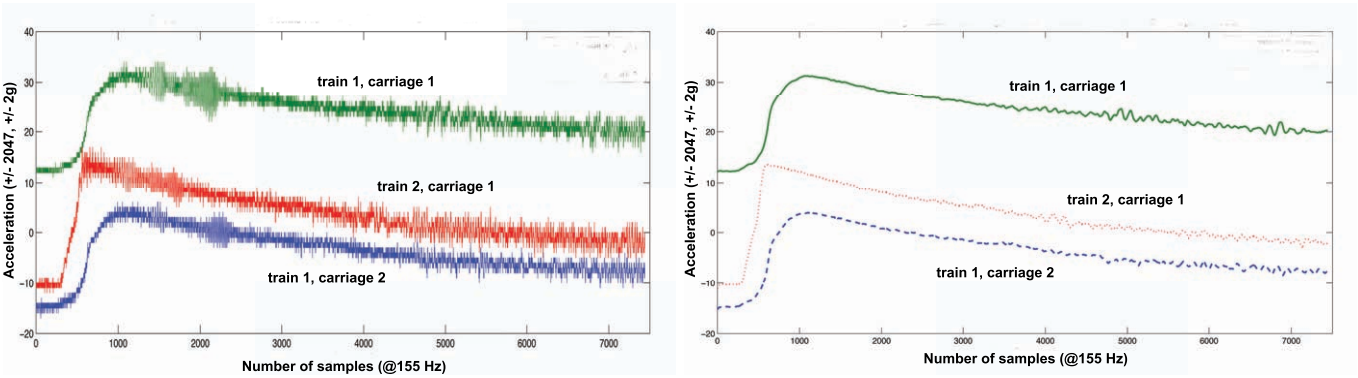
C. Preprocessing the Data

To filter the data, a second order Butterworth low pass filter is used. This choice is made because this type of filter is tested and runs on the used wireless sensor nodes. Figure 9 shows the results for the x-axis data samples of three carriages. The top and the bottom graph are from carriages in the same train. The graph in the middle is from a carriage in a different train. The graphs are synchronized, i.e., they are shifted in time so they show the trains starting at exactly the same time. In practise it will rarely happen that two trains in communication range will accelerate at exactly the same time. This is even discouraged, as it will cause peaks of electricity consumption in the power grid. Their is a clear difference between the two trains. The graph of the second train is steeper as a result of a higher acceleration.

The data sampling rate in the original data set as shown in the graphs is 155 samples per second. Because the data is filtered at 2 Hz, this high sampling rate is clearly overkill and can be reduced significantly in the final implementation. A frequency of 35 samples per second gives the same results as before. We did not test lower sampling rates, although this might have been better to reduce the CPU and memory consumption.

D. Data Correlation

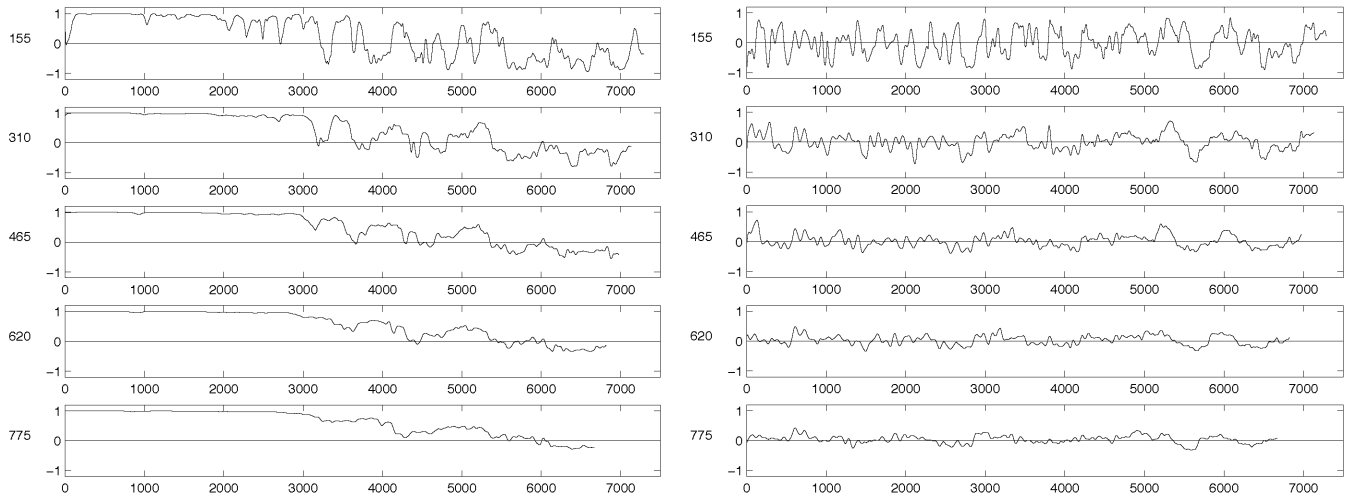
The last step in the algorithm is to check whether two carriages share the same context by way of correlating the data. The correlation between two nodes is done by



