Flexible Hardware/Software Platform for Tracking Applications

Junaid Ansari, José Sánchez, Marina Petrova, Janne Riihijärvi, Ossi Raivio, Krisakorn Rerkrai,
Christine Jardak, Frank Oldewurtel, Matthias Wellens, Lili Wu, and Petri Mähönen
Department of Wireless Networks, RWTH Aachen University, Kackertstrasse 9, D-52072 Aachen, Germany
Email: jan@mobnets.rwth-aachen.de

Abstract— **This paper describes the design, implementation and performance evaluation of a wireless sensor network based scalable outdoor vehicular tracking system. The system is highly flexible and configurable both from software and hardware architecture point of views and enables it to adapt to a wide range of vehicle tracking applications. We also present some intermediate results and the rationale behind our design approach. The system was tested for a network of 100 nodes and is scalable to a few thousand node setup. We believe that the vehicle localization and tracking results from our large scale deployment of sensor nodes and system design experience will be useful to the community.**

I. Introduction

Target tracking has always been considered as one of the most important applications of wireless sensor networks. Accordingly, several systems for tracking vehicles, people, wildlife and other types of targets have been designed and studied in the literature. The existing systems tend, however, to be very bespoke, carefully integrated solutions solving a particular, tightly defined tracking problem.

In this paper, we report on our work towards developing a *flexible* sensor network hardware/software platform for target tracking applications. Our objective was to design a modular software platform that could be adapted into a variety of scenarios, and to prototype it on a likewise novel hardware platform targeted for passive tracking scenarios. The software platform is carefully separated from the hardware by various abstraction layers, and due to the modular design various filtering, data processing and communication solutions can be used according to the needs of the particular application. Our hardware design is flexible as well, in the sense of not being confined to tracking applications with a particular target object in mind. Combination of magnetometers and passive infrared sensors makes the platform very versatile, and we enhanced the platform further by developing automatic calibration features to remove the influence of environmental changes.

The rest of the paper is structured as follows. In section III, we describe the developed hardware platform, followed by the software design in section IV, where we evaluate and analyze the performance of various sub-components comprehensively. In section V, we also present the vehicle tracking results from our large-scale sensor node deployment experiments. Finally, in section VI, we conclude the article and propose some future enhancements to the system.

II. Related Work

GPS (Global Positioning System) has extensively been used in outdoor navigation and guidance systems. It involves receiving active satellite signals for determining the position, which is vulnerable to active jamming in hostile environments. It cannot be used in monitoring and surveillance applications, which are based on passively detecting an object. Furthermore, GPS based systems cannot provide micro-level granularity and their performance suffers badly in highly urban areas and inside tunnels.

There have been many proposals for the WSN based vehicle tracking systems but real designs with extensive measurements and experimental campaigns have not been so common. One of the earlier attempts for WSN based outdoor vehicle tracking was demonstrated by UC Berkeley in 2001 [1]. However, this system is restricted to only $1D$ target tracking. C. Sharp *et. al.* designed a sensor network based large scale system for vehicle tracking [2], using magnetometer sensors. The authors found that the performance suffers badly from the effects of packaging and ferromagnetic content around.

Line in the sand project [3] aimed specially at military surveillance application.

Commercially available XSM nodes [4] use a much slower radio chip (Chipcon's CC1000) and a less attractive processor (Atmel's ATmega128L) compared to the current ultra-low-power designs. Also the absence of the magnetometer calibration scheme in some of the above mentioned systems can result in environmental interferences, which we have addressed in our design [5]. While Exscal [6] project aims at the scalability, we concentrate also on modularity and flexibility to target wider range of applications. Trio node [7] is specifically designed for outdoor applications where ample amount of sun-light is available. Approaches like image processing based methods for vehicle tracking are not only expensive but suffer heavily from constraints like illumination effects, camera transformation, scalability etc.

III. Hardware Design

The sensor selection is highly dependent upon application requirements. We opted a low-power architecture consisting of commercially available MoteIV's Telos [8] platform for computing and communication and providing standard analog and digital interfaces to connect various types of sensors. We designed our own customized sensor board platform containing the necessary sensors and signal conditioning circuitries that can be connected to the Telos platform. The sensor board consists of two types of sensors namely passive infrared (PIR) sensors and anisotropic magneto-resistive (AMR) sensors. PIR sensors detect the differential thermal energy signal rather than absolute values and are therefore highly suitable for tracking applications. AMR sensors generate an output voltage proportional to the magnetic field strength. A moving ferromagnetic object disturbs Earth's magnetic field and causes the AMR sensors to generate an output signal, which is used for detection purposes as explained in section IV-B.

