

# Orientation sensing for gesture-based interaction with smart artifacts

Alois Ferscha\*, Stefan Resmerita, Clemens Holzmann, Martin Reichör

*Department of Pervasive Computing, Johannes Kepler University of Linz, Linz, Austria*

Received 31 August 2004; accepted 15 December 2004

Available online 11 May 2005

## Abstract

Orientation sensing is considered an important means to implement embedded technology enhanced artifacts (often referred to as ‘smart artifacts’), exhibiting embodied means of interaction based on their position, orientation, and the respective dynamics. Considering artifacts subject to manual (or ‘by-hand’) manipulation by the user, we identify hand worn, hand carried and (hand) graspable real world objects as exhibiting different artifact orientation dynamics, justifying an analysis along these three categories. We refer to orientation dynamics as ‘gestures’ in an abstract sense, and present a general framework for orientation sensor based gesture recognition. The framework specification is independent of sensor technology and classification methods, and elaborates an application-independent set of gestures. It enables multi sensor interoperability and it accommodates a variable number of sensors. A core component of the framework is a gesture library that contains gestures from three categories: hand gestures, gestures of artifact held permanently and gestures of artifact that are detached from the hand and are manipulated occasionally. An inertial orientation sensing based gesture detection and recognition system is developed and composed into a gesture-based interaction development framework. The use of this framework is demonstrated with the development of tangible remote controls for a media player, both in hardware and in software.

© 2005 Elsevier B.V. All rights reserved.

*Keywords:* Inertial sensors; Orientation tracking; Embodied interaction; Gesture recognition; Tangible user interface

## 1. Introduction

Embodied interaction [21] aims at facilitating remote control applications by providing natural and intuitive means of interaction, which are often more efficient and powerful compared with traditional interaction methods [25]. It is related to Tangible User Interfaces (TUIs), which were introduced by Ishii and Ullmer in [7]. TUIs couple physical representations (e.g. spatially manipulable physical artifact) with digital representation (e.g. graphics and sounds), making bits directly manipulable and perceptible by people. In general, tangible interfaces are related to the use of physical artifact as representations and controls for digital information. Since the physical state of an artifact is usually changed through human manipulation, the position

and orientation of the artifact are ideal candidates for enabling gestural interaction. In particular, orientation can be seen as input to the digital world model of an artifact, as depicted in Fig. 1.

In this context, the present paper addresses the issues of gesture-based interaction for remote control, and the use of orientation sensors for gesture detection and recognition. The main objective is to speed-up practical realization of intuitive gestural interaction by providing application developers and sensor manufacturers with common specifications of information structures and operational requirements that will enable the use of various (different) types of sensors and artifact to control, by gestures, a wide spectrum of applications. The focus on inertial orientation sensors is motivated, on one hand, by the availability of a large array of sensors from different manufacturers. On the other hand, apparently the use of orientation for gestures has received far less attention from the research community than the use of position. For gesture sensing and recognition, the orientation is commonly encountered as an additional information to position. However, orientation sensing has advantages versus position sensing that are important in various circumstances, which are discussed in this paper.

\* Corresponding author. Tel.: +43 732 2468 8555; fax: +43 732 2468 8426.

*E-mail addresses:* [ferscha@soft.uni-linz.ac.at](mailto:ferscha@soft.uni-linz.ac.at) (A. Ferscha), [resmerita@soft.uni-linz.ac.at](mailto:resmerita@soft.uni-linz.ac.at) (S. Resmerita), [holzmann@soft.uni-linz.ac.at](mailto:holzmann@soft.uni-linz.ac.at) (C. Holzmann).

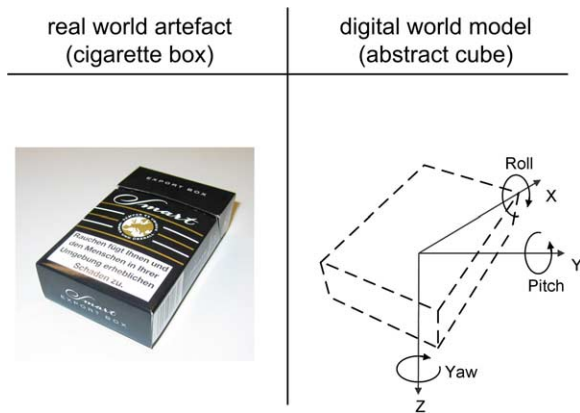


Fig. 1. Orientation as input to digital world model of artifact.