(a) Unfiltered data

(b) Filtered data

Figure 9. X-axis acceleration data from three carriages



(a) two carriages in the same train

(b) two carriages in different trains

Figure 10. Correlation of two carriages with varying window sizes

using the Pearson product-moment correlation coefficient. The Pearson correlation is +1 if there is a perfect positive linear relationship, and -1 if there is a perfect negative linear relationship. If the value is 0 there is no relationship. The closer the coefficient is to +1 or -1, the higher the relationship between the nodes. The correlation process uses a sliding window over which the data are compared. A wider window normally leads to a more precise result, but also takes longer to produce this result. A smaller window gives a better reaction time, but the result is less reliable. So a trade-off has to be made.

Figure 10a gives the correlation results for two carriages in the same train with window sizes varying from 155 to 775 samples, corresponding with 1 to 5 seconds. The measured time frame is just over 7000 samples, or 45 seconds. Remember that the detection of the composition of the train only takes place during the first seconds after the train starts moving. Once the composition is known, it suffices to periodically test the presence of the initially found carriages by means of the radio beacons. The composition must be known within the first seconds of a ride. Therefore, the correlation results after 2000 samples, or approximatelly 13 seconds will be ignored. Figure 10a shows that over the time frame of the first 2000 sample a window size of 155 samples, or 1 second, is enough. The Pearson correlation coefficient is already very close to +1. However, the algorithm must also avoid false positives: carriages in other trains that are seen as carriages of the same train. Thus, a carriage in a different train has to be positively identified as such. This translates in a correlation coefficient of 0 for two carriages in different trains. Figure 10b shows that a window size of 155 samples is not enough. The correlation coefficient alternates between positive and negative values and is nowhere near a constant value of 0. To distingish two carriages in different trains a minimum window size of 620, or better 775 samples is needed.

To find the composition of a train, the correlation algorithm not only has to identify carriages in the same train, but also, to avoid false positives, identify carriages in other trains. Identifying carriages in the same train is very fast and only takes 1 second. However, identifying carriages in other trains takes 5 seconds.

III. Implementation

In the following, some implementation details will be discussed. All previous simulations are done with Matlab [26]. They show that identifying carriages in the same or different trains is feasible. They do not show that the algorithms to identify carriages will execute on a wireless sensor node. Matlab runs on powerfull computers with sufficient resources, while wireless sensor nodes are resource lean. Before the algorithms are suitable to run on the nodes they must be optimized, taking into account limitations such as CPU power, CPU speed, available memory and energy consumption.

A. Optimizations

The Matlab simulation takes an approach where everything is centralized. It assumes that the identification algorithm executes in one central place and that all data are available when needed. In reality this is not the case, the algorithm and data are distributed over the wireless nodes. Every carriage only has its own data, but to correlate its movement with that of its neighbors, it needs their movement data as well. It will be clear that the correlation of two neighbors only needs to be calculated once. This calculation can take place at either one of the neighbors. This process is optimized by dynamically forming master/slave pairs. The slave sends its data to the master and the master calculates the correlation. When the calculation has finished, the master sends the correlation results back to the slave. For every master/slave pair the movement data are communicated once and the correlation calculations is performed once, thus reducing both bandwidth and execution load. Which of the neighbors is master and which one slave is not essential. After each calculation the neighbors turn role to balance the energy consumption in the nodes more evenly.