The sensor node requires 3 V for its operation. Low-power architecture is an important consideration in our design. The energy efficient design includes individual power control of the PIR and AMR sensors. The output signal of the sensors is amplified using two-stage instrumentation amplifiers before feeding it to the *ADC* of the Telos platform. The circuit also includes noise suppression filters. In order to demagnetize the AMR sensors, we included a provision for external set/reset circuitry, which we always used before any large scale deployment setups.

The packaged sensor node is shown in Fig. 1. A set of four PIR sensors (called as North-West-South-East) are mounted orthogonally for a complete 360-degree field of view. Fresnel lenses are used to increase the sensing range but at the same time not losing the beam-width below 90 degrees per PIR sensor. The block diagram of the sensor node is shown in Fig. 2. A combination of two axis magnetometer (AMRa and AMRb) enables the sensor node to detect moving ferromagnetic objects in a field.

The amplitude level from the AMR sensors is highly dependent on its orientation with respect to Earth's magnetic field, the ferromagnetic material content in the surroundings and obviously on the strength of Earth's magnetic lines of force. For the optimum swing of the signal, the amplified output of the magnetometer to be fed to the *ADC* should be at the mid-scale (around 1.5 V). This is done by tuning the resistance of a

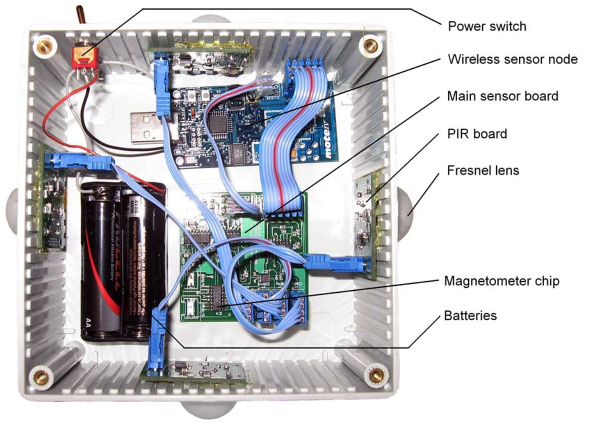

Fig. 1. Top view of the sensor node (with the weather-proof lid removed).

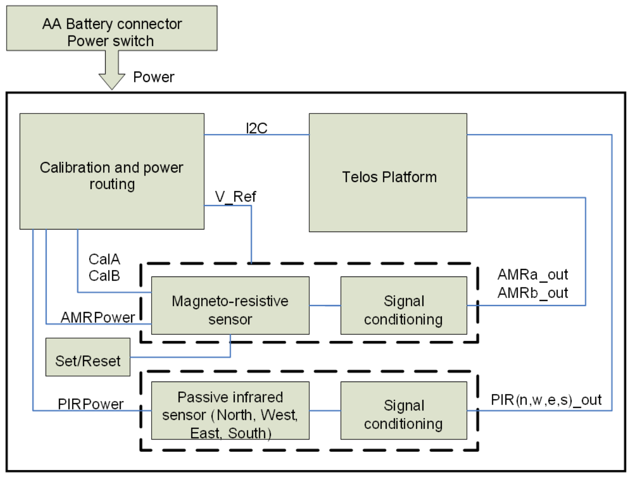

Fig. 2. Block diagram of the sensor node.

digital potentiometer, and hence adjusting the voltage levels at one of the inputs of the second stage amplifier. This process, known as calibration, is a recursive process. Firstly, a set of 10 AMR samples are taken and the mode value is calculated. The underlying reason is that the probability of an outlier is very low. The mode value is converted into voltage and is checked whether it lies within a small window around the mid-scale voltage. Otherwise, the value of the potentiometer is increased or decreased accordingly by sending appropriate commands to the digital potentiometer over the *I2C* bus. The process is repeated till the voltage assumes a mid-scale value. This process not only prevents clipping of the signal but also enables the sensor node to calibrate automatically to any environmental condition.

