# Autonomous Vehicle Coordination with Wireless Sensor and Actuator Networks

Mihai Marin-Perianu, Stephan Bosch, Raluca Marin-Perianu,
Hans Scholten, Paul Havinga
University of Twente

A coordinated team of mobile wireless sensor and actuator nodes can bring numerous benefits for various applications in the field of cooperative surveillance, mapping unknown areas, disaster management, automated highway and space exploration. This paper explores the idea of mobile nodes using vehicles on wheels, augmented with wireless, sensing and control capabilities. One of the vehicles acts as a *leader*, being remotely driven by the user, the others represent the *followers*. Each vehicle has a low-power wireless sensor node attached, featuring a 3D accelerometer and a magnetic compass. Speed and orientation are computed in real time using inertial navigation techniques. The leader periodically transmits these measures to the followers, which implement a lightweight fuzzy logic controller for imitating the leader's movement pattern. We report in detail on all development phases, covering design, simulation, controller tuning, inertial sensor evaluation, calibration, scheduling, fixed-point computation, debugging, benchmarking, field experiments and lessons learned.

Categories and Subject Descriptors: C.2.1 [**Network Architecture and Design**]: Wireless communication

Additional Key Words and Phrases: Wireless Sensor and Actuator Networks, Movement Coordination, Fuzzy Control, Vehicular Networks

## 1. INTRODUCTION

Wireless Sensor and Actuator Networks (WSANs) [Akyildiz and Kasimoglu 2004] and complementary technologies (Smart Collaborating Objects, Internet of Things) lay down the foundations for a future pervasive world in which everything will be networked. This paper makes a step forward in proving that *WSANs can sense, reason and react as a group with distributed intelligence, without any intervention from the back-end and despite the hardware limitations of sensor nodes.* More specifically, we address the problem of distributed movement coordination of autonomous vehicles equipped with wireless sensor nodes. The final goal is to have a self-organizing team (or swarm) of nodes that maintain a formation by periodically exchanging their sensed movement information. We aim to provide a *fully localized* solution, without any external PC-based control, and based solely on low-cost, low-power inertial sensors (no cameras or GPS, total cost of the hardware

Author's address: Mihai Marin-Perianu, Pervasive Systems, University of Twente, PO-Box 217, 7500 AE, Enschede, The Netherlands, m.marinperianu@utwente.nl

platform below 150 $). A broad range of application domains would benefit from having many such inexpensive wireless vehicles that autonomously team up and coordinate their movement, for example: manoeuvre learning [Coates et al. 2008], mapping unknown areas [Burgard et al. 2005], automated highway [Hayat 2002], truck convoys [Fritz 1999], space exploration [Gaura and Newman 2006].

Distributed team coordination in WSANs faces a number of challenges. Firstly, executing all the tasks on the node – sensor sampling, processing, communication and control – may easily exceed the computational and memory resources available. Consequently, we must find the right *scheduling* that trades-off between accuracy and responsiveness, on the one hand, and sampling frequency and wireless communication duty cycle, on the other hand. Secondly, actuator nodes must run a navigation control loop for regulating the movement of vehicles. Designing and implementing a suitable *controller* for this purpose is far from trivial on limited hardware. Thirdly, using inexpensive, low-power inertial sensors means a relatively low accuracy and robustness to noise. Therefore, *calibration*, *filtering* and *dynamic error compensation* are strongly required for improving the quality of measurements.

In view of these challenges, the key contributions of this paper are as follows. Firstly, we devise a miniaturized, low-cost navigation system using low-power wireless sensor nodes equipped with three-axial accelerometers and magnetic compasses. Secondly, we explore fuzzy logic as a lightweight and robust control solution for coordinating the group movement in a leader-follower fashion. Thirdly, we report on all development phases, covering simulation, controller tuning, inertial sensor evaluation, calibration, scheduling, fixed-point computation, debugging and benchmarking. Finally, we evaluate the performance of our prototype system through field experiments and we discuss the most important results. Visit [FollowMe 2008] to see a video demonstration of the system.

## 2. RELATED WORK

There is extensive literature on various control techniques for the formation control of mobile robots. The main approaches can be classified as: *behavior-based*, *bio-inspired*, *virtual leader* and *leader-follower* [Liu et al. 2007]. Balch and Arkin [Balch and Arkin 1998] evaluate reactive formation *behaviors* integrated with navigational behaviors to enable a robotic team to reach navigational goals and simultaneously avoid hazards. The application is military, with a focus on maintaining a line, column, diamond or wedge formation of Unmanned Ground Vehicles (UGVs). Starting from the pioneering work of Reynolds [Reynolds 1987], *bio-inspired* approaches consider the coordination of mobile agents in a leaderless fashion. The related work [Jadbabaie et al. 2003; Tanner et al. 2004] focuses on the convergence and stability of the local control laws at the theoretical level. In the *virtual leader* approaches [Egerstedt et al. 2001; Ögren et al. 2002], the robots follow a reference path parameterized by a virtual vehicle moving on that path. The *leader-follower* paradigm is a popular choice for implementing robot formation control. A significant number of studies [Swaroop and Hedrick 1996; Tanner et al. 2004; Stipanovic et al. 2004] assume that global communication among robots is possible and concentrate on aspects such as stability and controller synthesis. In [Fredslund and

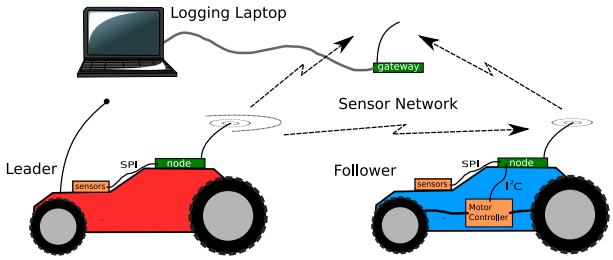



Fig. 1. Overview of the leader-follower scenario.

Mataric 2002] the communication is reduced to a periodic heartbeat and robots use only local knowledge (from camera, sonars and laser) to maintain chains of friends and thus follow each other. In [Das et al. 2002; Vidal et al. 2004] no communication is assumed among robots and consequently the followers have to implement continuous leader detection in addition to tracking, using color tracking or panoramic cameras. With respect to these studies, our work follows the leader-follower approach and assumes communication from leader to followers. The focus is however different. While the related work concentrates mainly on strategies for precise robot formation control, we explore a solution for distributed movement coordination of any motion-enabled entities and we analyze the real-world challenges associated with implementing the complete system on resource-constrained sensor nodes.

Mobile WSAN platforms based on MICA motes have been explored by Wang *et al.* [Wang et al. 2004]: MASmote, MICAmote, CotsBots and Robomote. The MASnet project also considers the usage of MASmotes for formation control [Wang et al. 2005], including a leader-follower strategy. The MASmotes rely on a pseudo-GPS location system (based on cameras) and odometry to measure node displacement. In contrast, our solution is fully localized and does not require any infrastructure. Allred *et al.* [Allred et al. 2007] describe SensorFlock, which is composed of small bird-sized nodes called micro-air vehicles (MAVs). The MAVs are expected to communicate wirelessly and follow a certain trajectory mission plan. The flight control system fuses information from the GPS and gyroscope sensors on board. SensorFlock focuses on keeping the nodes autonomously in the air and provides an in-depth study of the RF characteristics and networking connectivity. In comparison, we focus on inertial sensing and explore fuzzy logic as a lightweight yet robust control method for coordinating the movements of mobile nodes in the field.

## 3. NAVIGATION

As depicted in Figure 1, we consider the scenario of a *leader* vehicle, whose trajectory has to be copied by *follower* vehicles. The result is a moving ensemble that can be controlled from a single point or can follow a unique mission plan deployed only on the leader. Theoretically, synchronous movement is achieved when all the vehicles maintain the same *velocity* and *heading* with respect to a reference system. When moving, the leader computes its velocity and heading from sensor measurements, and broadcasts this data periodically to the followers. Each follower determines its own speed and heading, compares them with the information

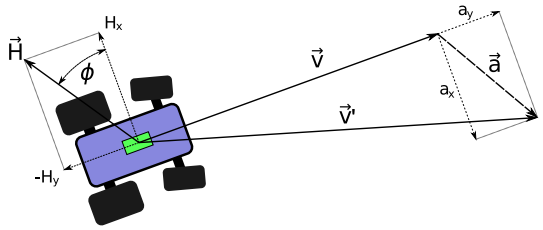



Fig. 2. Magnetic, velocity and acceleration vectors defined for the vehicle.

received from the leader and takes the necessary action to correct its trajectory if necessary.

### 3.1 Velocity Integration

In strapdown inertial navigation systems, velocity $v$ is computed from the following equation [Titterton and Weston 2004]:

$$\frac{d\vec{v}}{dt} = \vec{a} - \vec{\omega} \times \vec{v} - \vec{\omega} \times (\vec{\omega} \times \vec{r}) + \vec{g} \tag{1}$$

where $\vec{a}$ is the specific force acceleration, $\vec{g}$ is the gravitational field strength, $\vec{\omega} \times \vec{v}$ corresponds to the effect of Coriolis force and $\vec{\omega} \times (\vec{\omega} \times \vec{r})$ defines the centripetal acceleration.