The next step is optimization of the correlation algorithm. In its original form, all data in the sliding time window are used in the calculation [27]. When the window moves, the oldest data is replaced by new data, while all other data in the middle of the window stay the same. Marin-Perianu et al. [4] propose an optimization to this correlation calculation algorithm. The proposed algorithm stores intermediate values that can be used in the next calculation. This reduces the amount of calculations necessary for the computation of the correlation coefficient in the next window at the cost of a slightly increased memory usage. One disadvantage is, when running on small devices, the accumulation of rounding errors. The wireless sensor nodes execute their calculations with a limited number of bits and thus with limited precision. Every time a previous intermediate result is used, its rounding error is added to the rounding error of the current calculation. In due course this rounding error accumulates in unacceptable error margins, giving the wrong results. This can be counteracted by periodically resetting the intermediate results and start with a ”fresh” sliding window. Because in our case the algorithm runs for a limited time - only the first 5 seconds-, the error is small and within acceptable margins. Comparing the algorithm as run on the node with the Matlab version, no differences were found in the time frame of 5 seconds the correlation takes.

One more change in the original Matlab routines must be made before they can be implemented. The Matlab versions of the high-off filter and correlation algorithm are based on calculations that use floating point numbers. Using floating-point calculations on the wireless sensor nodes would stress the CPU unacceptably. Fixed-point calculations are better, though at the cost of possible loss of precision. Errors in rounding results would accumulate and might lead to significant deviations from the desired results over time. Figure 11 shows the difference between (Matlab version) floating point and fixed point calculation for the correlation algorithm. The deviation starts to become evident 15 seconds. Since the correlation algorithm to determine the train composition takes place in the first 5 seconds, replacing floating point by fixed point calculations does not influence the end result. The same results are observed for the high-off filter.

B. Timing and Memory Usage

Running the composition algorithm is distributed over all carriages of the train. The carriages are split in master/slave pairs, where each pair calculates the correlation between the master and the slave. The actual correlation calculation is done by the master. Before this calculation begins, the input sensor data is filtered. Master and slave filter their own data. When ready filtering the data, the slave sends its filter output to the master, after which the master correlates. The execution time depends on the size of the sliding window size of the correlation algorithm. Figure 12 shows the execution times for master and slave for increasing window sizes from 35 to 175 samples. Because the final sample rate implemented on de wireless sensor nodes is 35 samples per second, this correspondents with window sizes from 1 to 5 seconds. For a window size of 5 seconds, the execution times for master and slave are 171.9 msec and 85.9 msec respectively. The routines are executed uninterrupted and do not show significant deviations. Some of the ”extra” time the slave has, is used to assemble the filter data into packages to be sent to the master. The number of calculations that can be executed is 1000 / 171.9 is 5.8 per second.

The amount of memory that is needed for running the algorithm on a wireless sensor node depends on the number of neighbors it sees. With every neighbor the node will