IV. Software Design and Evaluation of Individual Components

We followed a modular and layered approach in the software development, which allows reuse of different modules, adaptability to application needs and high scalability. The embedded software application running on each sensor node was developed in TinyOS [9], whereas data gathering and tracking algorithm was implemented in C/C++ on a gateway PC. Keeping various components decoupled from each other also facilitated the debugging process, which otherwise can be very complicated, specially in embedded software development.

The system architecture for a vehicle tracking scenario is shown in Fig. 3. The sensor nodes detect the presence of a vehicle in the sensor field and transmit the relevant information to the gateway node, connected to the gateway PC. The gateway PC collaboratively uses the information from sensor nodes to track the vehicle. The software stack shown on the right hand side in the figure is implemented on the sensor nodes, whereas the one on the left hand side is implemented on the gateway. The lowest level of abstraction in our embedded software stack is at the device driver level, giving access to various peripherals and interfaces of the microcontroller. The system uses *ADC* interface for getting readings from the AMR and PIR sensors. The *I2C* interface is used to control the power to the AMR and PIR circuitries and also to calibrate the AMR circuit. The *SPI* interface is used by the microcontroller to communicate with the IEEE 802.15.4 compliant radio transceiver chip (CC2420 [10]) on the Telos platform. The *UART* interface is used at the gateway node to communicate with the gateway PC. Since all the interfaces share a common serial bus, so we introduced a bus arbitration mechanism to avoid potential conflicts.

We consider the sensor calibration scheme, the data acquisition from sensors and the time synchronization as middleware components in our software stack and use separate well defined interfaces for these. Our AMR calibration scheme is a recursive process and can flexibly be called from the application, when required. The data acquisition interface gives the data from the four PIR and two AMR sensors at a configurable sampling rate. The time synchronization protocol (Flood-time-synchronization-protocol [11]) running in the WSN does not need to be configured manually and automatically chooses the master node. All nodes including the gateway node get into synchronization with the master node at the start-up.

A. *Software Filtering*

Despite the use of hardware implemented noise removing filters, the effects of the thermal noise and the noise from the system electronics itself is still present. For this purpose, we implemented a low-pass filter. We restricted ourselves to finite impulse response (FIR) filters of lower order to keep the complexity as low as possible while guaranteeing the filter stability. However, in our design it is possible to adjust all filter parameters such as type, order, cut-off frequency and filter coefficients according to the characteristics of the incoming raw sensor readings.

Fig. 4 shows the signature of a slowly moving vehicle passing by a PIR sensor. The grey solid line shows the

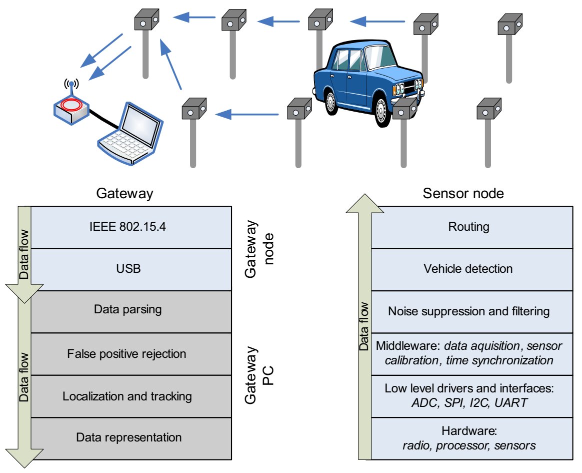
Gateway
IEEE 802.15.4
USB
Data parsing
False positive rejection
Localization and tracking
Data representation
Data flow
Data flow
Gateway node
Gateway PC
Sensor node
Routing
Vehicle detection
Noise suppression and filtering
Middleware: data aquisition, sensor calibration, time synchronization
Low level drivers and interfaces: ADC, SPI, I2C, UART
Hardware: radio, processor, sensors
Data flow

Fig. 3. Design and architecture of the vehicle tracking system.

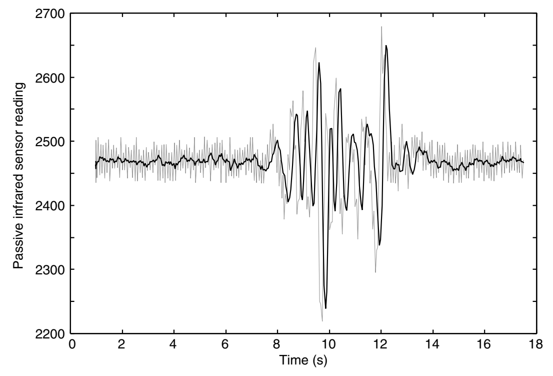

Fig. 4. The signature of a slowly moving vehicle as seen by the PIR sensor. The grey curve shows the raw PIR readings; the black curve is the same data after FIR filtering.

signature based on raw values and the black solid line depicts its noise filtered representation. The noise filtering smoothes the signature which enables the detection algorithm to obtain reliable results. The low-pass filter is applied to the AMR sensor readings as well and shows similar behaviour.