To measure these parameters, a typical inertial navigation system consists of three types of sensors: accelerometer, gyroscope and magnetic compass. However, the gyroscope increases the complexity of computations, the power consumption and the total cost. If we assume that the accelerometer is mounted firmly to the vehicle rigid frame and its Y axis points always to the direction of driving, it is possible to infer the velocity only from the accelerometer and compass data. Figure 2 illustrates our acceleration integration algorithm. The current velocity vector $\vec{v'}$ is computed by adding the instantaneous acceleration $\vec{a}$ to the vector $\vec{v}$ from the previous step (the size of $\vec{a}$ is intentionally exaggerated in the figure). The magnitude $v'$ is:

$$v' = \sqrt{a_x^2 + (v + a_y)^2} \tag{2}$$

where $a_x$ and $a_y$ are the components of the acceleration vector. The time step for integration is considered equal to unity.

A known problem of integrating acceleration is the accumulation of errors in time. If no external reference is available, the error can potentially increase to infinity. Our assumption is that the vehicles do not move continuously, but also have stationary periods during which the velocity estimate can be reset to zero. For determining the "standing still" situation, the sensor nodes continuously analyze the variance of the acceleration over a sliding time window with respect to a threshold determined experimentally. The downside of this approach is that it introduces a certain delay between the actual moment of stopping and the detection of standing still.

### 3.2 Heading Computation

The heading (or azimuth) $\phi$ is computed from the components of the magnetic field intensity $H$ measured by the magnetic compass (see Figure 2):

$$\phi = \arctan(H_y/H_x) \tag{3}$$

The heading $\phi$ indicates the vehicle orientation with respect to the magnetic North Pole. In order to map $\phi$ to the interval $[-\pi : \pi]$, we have:

$$\phi = \begin{cases} \pi + \arctan(H_y/H_x) & \text{if } H_x < 0, H_y \geq 0, \\ \pi/2 & \text{if } H_x = 0, H_y \geq 0, \\ \arctan(H_y/H_x) & \text{if } H_x > 0, \\ -\pi/2 & \text{if } H_x = 0, H_y < 0, \\ -\pi + \arctan(H_y/H_x) & \text{if } H_x < 0, H_y < 0 \end{cases} \tag{4}$$

This result holds only when the compass plane is perfectly horizontal. In practice, we must compensate for the roll ($\theta$) and pitch ($\psi$) tilt angles, which can be inferred from the accelerometer output when the system is standing still [Tuck 2007]:

$$\theta = \arctan(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}) \ , \ \ \psi = \arctan(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}) \tag{5}$$

In this case, the transformed magnetic intensities $H'_x$ and $H'_y$ to be used in Equations 3 and 4 are [Caruso 2000]:

$$\begin{aligned} H'_x &= H_x \cos(\psi) + H_y \sin(\theta)\sin(\psi) - H_z \cos(\theta)\sin(\psi) \\ H'_y &= H_y \cos(\theta) + H_z \sin(\theta) \end{aligned} \tag{6}$$

As we can see, the trigonometric computations are quite involved for a sensor node microcontroller. In addition, the variation of roll and pitch angles when the vehicles are moving cannot be measured without a gyroscope.

We take therefore a simpler approach and assume that the tilt will remain constant during driving. As a consequence, the tilt compensation from Equations 5 - 6 can be replaced by the calibration procedure described in Section 6.3. Our experimental results show that this method is robust to tilt effects whilst driving, as long as the vehicles remain on a relatively flat surface.

## 4. FUZZY CONTROLLER

We use fuzzy control because of several advantages with respect to the specifics of WSANs. Firstly, fuzzy logic can be implemented on limited hardware and is computationally fast [Henkind and Harrison 1988; Marin-Perianu and Havinga 2007]. Secondly, it handles unreliable and imprecise information (which is usually the case with sensor data), offering a robust solution to decision fusion under uncertainty [Samarasooriya and Varshney 2000]. Thirdly, fuzzy-based methodology substantially reduces the design and development time in control systems [Bih 2006]. Finally, fuzzy controllers handle non-linear systems (most real-life physical systems are non-linear) better than conventional approaches [Bih 2006].

A fuzzy controller executes three basic steps: *fuzzification*, *inference* and *defuzzification*. During fuzzification, the numeric input values are mapped to fuzzy sets by applying the membership functions. Based on the fuzzified inputs, the controller

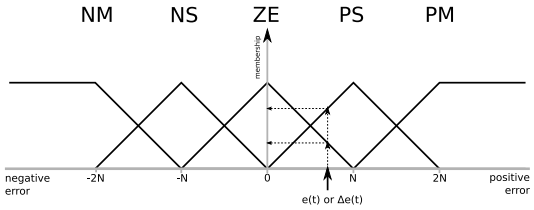



Fig. 3. Membership functions for fuzzy inputs.

| e \ $\Delta e$ | **NM** | **NS** | **ZE** | **PS** | **PM** |
|---|---|---|---|---|---|
| **NM** | PM | PM | PM | PS | ZE |
| **NS** | PM | PM | PS | ZE | NS |
| **ZE** | PM | PS | ZE | NS | NM |
| **PS** | PS | ZE | NS | NM | NM |
| **PM** | ZE | NS | NM | NM | NM |

Table I. Fuzzy inference rule table.

infers through its IF-THEN rule set and produces an aggregated fuzzy output. The final control action is derived by defuzzifying this aggregated fuzzy output.

In our case, the control objective of the follower is to adapt the velocity and heading according to the leader movements. Since the vehicle has separate motors for accelerating and steering, we decompose the problem into two independent controllers, with the benefit of simplifying the design and tuning. The two controllers are structurally identical. The only differences lie in the range of the input values, the definition of the membership functions and the post-processing operations.

The two fuzzy controllers follow the design methodology proposed by Mamdani [Mamdani and Assilian 1975]. The building blocks are:

—*Inputs.* As inputs we use the error $e$ and the change in error $\Delta e$ between the actual values of the movement parameters and the desired values. This is a common approach for improving the controller stability when the output value is close to the optimal operating point.
—*Fuzzy sets.* For an increased control granularity, we define five fuzzy sets for each input, namely $NM$ (Negative Medium), $NS$ (Negative Small), $ZE$ (Zero Equal), $PS$ (Positive Small) and $PM$ (Positive Medium).
—*Membership functions.* To leverage the computational effort, we use triangular membership functions spaced equally with distance $N$, as depicted in Figure 3. The width of the membership functions influences the *aggressiveness* of the controller, i.e. what margin of error it tries to achieve.
—*Rule-based inference.* We use the *max-min* fuzzy inference method over the rule set specified in Table I.
—*Output.* The controller output is decided by defuzzifying the aggregated inference result according to the *center of gravity* (COG) method.

## 5. SIMULATION

The simulation framework we developed plays an important role in designing and tuning the fuzzy logic controller. More specifically, we revert to simulation in the following phases:

(1) Controller design and evaluation using synthetic, idealized data.
(2) Tuning the controller parameters using real sensor data from experiments as input.
(3) Debugging the implementation of the controller on the sensor node platform.

In the following, we describe the first two phases, together with the simulated vehicle model. The third phase is explained in Section 6.7. The simulations are cross-validated with field experiments in Section 7.4.3.

### 5.1 Simulation Model

To model the dynamics of the vehicle realistically, we take both the friction and the throttle capacity of the motor into account. The effective acceleration $a_e$, i.e. the result of the net force applied to the vehicle, is given by:

$$a_e = ta_t \ - \ a_f \tag{7}$$

where $t \in [-1; 1]$ is the current throttle control value, $a_t$ is the maximum acceleration generated by the throttle (and depends on actual level of the battery powering the motor) and $a_f$ represents the acceleration induced by friction. The effective acceleration $a_e$ is further integrated to estimate velocity, as explained in Section 3.1.

Deriving an accurate model for the frictional acceleration $a_f$ is complicated due to the multitude of physical factors involved in the process. In our simulations, we use the following simplified model:

$$a_f = a_{fk} \ + \ c_d|v| \ + \ c_s|\delta| \ + \ ba_{fb} \tag{8}$$

where $a_{fk}$ is the usual kinetic friction, $c_d|v|$ represents the drag friction (proportional to velocity $v$ for small objects moving at low speeds, according to Stokes law), $c_s|\delta|$ yields the sliding friction when the vehicle is steering with angle $\delta$ and $a_{fb}$ is the friction induced by braking ($b \in \{0; 1\}$ being the brake control signal). [1]

The final component of the vehicle model is the heading. We assume that the front wheels of the vehicle can steer with an arbitrary angle between $-\delta$ and $\delta$. The equation for updating the current heading $\phi'$ is:

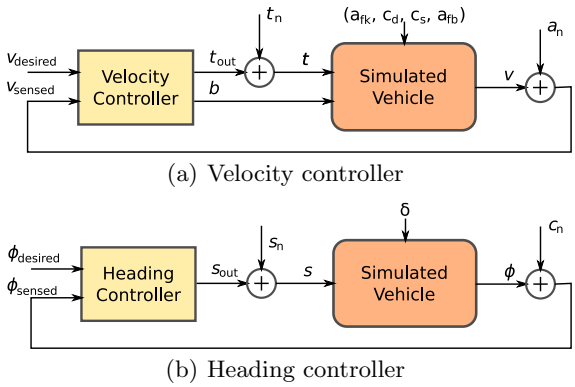
$$\phi' = \phi \ - \ s\delta \tag{9}$$

where $\phi$ is the heading at the previous time step and $s \in [-1; 1]$ is the steering control value.