To achieve the above objective of interoperability, first an application-independent set of gestures is presented, that considers a meaningful distinction of gestures as related to the involved artifacts: hand gestures, gestures of artifact held permanently, or worn by the human, and gestures of artifact that are detached from the human and are manipulated occasionally. Then, based on this alphabet of gestures, a general framework for gesture recognition is introduced. This framework is independent of sensor technology and classification methods, it enables sensor interoperability, and it accommodates a variable number of sensors. An implementation and an application of this framework are presented, where two artifact are used for controlling a media player.

According to the Oxford Concise Dictionary, a Gesture is a ‘movement of a limb or the body as an expression of thought or feeling’. In the research literature, gestures are investigated for various purposes such as: sign language recognition [22,25], manipulation of virtual objects [9], animation of avatars [10], and control of applications and devices [1,8]. The scope of this paper falls within the last category.

Gesture sensing and classification is subject to a large body of literature. Computer vision approaches have the main advantage of being non-intrusive (the user needs not wear sensors). In general, this is valid for controlled environments, although there are some exceptions [5,16,18,22]. In computer vision, the problem of orientation tracking is harder than the position tracking. Another method for sensing gestures are instrumented gloves, which measure hand and finger parameters [25]; however, it is not possible to detect the absolute orientation of an artifact the user holds with such a glove. The sensing systems based on inertial sensors are more suitable for autonomous and mobile systems. Orientation sensors have been considered for gesture detection in [11,14,23,26]. This paper presents a more comprehensive set of orientation-based gestures that is not limited to hand gestures. The specific issue of Euler singularity in the orientation data is tackled in a different manner than the ones presented in

[11,23] that does not require a priori assumptions on the dynamics of the sensor data, and it is therefore, sensor-independent.

Several papers have proposed gestural command sets and guidelines for obtaining such gestures [1,12,15]. In these papers, each of the proposed gesture set is application-dependent. It is argued in [15] that gesture sets should be application dependant. In contrast, this paper presents an application-independent collection of gestures and the usage of such a set within a general framework for gesture recognition. The work reported in [2] deals with a framework for gesture recognition that is targeted to a specific inertial sensor.

This paper is structured as follows. Section 2 presents a discussion on the use of orientation sensors for gesture recognition, as well as a classification of gestures according to the involved artifacts. Section 3 describes the gesture alphabet, at its current version. Section 4 deals with the framework for gesture recognition. In Section 5, an implementation of the framework is outlined, together with a sample application. Section 6 concludes the paper with discussions and future work.

## 2. Orientation sensing

This section focuses on the use of orientation information for gesture detection. In this paper, we look at the case where orientation is provided by inertial sensors, which can be easily embedded into mobile platforms [2]. For the remaining of the paper, unless otherwise specified, by ‘orientation sensor’ we shall mean ‘inertial sensor of orientation’.

From the viewpoint of the underlying technology, two different types of inertial sensors are distinguished, namely accelerometers and gyroscopes. Accelerometers are often used for gesture recognition as they are available at low cost and provide relative position information just by integrating the sensor values twice. However, they do neither provide drift-free relative position information nor absolute position information within a common reference frame without the need for outside support. Gyroscopes on the other hand measure the angular rate and thus give relative orientation information; as with accelerometers, this data is not drift-free and can thus not be used for detecting absolute orientation information. However, by using additional sensors for measuring the earth magnetic field as well as the gravitation, it is possible to provide orientation information which is absolute in the earth reference frame, without the need for a certain infrastructure. Fig. 2 presents a sample of orientation sensors currently available.