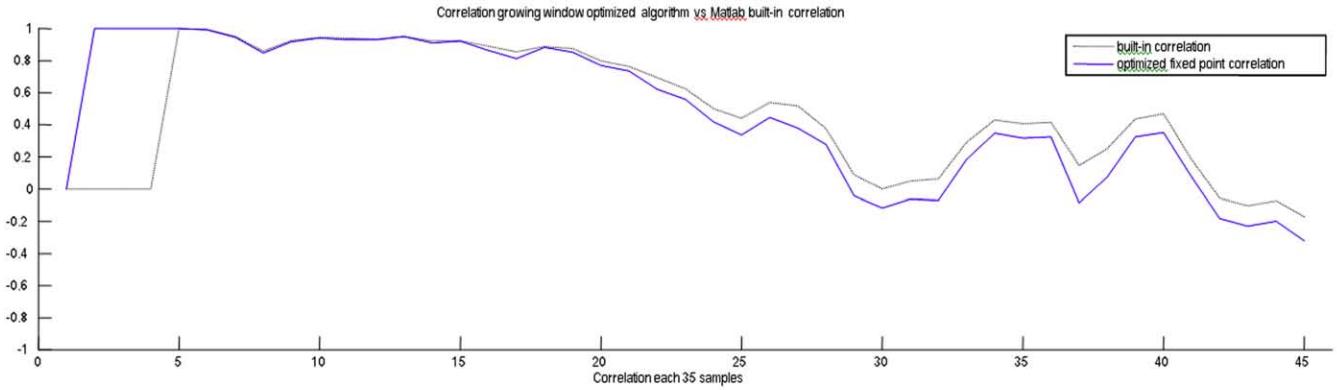


Figure 11. Comparison of floating point and fixed point calculations

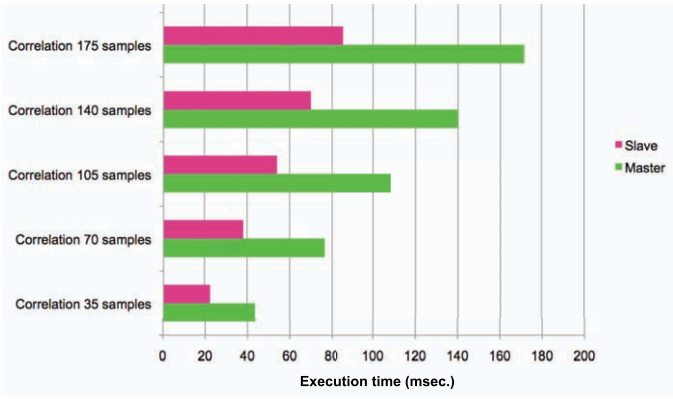


Figure 12. Execution times on a muNode with MSP430 microcontroller

Data structure	Size (bytes)
Dataset slave 175 samples	350
Timestamp start	2
Master mean data	4
Slave mean data	4
Master squares data	20
Slave squares data	20
Master sum data	20
Slave sum data	20
Sum products data	20
Total	460

Table I
MEMORY CONSUMPTION PER MASTER/SLAVE PAIR

for a master/slave pair. Table I summarizes the memory consumption per master/slave pair. Besides memory to store the sample data for one sliding window, the node stores intermediate results of the correlation calculation. The total amount of bytes is 460 per master/slave pair.

A standard Ambient Systems muNode 2.0 has 10kB of RAM available. For the normal operation of the node 2kB has been reserved, which leaves 8kB for the correlation algorithm. Given the memory consumption of 460 bytes for one master/slave pair and the availability of 8192 bytes, a node can store up to 16 master/slave pairs in memory.

IV. CONCLUSION

Train safety systems are going through a transition at the moment. Traditional safety systems depend on infrastructure-based sensors to detect the whereabouts of a train. These systems have serious drawbacks. It is a static infrastructure, where the track is divided in sectors, or blocks, of equal length. The length of a block is defined worst case, by the fasted, heaviest and longest train possible. As a result, the rail infrastructure is used far under its real capacity. This situation is complicated by the fact that European countries have their own safety system. Border crossing trains must have provisions on board for all safety systems on their route, which can accumulate to up to seven systems. The European countries have decided to develop a new system (ERTMS), where the safety system is not in the infrastructure, but on board the trains. One of the safety subsystems is responsible for the integrity of the train. It checks the composition of the trains and reports and changes of the initial composition.

In this paper, we have shown that such a subsystem can be implemented with a wireless sensor network. This network is deployed as an opportunistic sensor system that makes a selection out of a much larger set based on a common context. In this case, motion information is used to distinguish carriages belonging to different trains. The motion information consists of data obtained by accelerometers attached to individual train carriages. This data stream then was analyzed with Matlab on a PC to extract the best possible features to discriminate carriages. While sideways motion gives information on track changes and vertical movement indicates the quality of a track, the best information for our purpose is movement in the direction

of travelling (x-axis). A spectrum analysis learns that not all frequency components in the sampling data are equally useful: only those below a frequency of around 2 Hz are significant to separate carriages. After filtering, the data from two different carriages are correlated with a correlation window of 5 seconds. We found that a smaller window of approximately 1 second suffices to find two carriages in the same train, but also leads to false positives for carriages in different trains. Extending the window to 5 seconds, no false positives were detected.

After the theoretical confirmation that motion information can be used for context awareness, the algorithm is implemented on wireless sensor nodes. However, the nodes used have several limitations. One limitation is the absence of floating point calculations. The filter and correlation routines have been rewritten, so only fixed point calculations are used. This might lead to errors due to accumulation of rounding successive results, but we showed that in the time frame the algorithms run this is not a problem.