### 5.2 Controller simulation

We first present the separate simulations of the velocity and heading controllers, and then the combined simulation.

[1] The heuristic assumption that sliding friction varies proportionally with the steering angle is solely based on our observations from field tests. Coefficients $c_d, c_s$ are also obtained experimentally.




(a) Velocity controller

(b) Heading controller

Fig. 4. Simulated controller models.

5.2.1 *Velocity simulation.* The velocity simulation follows the model described by Equations 2, 7 and 8. In Figure 4(a) we represent the simulated environment, which accepts the throttle $t$ and brake $b$ control signals from the velocity controller, and yields the simulated velocity $v$.

We model two types of noise – throttle noise $t_n$ and accelerometer noise $a_n$ – as uniformly distributed random variables. The simulated noisy velocity measurement $v_{sensed}$ is input to the velocity controller. The controller tries to achieve the desired velocity $v_{desired}$ using its throttle $t_{out}$ and brake $b$ outputs.

5.2.2 *Heading simulation.* Figure 4(b) shows the structure of the heading simulation. The heading controller has two inputs – the desired heading $\phi_{desired}$ and the measured heading $\phi_{sensed}$ – and outputs the steering control value $s$. The car drives at constant velocity and, with each iteration of the simulation, the current heading $\phi$ is updated with the current steering control value $s$ using Equation 9.

Similar to the velocity simulation, we consider two types of uniformly distributed noise: steering noise $s_n$ (related to inaccuracies in the steering mechanism and external influences, such as a rough road surface) and compass noise $c_n$. In order to create a more realistic response of the steering, we apply a running average filter at the controller output. This corresponds to the actual behavior of the vehicle steering, which exhibits a certain delay in reacting to high rates of changes.

5.2.3 *Full simulation.* The full simulation includes both the velocity and the steering models described in the previous sections. This combination entails that the steering control value of the previous iteration is fed to the friction model of the velocity simulation. The current steering control value is inverted if the velocity becomes negative, to properly simulate backward driving.

Our simulations show that the follower controller works adequately when subjected to the model of dynamics explained in Section 5.1. The follower manages to closely keep to the leader trajectory and shows realistic changes in speed when taking turns. For detailed results on how field test results match the simulations, see Section 7.4.3.

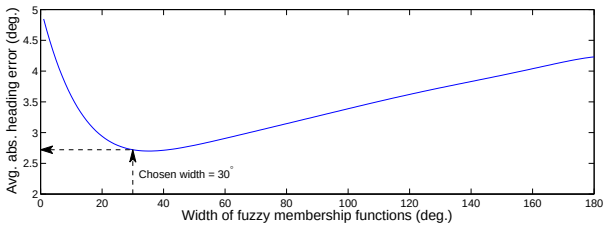



Fig. 5. Tuning the width of membership functions.

### 5.3 Tuning

For tuning the fuzzy controller membership functions, we run a series of simulations with different controller configurations. For example, Figure 5 shows the average absolute error in the heading of a simulated vehicle driving a straight line, plotted against the width of the fuzzy input membership functions. Each point in the plot is the average result of 20 simulation runs. As we explain in Section 4, the heading controller becomes more aggressive when the input membership functions of Figure 3 get narrower. On the one hand, if the controller is too aggressive, it can produce undesired oscillations in steering. On the other hand, if it is not aggressive enough, the follower has low responsiveness and performs suboptimally. Figure 5 provides an indication of the optimum value. We currently use a width of 30°for the membership functions presented in Figure 3.

## 6. IMPLEMENTATION

We implement the leader-follower system on two toy cars. The follower is modified to allow the sensor node to control its actuators. The leader car remains unmodified apart from the sensor node attachment. For a video demonstration of the system, visit [FollowMe 2008].

In the following, we provide a description of the hardware and software implementation. We start by presenting the sensors used, the interfacing to the nodes, the placement on the toy cars and the calibration procedure. Next, we describe the actuation on the follower vehicle, the scheduling on the sensor nodes and the implementation of the calculations from Section 3. We end this section with a description of the debugging and benchmarking of the controller implementation.

### 6.1 Sensing Interface

6.1.1 *Hardware.* Our sensor nodes are equipped with the low-power MSP430 microcontroller from Texas Instruments (48kB of FLASH memory and 10kB of RAM) and a radio transceiver with a maximum data rate of 100kbps. The selection of inertial sensors is driven by cost concerns, accuracy, form factor, power consumption and interfacing capabilities. As accelerometer, we choose the LIS3LV02DQ three-axial sensor [STMicroelectronics 2008]. The price is around 15 $ and the typical power consumption is 2mW. The list of features include user selectable full scale of ±2g and ±6g, $I^2C$/SPI digital interface, programmable threshold for wake-up/free-fall and various sample rates up to 2.56kHz. As compass, we choose the three-axial MicroMag 3 magnetometer sensor [PNI Corporation 2008]. The price

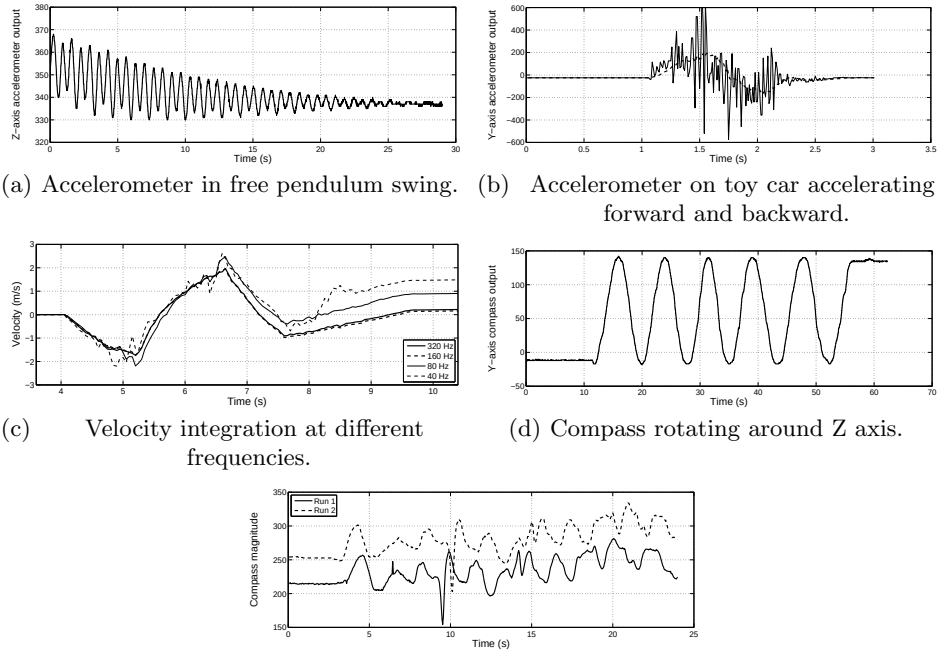
range is 60 $ and the typical power consumption is 1.2mW. The MicroMag3 uses the Magneto-Inductive (MI) sensing technique [Caruso et al. 1998] and provides a relatively large field measurement range (±11 Gauss) on the digital SPI interface. The compass sensor and the accelerometer are connected to the node using a shared SPI bus.

6.1.2 *Sensor Drivers.* Choosing the right sampling strategy for the two sensors creates an intricate trade-off among accuracy, power consumption and scheduling feasibility. For an accurate velocity integration, it is imperative to sample the acceleration at a rate higher than the Nyquist frequency of any significant noise in the input data. Not following this rule causes aliasing effects that map frequency components of the noise to seemingly arbitrary positions in the spectrum. We establish experimentally the sampling frequency of 160 Hz as an optimal value for our system (see details Section 6.1.3). The sampling rate of the compass is also configurable, but special attention is required, because it relates inversely proportional to the achievable sensor resolution and subsequently to the angular sensitivity. We choose to use the sampling frequency of 16 Hz, which provides theoretically an angular sensitivity higher than 0.5° , and additionally matches the control frequency.

Having the accelerometer sampled ten times faster than the compass can generate problems, as both sensors share the same SPI bus of the MSP430 microcontroller. Fortunately, the compass is able to complete its measurements in the background, meanwhile releasing the SPI bus for accelerometer sampling. The only complication is that the compass requires explicit read commands for each of the three axes. As a consequence, the sampling tasks of the two sensors have to be interleaved. The detailed scheduling strategy is explained in Section 6.5.