The sensor node platform, placed outdoors, is vulnerable to changes in ambient temperature and/or light. In addition, Earth's magnetic field changes both in orientation and strength all the time. This affects the ambient field strength detected by the magnetometer. Because of the changes in the environment, the base level of the magnetometer readings needs to be corrected. We achieve this by constantly calculating a long-term exponentially weighted moving average (EWMA) and subtracting this value from the noise filtered data. After applying the EWMA filter each value represents the weighted average of all the previous means, including the mean of the present sample. The weights decrease exponentially going backward in time. Only the base-level corrected data is used as an input for the detection algorithm which is described in the next subsection. Again the smoothing factor of EWMA filter is kept adjustable.

B. *Vehicle Detection*

After noise filtering and base level correction, we obtain vehicle signatures (both from AMR and PIR sensors) as presented in Fig. 4 and Fig. 5. These are the inputs to the detection algorithm. For simplicity, we describe the detection procedure using AMR sensor readings in the following.

Every time the sensor readings cross a positive and negative thresholds integration (called impulse integration) is started. These thresholds can be configured through software. When the sensor readings return between the thresholds, a dwell timer is started which is indicated by thick black line in Fig. 5. If the sensor readings cross the thresholds before the dwell timer expires, the timer is reset and the calculation of the impulse integration is continued. Two of such events are shown in the figure and are represented by red crosses. However, on the contrary, when the dwell timer expires, the impulse integration value is compared against a threshold

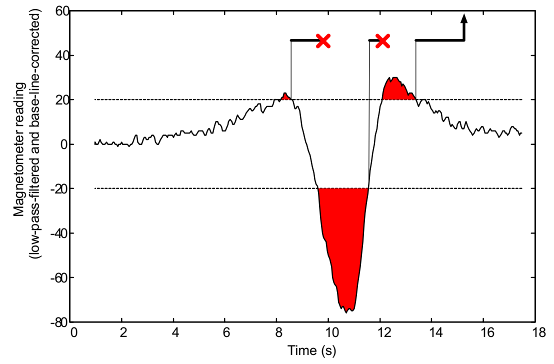

Fig. 5. The signature of a slowly moving vehicle as seen by the AMR sensor after FIR and EWMA filtering (solid black curve). The red area between this curve and the thresholds (dotted lines) represents the integrated impulse value. When the AMR readings return to the area between the thresholds, a dwell timer is started (horizontal black line). If the readings cross the threshold before this timer expires, the timer is reset and the integration of impulse continued (red cross terminating the black timer line). When the dwell timer expires (upward arrow), the impulse value is compared against a pre-set value. Too small impulses are considered as noise, and detection is not signalled (red cross over the upward arrow).

called the impulse threshold. A detection is triggered only when the integrated impulse value is greater than the impulse threshold. The dwell timer value and the impulse thresholds are selected empirically for different objects. A very small dwell timer value causes one event to be splitted into two or more detections; too large value causes two or more nearby events to be merged into a single detection. In case of too small impulse thresholds, noise peaks may cause detection and a too large value results real events to be considered as noise. When the object is detected, the event is time-stamped and a message is sent from the sensor nodes to the gateway using the TinyAODV [12] routing protocol.

C. False Positives Rejection

During our field experiments, we observed that sometimes the sensors (both PIR and AMR) triggered detections even when no vehicle was present in the sensor field. Due to sudden changes in the environmental conditions (abrupt clouds and sun), PIR sensors instigated false detections. Some grasshoppers and sparrows, very near to the sensor nodes also caused PIR detections, which of course is undesirable. In the field tests, a very few times magnetometer triggered false detections because of the drift of the node position/orientation due to air or vibration. The nodes may also trigger false detections because the detection algorithm is state dependent and hence a last state can possibly fulfil the detection conditions thereby causing a false detection.