Since orientation data from many orientation sensors is given as Euler angles, it is of interest to see how this representation can be used towards gesture classification. The main impediment for using the Euler angles directly as data for training and classification is the Euler singularity. Depending on the definition of the Euler angles for






Sensors	Ascension 3D-Bird	O-Navi Gyrocube 3A	Intersense InertiaCube 3	MicroStrain 3DM	XSens MT9-B
Internet	<a href="http://www.ascension-tech.com">http://www.ascension-tech.com</a>	<a href="http://www.o-navi.com">http://www.o-navi.com</a>	<a href="http://www.isense.com/">http://www.isense.com/</a>	<a href="http://www.microstrain.com">http://www.microstrain.com</a>	<a href="http://www.xsens.com/">http://www.xsens.com/</a>
					
Angular Range (Yaw / Pitch / Roll)	$\pm 180^\circ / \pm 180^\circ / \pm 90^\circ$	$\pm 150^\circ / \pm 150^\circ / \pm 150^\circ$	$\pm 360^\circ / \pm 360^\circ / \pm 360^\circ$	$\pm 360^\circ / \pm 360^\circ / \pm 360^\circ$	$\pm 360^\circ / \pm 360^\circ / \pm 360^\circ$
Angular Resolution	0,2°	n/a	0,03°	0,1°	0,05°
Maximum Angular Rate	1000°/s	400°/s	1200°/s	n/a	900°/s
Update Rate	160Hz	40Hz	180Hz	100Hz	25-512Hz (selectable)
Interface	RS232	RS232	RS232	RS232 (RS485)	RS232
Power	5VDC at 100mA	5VDC at 27mA	6VDC at 40mA	5,2VDC-12VDC at 65mA	5,5VDC at 40mA
Size [mm]	34,0 x 27,4 x 25,4	30,5 x 31,8 x 15,6	26,2 x 39,2 x 14,8	65,0 x 90,0 x 25,0	39,0 x 54,0 x 28,0
Weight	28,4g	7g	17g	74,6g	35g

Fig. 2. Orientation sensor survey.

a particular sensor, this means a  $180^\circ$  jump in at least one of the three angles when the other angles reach certain values. For example, in the case of the InertiaCube3 from InterSense [6] the yaw and roll angles jump with  $180^\circ$  whenever the pitch approaches  $+90^\circ$  or  $-90^\circ$ . Such a strong non-linearity affects the accuracy of the classification process. Moreover, one cannot derivate the data in order to obtain angular velocities. The authors in [11] and [23] discuss several ways for eliminating this discontinuity, and propose an approach based on the assumption of limited dynamics, where also both original and transformed data are available for classification purposes. We describe in the sequel a method that makes no assumptions on the dynamics, and which yields a three-dimensional vector as data for classification (which is the minimum required to fully describe orientation in 3D).

$$A = \begin{bmatrix} \cos(P)\cos(Y) & -\cos(P)\sin(Y) & \sin(P) \\ \cos(R)\sin(Y) + \sin(R)\sin(P)\cos(Y) & \cos(R)\cos(Y) - \sin(R)\sin(P)\sin(Y) & -\sin(R)\cos(P) \\ \sin(R)\sin(Y) - \cos(R)\sin(P)\cos(Y) & \sin(R)\cos(Y) + \cos(R)\sin(P)\sin(Y) & \cos(R)\cos(P) \end{bmatrix}.$$

The Euler angles can be used to determine the orientation matrix that describe the orientation of the local frame

$$\begin{bmatrix} x_g \\ y_g \\ z_g \end{bmatrix} = \begin{bmatrix} \cos(P)(\cos(Y) - \sin(Y)) + \sin(P) \\ \cos(R)(\sin(Y) + \cos(Y)) + \sin(R)\sin(P)(\cos(Y) - \sin(Y)) - \sin(R)\cos(P) \\ \sin(R)(\sin(Y) + \cos(Y)) + \cos(R)\sin(P)(\sin(Y) - \cos(Y)) + \cos(R)\cos(P) \end{bmatrix}.$$

(attached to the sensor) with respect to a reference (or global) frame (usually earth-related). The orientation matrix itself is not a good candidate as a representation of data for classification, since it contains redundant information (9, entries as compared to a minimum of 3). Let  $A$  denote the rotation matrix. Given the local coordinates  $[x_l, y_l, z_l]$  of a point in the three-dimensional space (which are the point's coordinates in the sensor frame), the global coordinates

$[x_g, y_g, z_g]$  of the same point are given by

$$\begin{bmatrix} x_g \\ y_g \\ z_g \end{bmatrix} = A \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}$$

The data that we use for classification represents the global coordinates of a specified point that is fixed in the local coordinates (in other words, a point that rotates together with the sensor).