6.1.3 *Preliminary Sensor Evaluation.* As a first step, we conduct a preliminary evaluation of the sensors, in order to study their accuracy and robustness to external influences. We present in Figure 6 several representative tests for each sensor. Figure 6(a) shows the accelerometer output when released freely to swing as a pendulum (for clarity only the Z axis output is plotted). The graphic follows closely the actual movement and the noise level is low. In contrast, the second experiment presented Figure 6(b) exhibits considerable noise during movement. In this case, the accelerometer is placed on a toy car that accelerates forward and then backward, as suggested by the low-pass filtered signal plotted with dotted line. During stationary periods, though, the sensor output remains stable, with only single bit fluctuations. A similar experiment is performed for Figure 6(c). The result of the velocity integration from Equation 2 is presented at different sampling frequencies supported by the accelerometer. The lower 40 and 80 Hz frequencies exhibit large velocity fluctuations, whereas 160 and 320 Hz provide a velocity progression that correlates well with the experiment. Because the improvement from 160 to 320 Hz is not significant and scheduling becomes more problematic at higher frequencies, we choose the 160 Hz sample frequency for the accelerometer.

Figure 6(d) depicts the output of the magnetic compass when rotating around the Z axis. The signal describes a smooth sinusoid recording the intensity of the magnetic field on the Y axis from $-H_y$ to $H_y$, as expected. In the second test of the compass, the sensor is mounted on a toy car pulled 25 meters through a




(a) Accelerometer in free pendulum swing.

(b) Accelerometer on toy car accelerating forward and backward.

(c) Velocity integration at different frequencies.

(d) Compass rotating around Z axis.

(e) Compass on toy car moving through hallway.

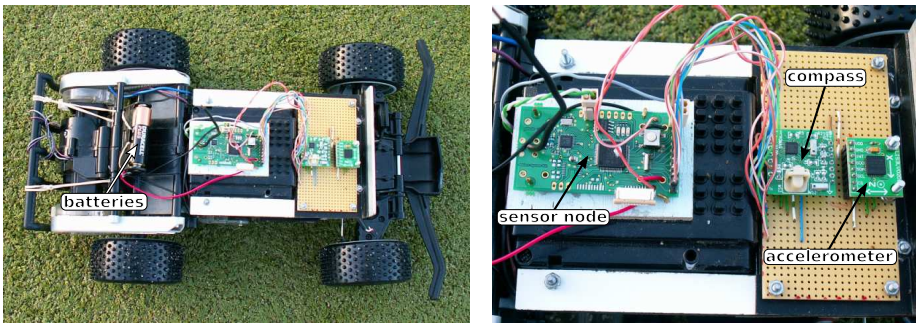
Fig. 6. Sensor evaluation experiments.

straight hallway from our building. The hallway is lined with large concrete pillars containing metallic reinforcement. Figure 6(e) shows the results of two such experiments, which both confirm strong influences of the nearby metals on the compass sensor. [2] The results clearly show that the compass cannot provide useful measurements when moving close to large metal objects, making indoor use infeasible in most cases. Additional experiments show that the magnetic field caused by the follower's motor has a significant influence on the compass sensor as well. This effect requires the compass to be shielded or placed at sufficient distance from any actuator.

### 6.2 Sensor Placement

Sensor placement is an important aspect that can seriously affect the system performance. Both the accelerometer and the compass need to be mounted rigidly to the vehicle frame, in order to minimize the level of vibrations during movement. Because we use different vehicles for leader and follower, the placement of the sensors is not identical. To account for the difference in placement we use the calibration procedure explained in Section 6.3. As indicated by the preliminary evaluation presented in Section 6.1.3, the placement of the compass sensor requires additional

[2]The influence pattern is consistent, as the results of the two experiments correlate well. In Figure 6(e), an artificial offset is introduced between the two signals, in order to enhance visibility.




(a) Top view (b) Detail

Fig. 7. Sensor placement on the follower.

care. During our experiments we have identified the following factors influencing the compass:

—The electromagnetic field created by the car motors, which has a strong effect on the compass output. To alleviate this phenomenon, we use a 5mm shielding metal plate mounted below the sensor board.
—The batteries of the sensor node. Surprisingly, changing the batteries or even swapping the places of the two batteries has a visible impact on the compass readings. The easiest solution is to place the battery pack as far from the compass as possible.
—The radio transceiver. We could not identify the exact cause of this influence. The most plausible explanation is a combination between the electromagnetic field around the antenna and the significant power drain when the node is transmitting or receiving. Our strategy to avoid this problem is to introduce an additional delay between compass sampling and radio tasks (see scheduling details from Section 6.5).

The final sensor placement on the follower car is shown in Figure 7.

### 6.3 Calibration

The calibration procedure compensates for both the magnetic field distortion (due to ferrous or magnetic nearby structures) and the inclination relative to the reference horizontal plane (see also Section 3.2). We drive the vehicles in a circular movement and collect the readings on the $X$ and $Y$ compass axis. Instead of the ideal circle centered in (0,0), we usually get an offset ellipse. From the minimum and maximum recorded values we determine the scale and offset calibration coefficients that project the ellipse back to the desired circle (for the details of this method see [Caruso 2000]).

The calibrated compass values are subsequently computed as:

$$\begin{aligned} H'_x &= H_x\, X_{scale} + X_{offset} \\ H'_y &= H_y\, Y_{scale} + Y_{offset} \end{aligned} \tag{10}$$

### 6.4 Motor Controller

The follower toy car has two electric motors: one for driving and one for steering. To control them, we use a separate MSP430 microcontroller and dedicated circuitry. The two microcontrollers communicate on a separate software $I^2C$ interface, so that the SPI-based dialog with the sensors is not affected. The following control commands are available:

—*Disable*, turns off the motor, the axle can turn freely.
—*Throttle*, controls the intensity of forward or backward acceleration.
—*Steering*, controls the angle of the front wheels.
—*Brake*, short-circuits the rear drive motor to induce inductive drag on the rear wheels.

### 6.5 Scheduling

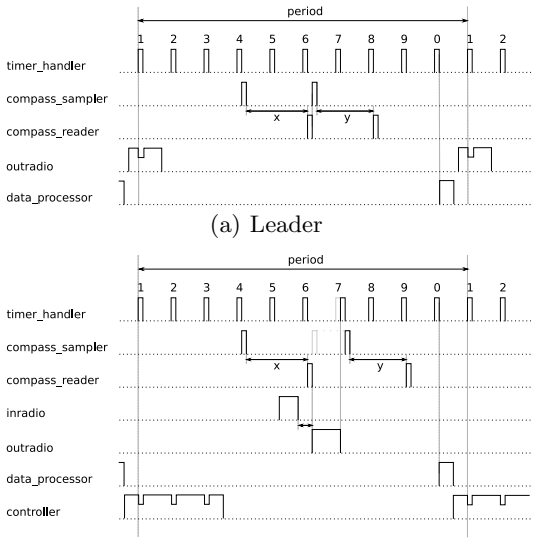
Getting the right scheduling on the limited sensor node is the most challenging part of the implementation. As explained in Section 6.1.2, the sampling tasks of the accelerometer and compass must be interleaved. In addition, the influence of radio operation on the compass described Section 6.2 creates additional dependencies between the tasks associated with these two resources. It is essential therefore to devise the task deadlines in such a way that the execution sequence generated by the scheduler fulfills all these requirements. On the leader, the process is simpler because we know when the radio transmission occurs (at the beginning of a new sampling period) and we do not have a control task (the leader car is driven remotely).

Figure 8(a) sketches the execution sequence on the leader during one sampling period (task durations are not to scale). The heartbeat of the system, termed as the *timer_handler* task, samples the accelerometer at 160 Hz. Being slower, the compass sampling requires two trigger tasks, *compass_sampler* and *compass_reader*, for initiating the measurement and reading the data, respectively. In addition, the X and Y axis cannot be read simultaneously (see also Section 6.1.2). At the end of the sampling period, the *data_processor* task updates the movement status based on the measured data and prepares the message to be transmitted by the *outradio* task.

The scheduling on the follower is complicated by the asynchronous incoming messages from the leader. Moreover, for logging and debugging, we also need an *outradio* task on the follower. As shown in Figure 8(b), it is possible to receive data over the radio (the *inradio* task) while sampling the X axis of the compass sensor. In this situation we try to minimize the time overlap between the compass and radio tasks by introducing two delays: (1) the *outradio* task is initiated only after *compass_reader* finishes and (2) the sampling of the Y axis is postponed until the radio transmission is over. The *data_processor* task has a similar function as on the leader, but its completion triggers the *controller* inference described in Section 4.

### 6.6 Fixed Point Computation

The calculations presented in Section 3 require the implementation of square root and arctangent functions. These are implemented using a simple piece-wise linear representation in a look-up table. The interpolation of values between table en-




(a) Leader

(b) Follower

Fig. 8. Task scheduling.

tries requires fractional number calculations. Since the MSP430 does not provide hardware floating point support, we use fixed point arithmetic exclusively.