Although, the noise suppression is done at various levels in the system, it is possible that some unwanted noisy samples can still reside in the system and can trigger false detections. Fig. 6 shows the packets received at the gateway node when a vehicle passes in front of a sensor field composed of orderly arranged 50 sensor nodes. It can easily be seen that a few detection packets (y-axis magnetometer and some x-axis

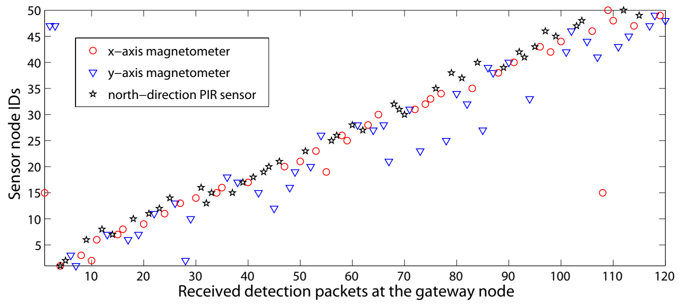

Fig. 6. Illustration of different types of detection packets arriving at the gateway sensor node as the vehicle traverses once through the sensor field starting from the first sensor node towards the last sensor node.

magnetometer) from some nodes do not adhere to the linearly increasing line. These may result in false position estimates and have to be suppressed.

On the wireless sensor node alone, there is no possibility to identify the cases of false detections. This information is only available from the network as a whole. Our system uses a neighborhood based false positives rejection scheme. The sensor detections of neighbouring nodes along with their global time-stamps are used to determine the outliers and false positives. By fusing the AMR and PIR detections using the spatial (neighbourhood) and time-stamp information in the network, anomalies are effectively suppressed.

V. Overall System Performance Results

We have found that the sensor nodes are able to detect the presence of normally seen on-road cars reliably from a distance of 5 m. We evaluated the performance of the system in a setup of 50 sensor nodes both in a paved asphalt surface and in uneven rugged terrain. The sensor nodes deployed in the sensor field have unique IDs and their individual positions are known before hand. The sensor nodes are deployed along a U-shaped track in two rows with every sensor node 5 m apart from any other. It may be noted that the deployment scheme uses a $1D$ sensor nodes setup but except for the localization logic, the rest of the system components can be used as-it-is for any arbitrary motion in $2D$. For tracking multiple objects at the same time, schemes like [13] needs to be included after the false positive rejection scheme.

The tracking algorithm computes the position of the vehicle using the detection packets received from the WSN. The gateway PC accepts a data packet from the gateway sensor node. For each wireless sensor node in the sensor field, there exists a data structure containing information like the sensor detections, time stamp value and the node position. When the gateway PC accepts a detection packet from a particular sensor node, it updates the corresponding fields for the sensor detections and time stamp. Due to the nature of the threshold based vehicle detection scheme, different sensors may detect the same vehicle at close but slightly different time instants. As a result, the gateway PC receives multiple detection packets from a particular sensor node, corresponding

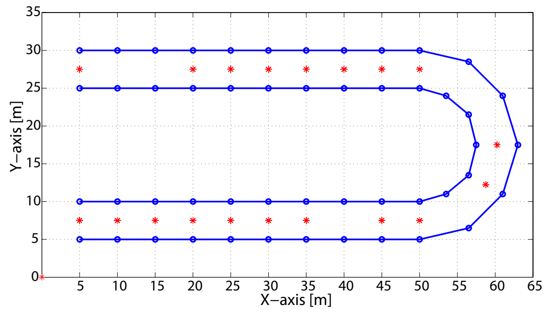

Fig. 7. Output of the tracking algorithm.

to different sensors. The algorithm is, therefore, developed to update the corresponding detection field values on the fly. Since in our scenario, two sensor nodes are deployed just in front of each other and the vehicle passes through the corridor between them, the two sensor nodes (theoretically) detect the vehicle at the same time instant. However since only one packet can be transmitted at a time, the tracking algorithm tries to compute the position of the vehicle at the arrival of each valid packet. When there is enough information available to compute the position incorporating the detections performed by the neighborhood and time-stamp values of the packets, the tracking algorithm results into a position update. It may be noted that this scheme does not impart any extra latency in comparison to the approach of first storing all the relevant detections and later applying a localization logic.