While most of the computer graphics libraries assume a right-hand reference frame, in the particular case of InertiaCube3, the fixed reference frame is a left-hand frame. In this situation, the rotation matrix corresponding to the Euler angles Pitch ( $P$ ), Roll ( $R$ ), and Yaw ( $Y$ ) is the following:

For simplicity, the local point  $[1, 1, 1]$  can be used, leading to the following transformation:

### 3. The Gesture Library

Using a standard, application-independent, set of gestures has advantages and disadvantages. While many authors consider implicitly or explicitly that gestures should be application-dependant, to our knowledge there are few that propose application-independent gestural command sets [2]. The main reason seems to be that not all gestures

are appropriate for all applications. We consider that a fairly large set of gestures can be employed for controlling fairly large classes of applications. For example, there are many applications that accept navigation commands (next, previous, start, stop, etc.); on the other hand, there are hand gestures that are intuitive for such commands, and this makes them good candidates for standardization. An application can offer the user the flexibility of mapping gestures to commands accepted by the application.

### 3.1. Categories of gestures

The individual gestures in this library are organized in three categories, according to the artifacts involved in the gestures. This classification is independent on sensing methods.

1. *Gestures of the user's hand.* The user is not particularly attentive at the hand, but just moves the arm and the hand. The continuous movement is sensed by an inertial sensor which is directly attached to the user's hand (e.g. a wrist watch or a dedicated sensor device like the EDEMO wireless hand gesture recognition system using acceleration signals [24]). Such hand-gestures are performed in a limited orientation domain (due to the human anatomy) and they are highly dynamic. According to [18], hand gestures can generally be classified in static (e.g. a certain posture of the hand) and dynamic (i.e. a time-sequence of static gestures which leads to further parameters like changes in velocity, time and space [20]). Dynamic gestures can then be classified in unintentional movements and gestures, which can again be manipulative (used to act on objects, e.g. movement and rotation) or communicative (i.e. having an inherent communication purpose). Finally, communicative gestures can be either acts (i.e. gestures that are directly related to the interpretation of the movement itself) or symbols (i.e. gestures with a linguistic role).
2. *Gestures of artifact held or worn by the user.* The main difference from the first category is that the user is not only able to perform gestures with his hand, but he can also change the orientation of the artifact in hand. Examples for artifact in this category are mobile phones equipped with orientation sensors or smart objects like iCube [19], a cube-based TUI with integrated accelerometers and means for identification. Gestures in this category are characterized by low to high dynamics and they do not suffer from a limited orientation domain, as it is the case in the first category.
3. *Gestures of artifact that are manipulated on demand or occasionally.* This category pertains to artifact that are detached from the human body and are moved from time to time (e.g. chair, table, door). Gestures in this category are amenable to cases where the static aspect of orientation predominates in the interaction.

The following table summarizes the main differences between these three categories. They imply differences in

Table 1  
Categories of gestures

Category	Timing	Type of object	Dynamics of movement
Hand of the user	Continuously	Hand	High
Object the user holds permanently	Continuously	Artifact (small)	Static-high
Object manipulated occasionally	Occasionally	Artifact (large)	Static-low

the implementation of classification and segmentation algorithms, which is discussed in Section 5.

With respect to these categories of gestures, we consider absolute orientation information as very important for recognizing gestures, as it leads to several advantages compared to relative orientation and position information, respectively. First, it can be used to distinguish different static states of artifact (e.g. which side of a cube is up) as well as different directions (e.g. north or south), which is crucial especially for recognizing gestures in the third category. Moreover, as multiple artifact have a common reference frame, it is possible to take into account their relative orientation without explicit coordination or communication. This enables, for example, recognition of static hand gestures (postures) performed with both hands (Table 1).