### 6.7 Debugging

Debugging the nodes in the field is impractical and therefore we rely on lab reconstruction of failed tests. The follower controller cannot be tested without an environment that responds appropriately to its actions, therefore the input values for the follower controller are obtained from simulation. The leader data recorded from failed field tests is used as input to the simulation to reproduce the situation. In order to find platform-specific problems, we run the software on a gateway sensor node with a serial connection to the PC. The PC provides the input values for the software function under test and reads back the results.

Most of the platform-specific problems relate to different integer size computations. The MSP430 is a 16 bit processor, meaning that the C-compiler generates 16 bit calculations by default. If 32 bit calculations are not specified explicitly, 16 bit calculations are generated for the node and 32 bit calculations are generated for the PC. As a consequence, the software works on the PC and fails on the nodes.

### 6.8 Controller Benchmarking

Using the debugging method from Section 6.7, we determine how fast the controller and its sub-components execute. Table II shows the typical execution times. The time spent in the *fuzzification*, *inference* and *defuzzification* stages for the heading and velocity controllers is identical. Notably, the execution of the whole heading controller takes longer than the velocity controller because a modulo function is used to map the heading error inputs to the range $[-\pi, \pi]$ (see Equation 4).

| Controller Component | Avg. Execution Time |
|---|---|
| **Full Controller** | 2.1 ms |
| **Velocity Controller** | 0.9 ms |
| **Heading Controller** | 1.2 ms |
| **Fuzzification** | 0.1 ms |
| **Inference** | 0.5 ms |
| **Defuzzification** | 0.2 ms |

Table II. Controller benchmark.

## 7. FIELD TESTS AND RESULTS

This section outlines the field experiments performed with the system at our university campus. We first test the behavior of the heading controller and subsequently we evaluate the complete leader-follower system.

### 7.1 Heading Tests

To isolate problems specific to the compass and heading controller, we test the ability of the follower to maintain a specific heading without velocity control or post-processing. The follower starts with a preprogrammed desired heading and changes it with $\pm 180^\circ$ at fixed intervals of time, theoretically following a linear stretch back and forth. This test is performed at our university's running track, oriented approximately 31° NE. The heading change time interval is chosen such that the follower drives over approximately half the length of the track at maximum velocity.

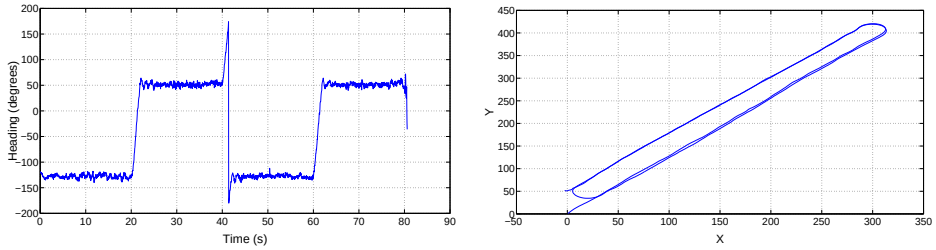
The 180° turns are clearly visible in Figure 9(a). Every 20s, the car takes a fast turn to right and thus remains approximately on the same path, as suggested by Figure 9(b). In this test, the measured heading deviates with 2° on average (with a maximum of 10° ) from the desired heading when the car is not actively turning. This result indicates a good behavior of the fuzzy logic heading controller at full speed. However, the performance can deteriorate when the compass inclination changes during driving due to surface irregularities. Statistics extracted from a total of 24 minutes of testing yield an average deviation of 3° , with a maximum of 39° .

The running track tests show that the compass sensor is sensitive to changes of position, orientation and inclination. Changes in placement require re-calibration. Additionally, when the vehicle gets close to large metal objects or faces surface irregularities, it performs worse due to measurement errors. However, when the sensor is correctly calibrated and not otherwise influenced, the follower maintains a constant heading at fixed throttle.

### 7.2 Complete Tests

We experiment with the complete system at our university's hockey field, oriented 12° NE. This location has the advantage of being a large, relatively flat surface, without metallic structures close by. To evaluate the performance of the system in different situations, we perform tests with three different drive patterns:

—*Linear tests.* The leader drives a 20m straight line.




(a) Heading (b) Integrated position

Fig. 9. Results of a test on the running track.

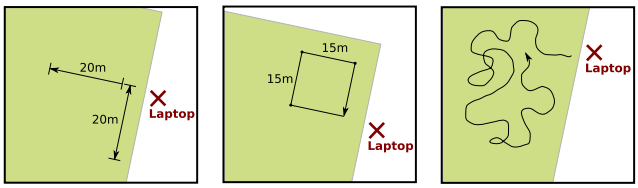
—*Square tests.* The leader drives a square of side length 15m. The finish line is perpendicular to the start line.
—*Random tests*: The leader drives along a random path for 30s with varying velocity.

The three driving patterns are illustrated in the closeups from Figure 10. We perform 20 experiments for each type of test. We vary the driving speed, alternate between driving forward and backward (in random tests), and change the path orientation (in linear tests). In each experiment, the distance between the leader and the follower is 2m at the starting line. At the end of the experiment, we measure the *final heading* of the vehicles (using a regular compass), the *relative distance* between them and the *distance to the finish line* (for linear and square tests). In addition, we log the velocity and heading computed by the two sensor nodes, as well as the throttle and steering outputs of the fuzzy controllers running on the follower. Although not an absolute reference, these logs provide useful information about what is actually happening on the two nodes. Therefore, Figures 11 and 15 represent actual measurements of heading and distance, while Figures 12 - 14 and Table IV present the values estimated by the nodes from the onboard sensors. Details are presented in the following sections.

The logs also provide the means to calculate the packet loss as identified by missing sequence numbers. The results listed in Table III show the average percentage of packets lost during the tests. The random tests show less packet loss, possibly due to the fact that these tests were performed closer to the gateway node.

7.2.1 *Linear Tests.* The linear tests are particularly valuable to assess whether the follower can maintain a constant heading while imitating the velocity of the leader. Figure 11(a) depicts the ending positions and headings of the two vehicles with respect to the finish line (thick horizontal line at 0m). We see that the follower succeeds to copy the leader movement in most of the experiments. The final relative distance is 3.7m on average, which means that the follower deviates with approximately 9cm per traveled meter (the initial relative distance is subtracted when computing this value). The difference in final headings is 8° on average and in 90% of the cases below 11° .

The results presented in Figure 11(a) show only the final situation of each test. Figure 12 gives further insight on how the performance of the follower may actu-




(a) Linear tests (b) Square tests (c) Random tests

Fig. 10. Schematic view of the hockey field tests.

| | Leader | Follower |
|---|---|---|
| **Linear tests** | 1.3% | 1.1% |
| **Square tests** | 0.6% | 1.4% |
| **Random tests** | 0.2% | 0.1% |

Table III. Packet loss.

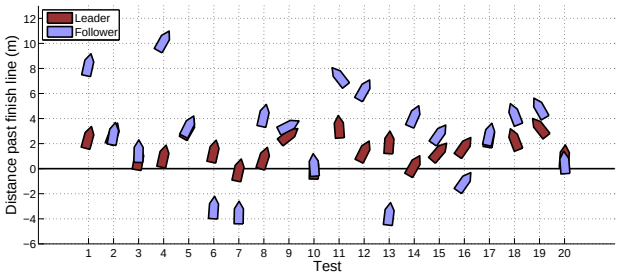
ally vary in a particular experiment (test 10). The top half shows the velocity as integrated by the two nodes and the corresponding throttle control of the follower, while the bottom half shows the changes in heading and the steering output of the heading controller. We make the following observations:

—In the interval 4-8s, the leader accelerates to a velocity higher than what the follower can manage. During the rest of the test, the follower succeeds to keep up. Eventually, the velocity is reset through the motion detection technique explained in Section 3.1. The two cars end up very close to the finish line: 0.1m and 0.3m.
—Although the final difference in heading is only 6°, the follower sways much more in the interval 4-8s. The resulting excessive steering causes extra friction and reduces the velocity of the follower, which remains behind. This shows that the heading controller can become unstable when driving a straight line, although it corrects the orientation in the end.

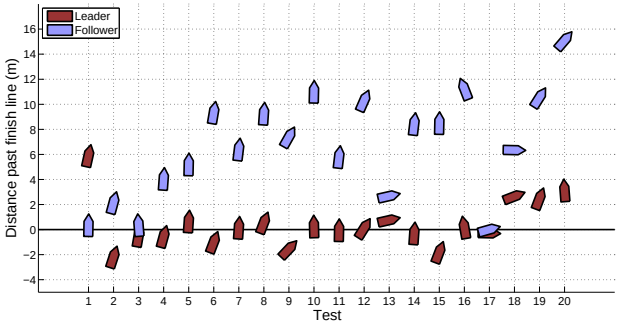
7.2.2 *Square Tests.* Compared to linear driving, during square tests the leader successively steers by 90° and runs over a larger distance (60m in total). Figure 11(b) summarizes the results of the 20 experiments. As expected, the differences in position and orientation increase compared to linear tests. The average final distance between vehicles is 8.3m, meaning that the follower deviates approximately 11cm per traveled meter. The difference in final heading is 12° on average and in 90% of the experiments below 20°.