The second limitation is power consumption. Filtering and correlation are so computing intensive that they cannot run over longer periods of time without exhausting the battery quickly. A first step is the reduction of the sampling rate from the original 160 Hz to 35 Hz. This is made possible because only the lower frequencies in the sampling data are significant. A lower sampling frequency would have been possible, but this does not substantially contribute to decreasing the processing load. Sampling data, filtering and performing one correlation per second takes 171.9 ms (worst case), which results in a duty cycle of around 17 percent. This exhausts the battery in a couple of hours, at most days, where 6 months is needed. The solution is found by executing the algorithm only for a period of 5 seconds from the moment the train starts moving after each stop. This is enough to establish the composition of the train and distinguish own carriages from those of different trains. During the ride, the initially detected carriages (but not the sequence of the carriages) needs to be confirmed, which can be accomplished by pinging all known carriages at regular intervals.

The last limitation is memory capacity. In our algorithm, carriages are correlated in pairs. A carriage is part of as many pairs as it has neighbors. Each pair consumes up to 460 bytes of memory in both partners. With the given memory capacity, a node can accommodate up to 16 neighbors. With a maximum of 6 correlations per second, it takes a node 3 seconds to check all its neighbors.

The circumstances in which the data are collected for the simulations, and the implementation is tested are a worst-case scenario. The trains that are used in the tests are of the same type with similar characteristics. They all exhibit the same pattern in acceleration and braking, making the data to correlate very similar. This is the main reason it takes up to 5 seconds to check carriages from different trains and only 1 second when they are in the same train.

A future research topic is the use of better accelerometers. Those used now measure up to 2g acceleration and can be used on trains that accelerate moderately. However, heavy trains that accelerate and brake more slowly have less distinctive movement patterns and need more sensitive sensors.

The principle of opportunistic sensor networks is shown here in an application that determines the composition of a train. The same principle can be used in many more applications, where movement, or any other type of sensor input, is a distinctive feature. An example is a training application, pairing people with objects they hold in their hands [28]. An other class of applications concerns logistics of goods, where groups of objects must be tracked. An example of this application class is a flower auction, where flowers in containers are loaded in trucks for transport. Correlation of the movement of the truck with the loaded containers checks whether the right containers are in the right truck. More recently, research has started in the SenSafety project in the Dutch national research program COMMIT, using sensors in mobile phones in opportunistic sensor networks [29]. Anticipated distinctive features will be acceleration, direction of movement (compass), sound and camera input. The area of application is safety in public spaces.

ACKNOWLEDGMENT

The work on opportunistic sensing in this paper is supported by the SenSafety Project in the Dutch national research program COMMIT. Other parts are supported by the iLAND Project, ARTEMIS Joint Undertaking Call for proposals ARTEMIS-2008-1, Project contract no. 100026

REFERENCES

[1] J. Scholten, J. and P. Bakker., *Opportunistic Sensing in Wireless Sensor Networks*, The Tenth International Conference on Networks, ICN 2011, January 23-28, 2011, St. Maarten, The Netherlands Antilles, pp. 224-229. IARIA.

[2] J. Scholten, R. Westenberg and M. Schoemaker , *Trainspotting, a WSN-based train integrity system*, The Eighth International Conference on Networks, ICN 2009, 1-6 March 2009, Gosier, France. pp. 226-231. IEEE.

[3] S. Bosch, M. Marin-Perianu, R.S. Marin-Perianu, J. Scholten and P.J.M. Havinga, *FollowMe! Mobile Team Coordination in Wireless Sensor and Actuator Networks*, Proceedings of the IEEE International Conference on Pervasive Computing and Communications 2009, 9-13 March 2009, Galveston, Texas, USA. pp. 151-161. IEEE.

[4] R.S. Marin-Perianu, C. Lombriser, P.J.M. Havinga, J. Scholten and G. Troester, *Tandem: A Context-Aware Method for Spontaneous Clustering of Dynamic Wireless Sensor Nodes*, Proceedings of the First International Conference on Internet of Things (IOT2008), March 2008, Zurich, Switzerland. pp. 341-359. Lecture Notes in Computer Science (4952). Springer Verlag.

[5] European Rail Traffic Monitoring System, [Online]. http://www.ertms.com. January 2012.

[6] European Railway Agency, [Online]. http://www.era.europa.eu. January 2012.

[7] H. Scholten, R. Westenberg, and M. Schoemaker, *Sensing train integrity*, IEEE Sensors 2009 Conference, 25-28 October 2009, Christchurch, New Zealand. pp. 669674. IEEE.