### 3.2. Gesture definitions

A gesture is represented as a pair

$$G \equiv (\langle \text{Object} \rangle, \langle \text{Name} \rangle),$$

where  $\langle \text{Object} \rangle$  is a generic denomination of the object that performs the gesture (e.g. hand, artifact type), and  $\langle \text{Name} \rangle$  is a unique identifier for the gesture. A gesture can be defined in two ways:

- As a sequence of relevant states of the object to which the gesture is attached.
- As a concurrent composition of two or more gestures.

By specifying that gesture  $G_3 = (O_3, N_3)$  is a composition of gestures  $G_1 = (O_1, N_1)$  and  $G_2 = (O_2, N_2)$ , we mean the following: if  $G_1$  and  $G_2$  occur simultaneously and the (same) object  $O_3$  contains the objects  $O_1$  and  $O_2$ , then gesture  $G_3$  must be issued instead of  $G_1$  and  $G_2$ . For example, the gesture (person, Applause), is a composition of (left-hand, Wave\_vertical) and (right-hand, Wave\_vertical). Clearly the composition refers to the situation where both hands are of the same person. This must be detected by any implementation that uses the Gesture Library.

### 3.3. Initial set of gestures

The following gestures are determined solely by orientation information. Clearly, position information is important in order to distinguish distinct gestures with the same

orientation pattern. Using only orientation in the initial specification is intended to reveal the potential of orientation information for gestures.

### 3.3.1. Hand gestures

Each gesture in the following list starts from a standing initial position where the upper arm lies alongside the body, the forearm is horizontal, the hand is open, with the palm facing up. When performing the gestures, the upper arm stays always relaxed alongside the body.

1.  $G_1 \equiv (\text{Right Hand, Take})$ . Move the forearm to the left and up, until the hand touches the body near the heart. At the same time, rotate the wrist about the forearm in counterclockwise direction with about  $90^\circ$ . The palm should be always in continuation of the forearm. Thus, the palm will actually touch the body. Then return to the initial position by doing the reverse movement. This gesture is presented in Fig. 3(a).
2.  $G_2 \equiv (\text{Right Hand, Commit})$ . Move the forearm vertically up as much as conveniently possible and rotate at the same time wrist about the forearm in counterclockwise direction with about  $180^\circ$ . When the forearm is about vertical, the palm should face to the front. The palm should be always in continuation of the forearm. Then return to the initial position by doing the reverse movement. This gesture is presented in Fig. 3(b).
3.  $G_3 \equiv (\text{Right Hand, Throw})$ . Rotate the forearm horizontally to the right with about  $30\text{--}40^\circ$ , or as much as conveniently possible. At the same time, rotate the wrist about the forearm counterclockwise with about  $180^\circ$ . When the forearm is maximally to the right, the palm should face down. The palm should be always in

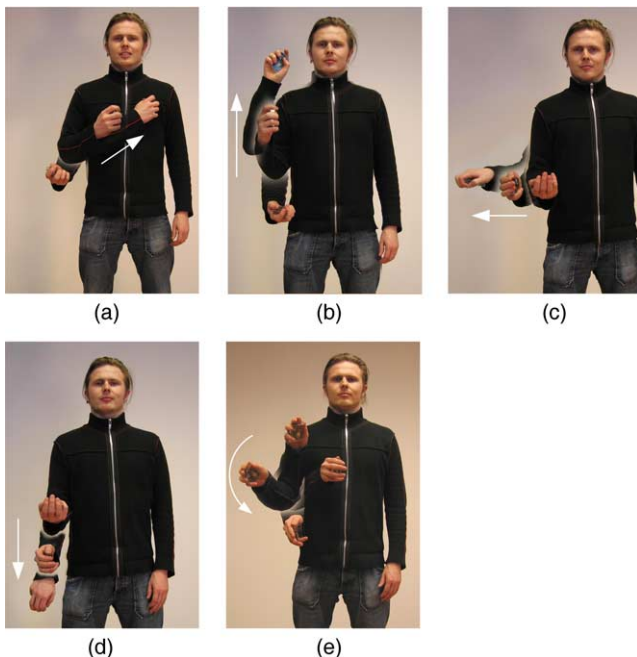


Fig. 3. Hand gestures.

continuation of the forearm. Then return to the initial position by doing the reverse movement. This gesture is presented in Fig. 3(c).