Figure 13(a) plots the data transmitted by the sensor nodes during one particular square test (test 6). Gaps in the graphs correspond to occasional packet losses at the gateway. We make the following observations:

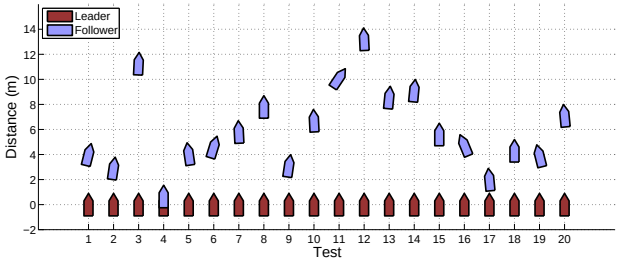
—Both the leader and the follower measure consistently the changes in heading. We can see clear turns corresponding to the rectangle corners at intervals of approximately 5s. A certain amount of swaying by the leader, especially around




(a) Linear tests.




(b) Square tests.




(c) Random tests.

Fig. 11. Finish results of the field tests.

3s and 13s, cannot be avoided because the leader is driven manually. The final difference in heading between the two is 11° .

—The velocity increases correctly in the first 6s, more precisely until the first turn. From this moment, the leader accumulates errors and ends up with a high value at the moment of stopping. The velocity is reset through the motion detection technique explained in Section 3.1. The follower does not exhibit this problem. Its speed remains relatively constant from 6s until 21s, when it finds out that the leader has stopped and, consequently, applies the braking procedure. The follower velocity returns to a value much closer to zero compared to the leader.

—Looking back at Figure 11(b), we understand the effect of error accumulation in

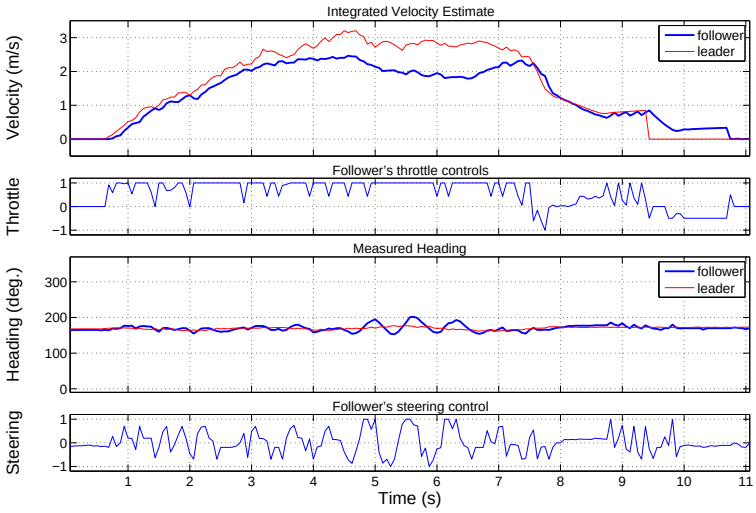



Fig. 12. Velocity and heading of the $10^{th}$ linear test.

the leader velocity. The follower tries to keep up and accelerates to its maximum. The leader detects that it stopped with a certain delay and then informs the follower. The latter also needs some time to brake, so it eventually stops at 10.7m away from the leader.

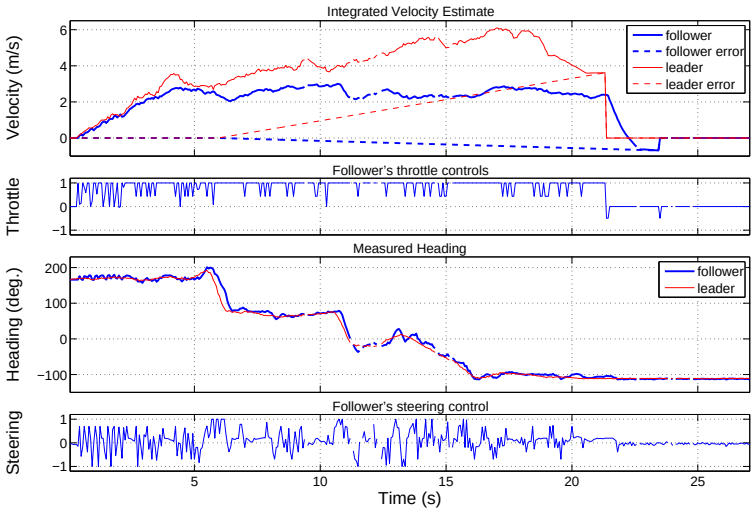
7.2.3 *Random Tests.* The random tests involve steering in random directions and large variations in velocity. Each of the 20 experiments lasts 30 seconds (there is no finish line). Figure 11(c) shows the relative distances and headings of the two vehicles. We obtain an average of 6m final distance between vehicles and a final heading difference of 8° .

Figure 14 depicts one particular experiment (random test 17). In this experiment, the leader drives with high variations in velocity and heading. We make the following observations:

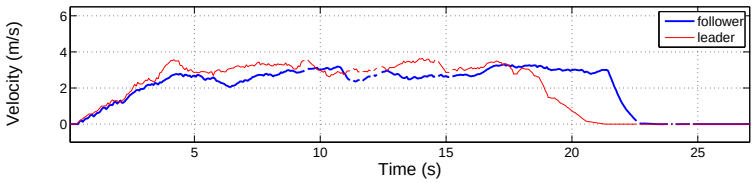
—Both the leader and the follower are responsive to the changes in heading. The abrupt transition just after 16s is in fact the trigonometric turning point from -180° to 180° . At this instance, we notice a delay in the response of the follower steering. The final difference in heading between the two vehicles is 6° .

—Both nodes record clearly the steep acceleration and deceleration. The leader velocity accumulates few errors and returns close to zero at the end of the experiment. The follower has a larger negative offset, which means that he drives faster than he thinks.

—The follower matches the leader velocity during the periods with less steep acceleration, for example around 1s and 16s, but falls behind during high peaks, such as 4s and 12s. The reason lies in the constructive limitations of the follower car, which cannot accelerate as fast as the leader. Overall, the follower moves slower but, since it incurs a delay in braking, it ends up at 2m distance to the leader, which is is approximately the same distance as the cars started at.




(a) Raw data.




(b) Corrected velocity.

Fig. 13. Velocity and heading of the $6^{th}$ square test.

Surprisingly, the random tests show better performance than the linear and square tests with respect to the finishing position, although the distance covered is larger. We identify two factors that can contribute to this result. Firstly, repeated turns in the same direction (as in the case of square tests) may lead to accumulated errors, while random paths and velocities tend to cancel out the errors (see also Section 7.4.2). Secondly, the follower sways less than the leader when steering (random tests involve more steering) and consequently can match the velocity of the leader easier than in straight line sections, where the difference of engine power becomes more visible.

## 7.3 Summary of results

Figure 15 summarizes the results of final differences in distance and heading. The initial relative distance of 2m between cars is not subtracted from these results. The error bars are plotted with 5 and 95 percentiles.

In order to better characterize the continuous performance of the leader-follower system, Table IV provides detailed statistics from all the logged tests. For both velocity and heading, we compute the mean and standard deviation of the absolute error between the two vehicles at any moment in time, as well as the correlation

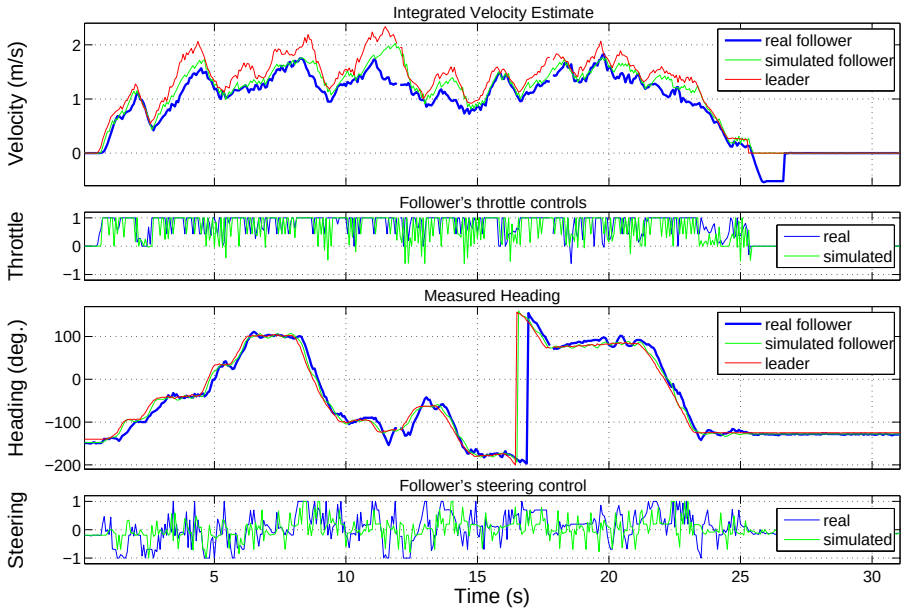



Fig. 14. Velocity and heading of the $17^{th}$ random test (including simulation).

coefficient [3] of the signals recorded in each test. We make the following observations:

—In contrast to the results of ending positions (from Figure 15), the values in Table IV offer just the relative image of what the nodes "think" about their velocity and heading. More specifically, these values include the inherent sensor and integration errors.
—The heading correlation coefficient in linear tests is smaller than in square and random tests (0.64 compared to 0.99). The reason is that the heading varies much less in linear tests, as there are no turns, and therefore any small difference in the compass readings of the two nodes has a strong impact on the correlation.
—The performance in random tests is slightly worse than indicated by the results of ending positions. A detailed analysis of the logs reveals two main reasons. Firstly, during random tests, the leader is subjected to steep acceleration and steering at times. Due to constructive limitations, the follower can not match this behavior, but later on he recovers the difference. Secondly, the vehicles drive longer distances in the random tests than in the other tests and thus accumulate higher errors through acceleration integration. However, their velocity is physically limited, which explains why the actual ending results are better than what the sensor nodes compute.