Fig. 7 shows the output of the tracking algorithm. The sensor nodes are shown by small circles on the U-shaped sensor field deployment scheme. The asterisks show the position of the vehicle, running at a speed of approx. 50 km/h, computed by the tracking algorithm. Before the vehicle enters into the sensor field, the default position is at the origin. The tracking algorithm approximates the position of the vehicle to be in the middle of two oppositely facing nodes. It may be observed that some of nodes did not trigger detections. Using the positioning information, the tracking algorithm also approximates the speed of the vehicle, which we found to be reasonably accurate. In the data representation component, the vehicle's trajectory is computed on GPS scale. This is achieved by introducing two GPS anchor points in the sensor field and performing a mapping from Cartesian to GPS coordinate system based on Vincenty's work [14].

VI. Conclusions and Future Work

In this paper, we have presented both the hardware and software design of our platform for tracking applications. The modular and layered approach makes our platform flexible and adaptable to a broad category of passive target tracking applications. We also describe the sub-components of our target tracking system and the underlying logic deriving from realistic experimental data. We experimented it for the presented outdoor vehicle tracking application and also for an indoor intruder detection application. For these applications, we could simply set/tune the parameters such as sampling rates for the sensors, type, order and cut-off frequency of the FIR filters, the coefficient of the EWMA filter in a flexible way and finally optimize the dwell timer and impulse threshold values used in the detection algorithm.

We obtained reliable vehicle tracking results from a large scale system deployment experiment in real-time. We found that the system has a lower latency and is scalable. Because of the modularity in the design, we could also easily integrate a dynamic gateway selection mechanism on top of the routing for making the system robust against possible gateway failures.

In the future, finer granularity (reliability) in position estimates may be introduced by mapping the confidence levels of individual detections against a suitable cost function. Different types of vehicles have different but unique signal signatures [15], [16]. Using pattern recognition and signal processing algorithms, these signatures may be used to identify the type of the vehicle e.g. a car, van, truck, bus, etc.

At present, the system needs to be deployed according to a prior node location map. In the future, a self-localizing service may be added on the sensor nodes. As a result, the sensor nodes may be thrown randomly in the sensor field and will be able to localize themselves automatically (e.g. [17]).

Acknowledgment

We would like to thank European Union (project IST-004536-RUNES), IABG and RWTH Aachen University for the financial support.

References

[1] Tracking vehicles with a UAV-delivered sensor network, "http://robotics.eecs.berkeley.edu/∼pister/29palms0103/."

[2] C. Sharp *et al.*, "Design and implementation of a sensor network system for vehicle tracking and autonomous interception," in *Second European Workshop on Wireless Sensor Networks*, Jan.-Feb. 2005.

[3] A. Arora *et al.*, "A line in the sand: A wireless sensor network for target detection," *To appear in Computer Networks (Elsevier)*, 2004.

[4] "Wireless security system - MSP410," http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MSP410_Datasheet.pdf.

[5] J. Ansari *et al.*, "Demo abstract: Flexible hardware/software platform for tracking applications," in *SenSys*, 2006.

[6] A. Arora and R. Ramnath and E. Ertin, "Exscal: Elements of an extreme scale wireless sensor network," 2005.

[7] P. Dutta *et al.*, "Trio: enabling sustainable and scalable outdoor wireless sensor network deployments," in *IPSN '06*, 2006.

[8] "Datasheet for MoteIV's Telos," http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf.

[9] TinyOS: An open-source Operating System for the networked sensor regime , "http://www.tinyos.net/."

[10] "Datasheet for chipcon's CC2420 2.4 GHz RF Tranceiver," http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf.

[11] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *SenSys '04*, 2004.

[12] Tiny AODV, "http://tinyos.cvs.sourceforge.net/."

[13] S. Oh, L. Schenato, P. Chen and S. Sastry, "A scalable real-time multiple-target tracking algorithm for sensor networks," Dept. of EECS, UC Berkeley, USA., Tech. Rep. UCB//ERL M05/9, Feb. 2005.

[14] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," Survey review 22 (176), 1975.

[15] M. F. Duarte and Y. H. Hu, "Vehicle classification in distributed sensor networks," *J. Parall. Distr. Comp.*., vol. 64, no. 7, pp. 826–838, 2004.

[16] L. Gu *et al.*, "Lightweight detection and classification for wireless sensor networks in realistic environments," in *SenSys '05*, 2005, pp. 205–217.

[17] N. Priyantha *et al.*, "Anchor-Free Distributed Localization in Sensor Networks," CSAIL, M.I.T., Tech. Rep. TR-892, April 2003.