4.  $G_4 \equiv (\text{Right Hand, Leave})$ . Move the forearm down with about  $90$  degrees. At the same time, rotate the wrist about the forearm counterclockwise with about  $90$  degrees. At this stage, the forearm should lie alongside the body, with the palm facing to the left. Then return to the initial position by doing the reverse movement. The gesture is presented in Fig. 3(d).
5.  $G_5 \equiv (\text{Right Hand, Move\_On})$ . This gesture involves rotating the forearm and the palm such that the tip of each finger describes approximately a circle in a vertical plan. The center of the circle should correspond to the initial position. After 2, 3, or 4 complete rotations, the hand comes back to the initial position. Fig. 3(e) illustrates this gesture.

### 3.3.2. Gestures of artifacts held or worn by the user

For the beginning, we considered several gestures of box and cylinder artifacts. A box is defined by six adjacent faces, where two faces are adjacent if they have a common edge. A cylinder is defined by two planar surfaces (1 and 2) and a circular surface.

6.  $G_6 \equiv (\text{Six\_Face\_Box, Shake\_Face1})$ . Holding the box in hand, with the face number 1 up, shake the box twice in a vertical plane. For enabling the recognition of this gesture, the box manufacturer must specify the ordering of the box's faces (from 1 to 6). Also, the sensor module associated to the box must be able to determine the absolute orientation of the box. The gesture is illustrated in Fig. 4(a).
7.  $G_7 \equiv (\text{Six\_Face\_Box, Rotate\_Face1\_Face2})$ . With the box lying on a horizontal surface, with face number 1 upwards, rotate the box with a continuous movement in order to bring the adjacent face number 2 up.
8. 30. Define  $G_8\text{--}G_{30}$  similarly to  $G_7$ , by considering all pairwise combinations of adjacent faces. Two of such gestures are illustrated in Fig. 4(b) and (c).
31.  $G_{31} \equiv (\text{Cylinder, Rotate\_10\_Clockwise\_Face1\_Up})$ . A cylinder with face number 1 upwards is rotated with 10 degrees clockwise about the axis that is orthogonal to both face 1 and face 2. The manufacturer of the smart cylinder must specify which of the two planar faces is number 1 and which is number 2.

### 3.3.3. Gestures of detached artifacts

32.  $G_{32} \equiv (\text{Chair, Rotate\_15\_Clockwise})$ . A chair is rotated about its vertical axis with  $15^\circ$  clockwise.

33.  $G_{33} \equiv (\text{Chair, Rotate\_15\_CounterClockwise})$ . A chair is rotated about its vertical axis with  $15^\circ$  counter clockwise. These gestures are illustrated in Fig. 5.

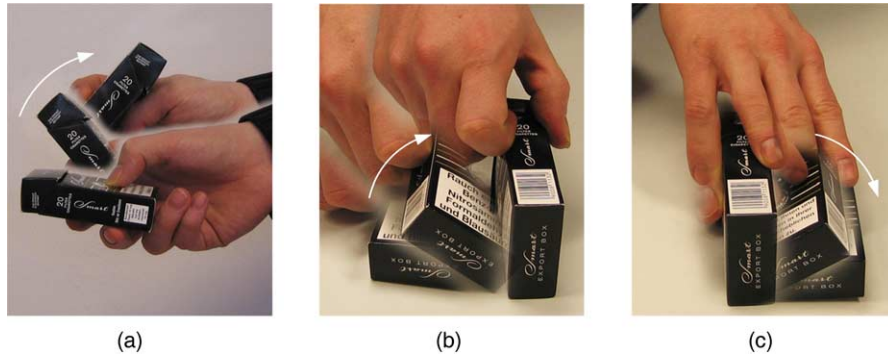


Fig. 4. Gestures of artifacts held in hand.

#### 4. A framework for gesture recognition

In this section, we propose a framework for gesture recognition that is based on the Gesture Library described above. The main goal is to provide gestures from the standard gesture set as input to interested applications, via an application programmer interface (a Gesture API). Our framework specification is independent of sensor technology and classification methods. Before giving the formal definitions, let us look at a simple example.