[3]The correlation coefficient is a commonly used measure of similarity between two signals. A correlation coefficient close to 1 indicates well matching signals.

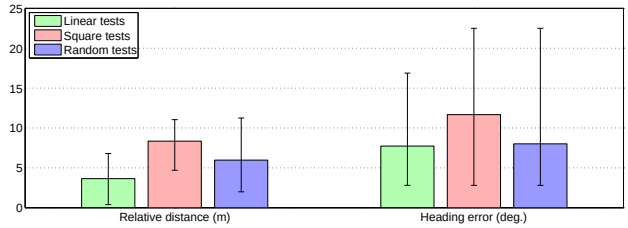



Fig. 15. Summary of results.

## 7.4 Discussion

The results presented so far open several points of discussion. In the following, we briefly examine the effects of error accumulation and error canceling, as well as the accuracy of our simulations with respect to the experimental results.

7.4.1 *Error Accumulation.* In Figures 11(a) and 11(b), we notice that the follower generally ends up further relatively to the finish line than the leader (the random tests are irrelevant in this case, as there is no finish line). As mentioned before, this is caused by the errors accumulating in the acceleration integration process and the inherent delay of the motion detection technique (see Section 3.1). The pertinent question that remains is how does the error accumulation look like in practice?

To answer this question, we go back to the square test depicted in Figure 13(a). The top plot shows an estimation of the accumulated error in velocity for both leader and follower (dotted lines). The error seems to start accumulating when the cars make the first turn, probably because of the small changes in tilt angles. This is also the moment where the difference in the integrated velocity of the two vehicles starts to be significant. We attribute the worse performance of the leader to the soft suspension and the weaker mounting of the sensor. Figure 13(b) shows the corrected velocities, where the linear error estimation is subtracted from the raw data. The matching between the two signals is much better and corresponds to the actual behavior observed in the field.

In conclusion, error accumulation becomes significant due to changes in tilt angles and increases linearly in absolute value. Whether the error can be identified and compensated for at runtime remains an open problem.

7.4.2 *Error Canceling.* Despite the integration errors at the leader and the swaying behavior of the follower, the latter still manages to keep on the right trajectory, especially when the leader is driven with frequent changes in heading. This effect is shown both during the random tests and in an additional test that lasted for more than 10 minutes. This is partly explained by the fact that errors tend to compensate for one another. In addition, the limited velocity of the vehicles improves the overall performance. When the leader does not drive faster than what the follower can manage, the latter follows correctly, even when the estimate of the leader velocity rises beyond its constructive possibilities.

| | Follower vs. Leader | | | Simulated vs. Follower | | |
|---|---|---|---|---|---|---|
| | linear | square | random | linear | square | random |
| **Velocity (m/s):** | | | | | | |
| mean abs. err. | 0.53 | 1.97 | 2.75 | 0.20 | 0.48 | 0.58 |
| std. dev. | 0.55 | 1.42 | 1.93 | 0.24 | 0.39 | 0.34 |
| corr. coeff. | 0.93 | 0.63 | 0.54 | 0.96 | 0.84 | 0.84 |
| **Heading (°):** | | | | | | |
| mean abs. err. | 7.38 | 10.23 | 14.99 | 7.35 | 9.11 | 11.38 |
| std. dev. | 9.79 | 14.23 | 21.06 | 9.78 | 11.96 | 15.45 |
| corr. coeff. | 0.64 | 0.99 | 0.99 | 0.64 | 0.99 | 0.99 |

Table IV. Test error and correlation statistics.

7.4.3 *Simulation validation.* The experimental results offer the possibility to validate and refine the simulation vehicle model presented in Section 5. For this purpose, we use the data recorded during the field tests as input to the simulation framework. Figure 14 shows the simulated velocity and heading of the follower, as compared with the perceived velocity and heading derived from sensor data. We notice that the field results validate the simulation output. The most significant difference occurs in the steering control output, with the real follower steering slower than in simulation. The reason is not the maximum steering angle the vehicle can achieve, but rather the delay between a change in the steering signal from the controller and the actual change in the heading of the vehicle.

Table IV gives a quantitative view on how the simulation matches the real behavior. Similar to Section 7.3, we provide three statistical indicators for each type of test: the mean and standard deviation of the absolute error, and the correlation coefficient of the recorded signals. In general, the real and simulated behaviors match well. The correlation coefficient in linear tests is relatively small for similar reasons as explained in Section 7.3. A detailed analysis of individual tests shows that a more accurate friction model is needed to further enhance our simulations.

## 8. CONCLUSIONS

Our system demonstrates that *WSANs can sense, reason and react as a group with distributed intelligence, without any intervention from the back-end and despite the hardware limitations of sensor nodes*. As concluding remarks, we briefly iterate the main advantages and limitations of our method, name the practical lessons learned and outline the future work directions.

**Advantages**. The method we described enables autonomous mobile team coordination based solely on compact, low-cost wireless sensors and actuators. The solution is not restricted to vehicles on wheels, but supports *any moving entities capable of determining their velocity and heading*. Therefore, it creates a promising basis for both machine-to-machine and human-to-machine spontaneous interactions in the field. Another advantage is the constructive robustness to errors: from any initial or intermediate faulty orientation, the follower is able to return to the correct heading, due to the usage of a global reference system. Furthermore, making use of a fuzzy controller facilitates the implementation on resource constrained sensor nodes and handles robustly the noisy sensor data as well as the rough mechanical capabilities of the vehicles. With respect to the wireless communication, the

leader-follower model is inherently scalable and tolerates transitory packet losses.

**Limitations**. As any inertial navigation solution, errors can accumulate through acceleration integration. Without stopping periods or external reference sources (e.g. GPS), the error can potentially grow to infinity. Additionally, our current implementation does not support dynamic tilt compensation (for example when moving on inclined terrain). The compass utilization is restricted to environments without strong magnetic influences, typically outdoors. With respect to convoy-like applications, the main limitation is the lack of any information and subsequently control of the relative distance between the vehicles (see also future work).

**Lessons learned**. The main lesson learned during the implementation on sensor nodes is that sampling inertial sensors requires a major share of the available CPU power and preferably a multitasking operating system. Scheduling all the tasks is difficult, especially due to the compass sensitivity to various interferences. The large amount of computation involved in inertial navigation algorithms is another potential source of hard-to-identify bugs. Introducing a feedback loop from the sensor node to the simulator proved to be an efficient debugging method for this problem. From field experiments, the most important lesson learned is that the constructive limitations of the vehicles can have both a negative impact (such as the lack of a braking system) and a positive influence (the limited engine power of the follower bounds the effect of error integration in velocity). Compass calibration remains a necessary step for producing accurate experimental results. Driving the toy cars proved to be challenging, as collisions at 30 km/h would destroy the sensors mounted on top. In general, experiments were more time-consuming than expected, required a dedicated, large open field and, last but not least, were dependent on the occasional good weather.

**Future work**. We plan to pursue two directions for future work. One is to incorporate relative distance estimation (for example RSSI information) into the fuzzy controller, in order to apply our solution to convoy-like applications. The other one is to explore group interactions among heterogeneous moving entities (not only vehicles on wheels) using the same general idea.

## REFERENCES


Akyildiz, I. and Kasimoglu, I. 2004. Wireless sensor and actor networks: Research challenges. *Ad Hoc Networks Journal 2,* 4, 351–367.

Allred, J., Hasan, A., Panichsakul, S., W. Pisano, P. G., J. Huang, R. H., Lawrence, D., and Mohseni, K. 2007. Sensorflock: an airborne wireless sensor network of micro-air vehicles. In *SenSys*. ACM, 117–129.

Balch, T. and Arkin, R. C. 1998. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation 14,* 6, 926–939.

Bih, J. 2006. Paradigm shift - an introduction to fuzzy logic. *IEEE Potentials 25,* 1, 6–21.