[8] R.S. Schwartz, E.M. van Eenennaam, G. Karagiannis, G. Heijenk, W. Klein Wolterink and J. Scholten, *Using V2V communication to create Over-the-horizon Awareness in multiple-lane highway scenarios*, IEEE Intelligent Vehicles Symposium (IV) 2010, 21-24 June 2010, La Jolla, CA, USA. pp. 998-1005. IEEE.

[9] M. Kumar, *Distributed computing in opportunistic environments*, UIC 09: Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing, Berlin, Heidelberg, 2009. Springer Verlag.

[10] L. Lilien, A. Gupta, and Z. Yang, *Opportunistic networks for emergency applications and their standard implementation framework*, Performance, Computing, and Communications Conference, 2002. 21st IEEE International, 0:588593, 2007. IEEE.

[11] L. Pelusi, A. Passarella, and M. Conti, *Opportunistic networking: data forwarding in disconnected mobile ad hoc networks*, Communications Magazine, IEEE, 44(11):134 141, November 2006. IEEE.

[12] I. Akyildiz, W. Su, and Y. Sankarasubramaniam, *A survey on sensor networks,* IEEE Comm. Magazine, vol. 40, pp. 102114, 2002. IEEE.

[13] T. Spyropoulos, K. Psounis, and C. Raghavendra, *Single-copy routing in intermittently connected mobile networks,* in Proc. Sensor and Ad Hoc Communications and Networks (SECON), 2004, pp. 235244.

[14] A. Vahdat and D. Becker, *Epidemic routing for partially connected ad hoc networks,* Department of Computer Science, Duke Univeristy, Durham, NC, Tech. Rep., 2000.

[15] Y. Wang and H. Wu, *Delay/fault-tolerant mobile sensor network (dftmsn): A new paradigm for pervasive information gathering,* IEEE Trans. Mobile Computing, vol. 6, pp. 1021 1034, 2007. IEEE.

[16] A. Lindgren and A. Droia, *Probabilistic routing protocol for intermittently connected networks,* Internet Draft draft-lindgren-dtnrg-prophet-02, Work in Progress, 2006.

[17] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, *Maxprop: Routing for vehicle-based disruption-tolerant networks,* in Proc. of IEEE INFOCOM, 2006. IEEE.

[18] D. Camara, C. Bonnet, and F. Filali, *Propagation of public safety warning message: A delay tolerant approach,* in Proc. IEEE Communications Society WCNC, 2010. IEEE.

[19] R. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh, *Citysense: A vision for an urban-scale wireless networking testbed*, Proceedings of the 2008 IEEE International Conference on Technologies for Homeland Security, pages 583588. 2008. IEEE.

[20] M. Wirz, D. Roggen, and G. Troester, *Decentralized detection of group formations from wearable acceleration sensors*, Proceedings of the 2009 IEEE International Conference on Social Computing, August 2009. IEEE.

[21] M. Wirz, D. Roggen, and G. Troester, *A methodology towards the detection of collective behavior patterns by means of body-worn sensors*, Proc. of UbiLarge workshop at Pervasive, 2010. IEEE.

[22] N. Davies, D. P. Siewiorek, and R. Sukthankar, *Special issue: Activity-based computing*, IEEE Pervasive Computing, 7(2):2021, 2008.

[23] S. Mann, *Humanistic computing: wearcom as a new framework and application for intelligent signal processing*, Proceedings of the IEEE, 86(11):21232151, 1998. IEEE.

[24] B. Myers, J. Hollan, I. Cruz, S. Bryson, D. Bulterman, T. Catarci, W. Citrin, E. Glinert, J. Grudin, and Y. Ioannidis, *Strategic directions in human-computer interaction*, ACM Computing Surveys, 28(4):794809, 1996. ACM.

[25] Ambient Systems, [Online]. http://www.ambient-systems.net. January 2012.

[26] MATLAB - The Language Of Technical Computing, [Online]. http://www.mathworks.com/products/matlab/. January 2012

[27] Correlation and dependence, [Online]. http://en.wikipedia.org/wiki/Correlation. January 2012.

[28] S. Bosch, R.S. Marin-Perianu, P.J.M. Havinga, M. Marin-Perianu, A. Horst and A. Vasilescu, *Automatic Recognition of Object Use Based on Wireless Motion Sensors,* International Symposium on Wearable Computers 2010, 10-13 October 2010, Seoul, South Korea. pp. 143-150. IEEE.

[29] COMMIT, [Online]. http://www.commit-nl.nl. January 2012.