##### 4.1. Use-case: smart-phone

Clara buys a new Smart-Phone. This phone has an inertial sensor in it. In some cases, the phone is able to detect its current place: in Clara's hand, in her pocket, on a table, etc. This detection can be achieved by analyzing the dynamics of the sensor data, or by using various proximity detection mechanisms. When in hand, the phone is able to recognize certain hand gestures, which are specified in the phone's documentation. Thus, Clara uses gestures from this set in hand in order to control various devices that accept commands from her phone.

Later, Clara receives a smart watch as a birthday present. Among other things, the watch has an inertial sensor in it, that can be used to recognize certain hand gestures, which are specified in the watch's documentation. Since the watch has limited computational power, the sensor can only send its data via a wireless connection. However, the watch comes with a CD containing a lightweight software module that can run e.g. on a mobile phone. The software can process the watch sensor data. Clara installs the watch software on her smart phone. Now, whenever the phone is in the proximity of the watch, the watch software is executed on the phone. Thus, the phone is able now to recognize certain gestures of Clara's left hand (from the watch sensor), and, whenever the phone is in the right hand, also certain gestures of Clara's right hand. Assume now that the same classification method is used for recognition of gestures from both sensors (which makes sense, since both are inertial and are used for hand gestures).

The straightforward approach of having two stand-alone

applications running in parallel on the phone (one dealing with the watch gestures and the other with the phone gestures) has significant drawbacks:

- Each application performs basically the same type of recognition. Thus, computational resources are utilized poorly.
- A composed gesture of the two hands cannot be dealt with, unless another (specialized) application is used in this respect. For example, Clara could use the 'applause' gesture, which is composed from a gesture of the left hand, and, respectively, of the right hand. Each of the two individual gestures has its own consequence in terms of the actions that Clara has associated to them. To use the applause gesture, and map it to a third action, one needs to coordinate the two applications (which means that they are not independent anymore).

This example will be used for illustration purposes throughout this section.

##### 4.2. Architecture overview

We propose a conceptual framework for gesture recognition (cf. Fig. 6) that has the following properties:

- Employs a standard set of gestures (the Gesture Library).
- Allows for a variable number of sensor or sensor systems, which can come from different manufacturers.



Fig. 5. Gestures of detached artifacts.

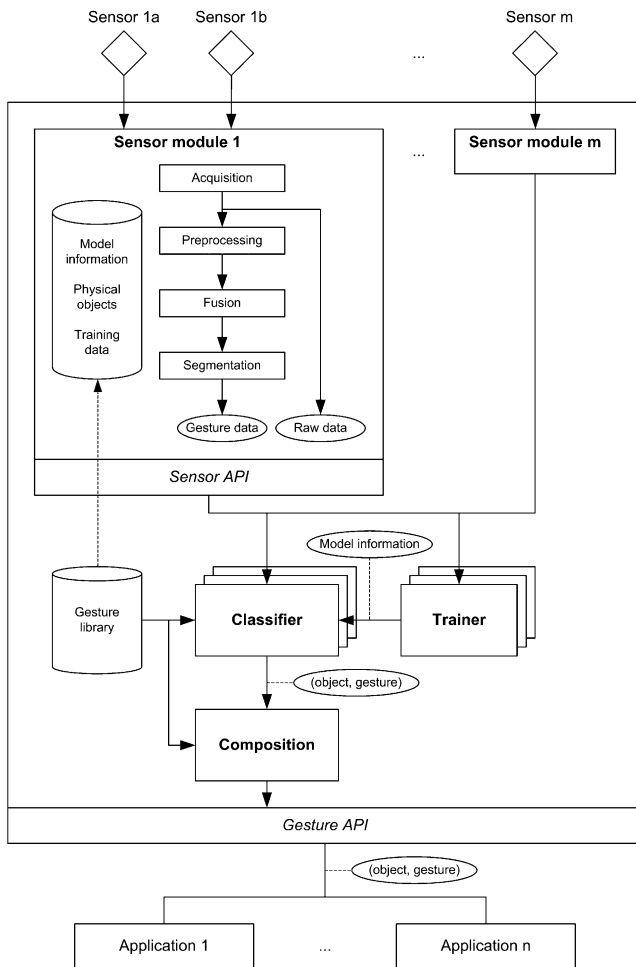


Fig. 6. The framework architecture.