Burgard, W., Moors, M., Stachniss, C., and Schneider, F. 2005. Coordinated multi-robot exploration. *IEEE Transactions on Robotics 21,* 3, 376–386.

Caruso, M., Bratland, T., Smith, C., and Schneider, R. 1998. A New Perspective on Magnetic Field Sensing. *Sensors 15,* 12, 34–46.

Caruso, M. J. 2000. Applications of magnetic sensors for low cost compass systems. *IEEE Position Location and Navigation Symposium*, 177–184.

Coates, A., Abbeel, P., and Ng, A. 2008. Learning for control from multiple demonstrations. In

*Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, A. McCallum and S. Roweis, Eds. Omnipress, 144–151.

Das, A., Fierro, R., Ostrowski, V. K. J., Spletzer, J., and Taylor, C. 2002. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation 18,* 5, 813–825.

Egerstedt, M., Hu, X., and Stotsky, A. 2001. Control of mobile platforms using a virtual vehicle approach. *IEEE Transactions on Automatic Control 46,* 11, 17771782.

FollowMe. 2008. Demonstration video. http://www.youtube.com/watch?v=ZzWYO5dbo1M.

Fredslund, J. and Mataric, M. 2002. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation 18,* 5, 837–846.

Fritz, H. 1999. Longitudinal and lateral control of heavy duty trucks for automated vehicle following in mixed traffic: experimental results from the chauffeur project. *Control Applications 2*, 1348–1352 vol. 2.

Gaura, E. and Newman, R. 2006. Wireless sensor networks: The quest for planetary field sensing. In *LCN*. 596–603.

Hayat, D. 2002. Traffic regulation system for the future automated highway. *Systems, Man and Cybernetics 3*.

Henkind, S. J. and Harrison, M. 1988. An analysis of four uncertainty calculi. *IEEE Tran. on Systems, Man and Cybernetics 18,* 5, 700–714.

Jadbabaie, A., Lin, J., and Morse, A. 2003. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on 48,* 6, 988–1001.

Liu, S., Tan, D., and Liu, G. 2007. Robust leader-follower formation control of mobile robots based on a second order kinematics model. *Acta Automatica Sinica 33,* 9, 947–955.

Mamdani, E. H. and Assilian, S. 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies 7,* 1, 1–13.

Marin-Perianu, M. and Havinga, P. 2007. D-FLER: A distributed fuzzy logic engine for rule-based wireless sensor networks. In *UCS*. 86–101.

Ögren, P., Egerstedt, M., and Hu, X. 2002. A control lyapunov function approach to multiagent coordination. *IEEE Transactions on Robotics and Automation 18,* 5, 847851.

PNI Corporation. 2008. http://www.pnicorp.com.

Reynolds, C. 1987. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics 21,* 4, 25–34.

Samarasooriya, V. N. S. and Varshney, P. K. 2000. A fuzzy modeling approach to decision fusion under uncertainty. *Fuzzy Sets and Systems 114,* 1, 59–69.

Stipanovic, D., Inalhan, G., Teo, R., and Tomlin, C. 2004. Decentralized overlapping control of a formation of unmanned aerial vehicles. *Automatica 40,* 8, 1285–1296.

STMicroelectronics. 2008. http://www.st.com.

Swaroop, D. and Hedrick, J. 1996. String stability of interconnected systems. *IEEE Transactions on Automatic Control 3,* 41, 349–357.

Tanner, H., Jadbabaie, A., and Pappas, G. 2004. Flocking in teams of nonholonomic agents. In *Cooperative Control*. Springer, 229–239.

Titterton, D. and Weston, J. 2004. *Strapdown Inertial Navigation Technology, 2nd Edition*. Institution onf Electrical Engineers.

Tuck, K. 2007. Tilt sensing using linear accelerometers. Tech. Rep. AN3461, Freescale Semiconductor.

Vidal, R., Shakernia, O., and Sastry, S. 2004. Following the flock. *IEEE Robotics and Automation Magazine 11,* 4, 1420.

Wang, Z., Chen, P., Song, Z., Chen, Y., and Moore, K. 2005. Formation control in mobile actuator/sensor networks. *Unmanned Ground Vehicle Technology VII 5804,* 1, 706–717.

Wang, Z., Song, Z., Chen, P.-Y., Arora, A., Stormont, D., and Chen, Y. 2004. Masmote - a mobility node for mas-net (mobile actuator sensor networks). *Robotics and Biomimetics*, 816–821.

# A. APPENDIX

## A.1 Wireless Communication

The communication protocol required by our approach is straightforward: the leader broadcasts its movement parameters periodically, while the follower just listens for the incoming data packets. This scheme ensures the scalability of the solution by allowing, theoretically, an indefinite number of followers. Nevertheless, a method of logging the behavior of both leader and follower is needed for testing and evaluation purposes. To solve this practical problem, we use the following simple scheme (see Figure 16):

—At the end of each sampling period, the leader transmits its movement parameters.
—Upon receiving the message from the leader, the follower sends its own data.

This method has the advantage of avoiding collisions implicitly. The drawback is that scheduling the sampling and communication tasks on the follower becomes problematic (see Section 6.5).

A gateway node listens to all incoming packets and logs them to a PC. Packet losses are identified based on sequence numbers. In addition to the movement data, the messages sent by the follower include also the controller outputs and raw sensor data. This is a valuable feature because we can reproduce the experiments in the PC-based simulator and correct or tune the controller accordingly.

## A.2 Fuzzy Controller

Figure 17 shows the decision surface of the heading controller. The steering output corresponds to the vertical axis and ranges from -1 (maximum left turn) to 1 (maximum right turn).

## A.3 Simulation

Figure 18 shows the behaviour of the heading controller. The simulation is instructed to change the desired heading 90° to the right every 100 iterations, starting with a heading of 30° . This causes the simulated car to drive an approximately square pattern. At the current instance of time, we see the measured heading, the error $e$, the change in error $\Delta e$ and the steering command.

## A.4 Implementation

Figure 19 shows the two vehicles at our test location, the red car being the leader. The sensor nodes are mounted on top of the two vehicles.

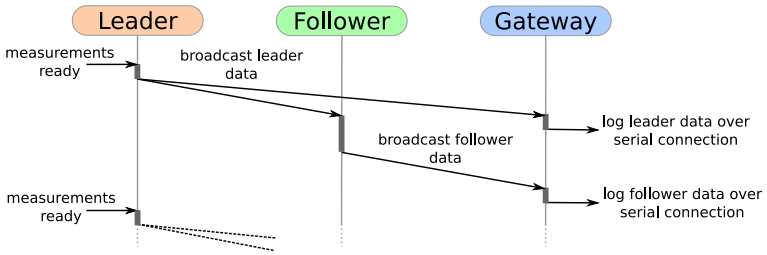



Fig. 16. Communication protocol sequence.

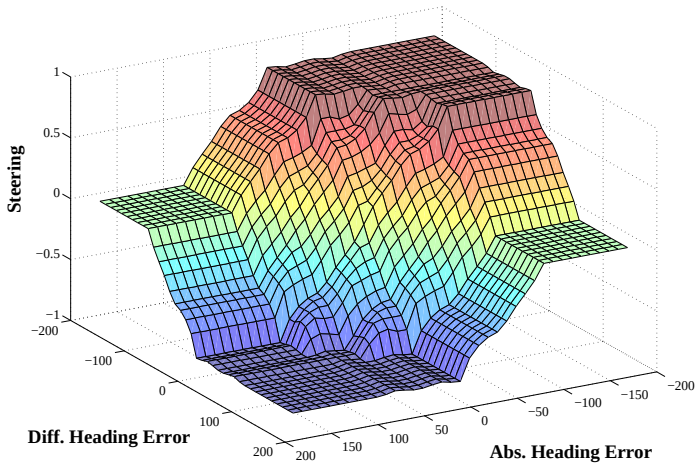



Fig. 17. 3-D view of the fuzzy controller output.

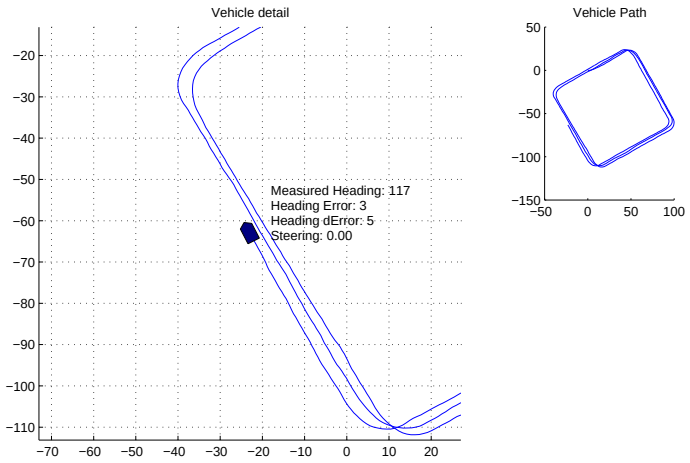



Fig. 18. Simulation view of the heading controller.

Fig. 19. Leader and follower toy cars in the field.