Sensors can occur or disappear at any time during the operation.

- Does not enforce particular methods for data filtering, segmentation, training, and classification.
- Provides a unified structure for recognition of gestures independent of the objects related to the gesture (e.g. hand, artifact, large objects).
- Provides gesture composition for gestures within the Gesture Library.
- Is adaptable to new sensors and classification methods.
- Enables reusability of classification software.

The framework architecture is presented in Fig. 6.

#### 4.3. The sensor module

The sensor module is a processing unit associated to one or more sensors. The main tasks of a sensor module are (i) data acquisition, (ii) pre-processing: filtering, normalization etc. (iii) fusion of sensor data, if more than one sensor is dealt with by the same module, (iv) gesture segmentation, and (v) notification of state changes.

The information structure of the sensor module is exposed by a Sensor API, and it contains mainly the following:

- A description of the physical objects to which the sensors are attached. This information should describe the object types as they are specified in the Gesture Library. The containment relation among these objects must be specified. There should be at least one object containing all sensors. That object must have an identification tag. For example, consider one inertial sensor embedded in a watch. A description of the physical objects to which the sensor is attached is (person, [name], (left hand, (wrist))). Another example is when the same sensor module deals with two sensors, one in a watch and the other one in a ring. When the same person wears the watch at the left hand and the ring at the middle finger of the right hand, the description will be (person, [name], (left hand, (wrist)), (right hand, (middle finger))).
- A database of supported gestures, which specifies the classes in which the sensor data can be distinguished. Each class (gesture) is identified by one of the entries in the Gesture Library. Classes can be defined in two ways:
  - By providing model information, which is specific to classification methods. For example, a sensor module can be trained (e.g. by the manufacturer) using Support Vector Machines. A model or a set of models is obtained, which is stored in the database. In addition, the same sensor module can be trained using a Hidden Markov Model method, in which case the module will also contain the Markov models obtained from the training.
  - By providing directly the training samples. In this case, for each class, several examples of sensor data are provided. The main advantage of this approach is that no classification method is imposed from the outset. The disadvantage is that automated training must be performed whenever the sensor module is instantiated. This makes it difficult to fine-tune the resulting model, as opposed to manual training, when experts usually adjust model parameters in order to obtain high accuracy.
- The raw sensor data represents the current value of the sensor data.
- The gesture data is a time series of sensor values that is relevant for the classification process. In this respect, the gesture data can represent:
  - A gesture, if the sensor module provides gesture segmentation.
  - Sensor data that spans a time window of suitable size, if the module is unable to perform segmentation. In this case, the data is usually updated at regular time intervals.

The containment relationship of objects can change dynamically. For example, person A has a phone in the right hand, and person B has a watch at the left hand. Person A can change the phone's place from the right hand to the left one, or

it can hand in the phone to person B. Various techniques can be used to detect an object's relative place in the environment. These can be based on proximity detection and/or analysis of the dynamics of the inertial sensor data [13].

#### 4.4. The classifier module

Upon instantiating a new sensor module, the framework searches for an appropriate classifier that is able to distinguish the sensor data according to the model information provided by the sensor module. If no classifier is found that matches this model information, then the framework checks if the sensor module provides training data. If so, a (default) trainer is dispatched, that generates the model information based on the training data. Then, an appropriate classifier is started.

A classifier is notified whenever the gesture data of a compatible sensor changes. Then, based on the data provided by the sensor, the classifier decides which gesture (from the Gesture Library) was performed.

The association between sensor modules and classifiers can be many to one. Thus, the same classifier may be able to serve distinct sensor modules, especially in cases where the gesture frequency is low (e.g. gestures of the third category), and/or when time constraints are not tight and simultaneous gestures can be queued for classification by the same classifier.

#### 4.5. The trainer module

A framework implementation can provide default trainers and classifiers. These are used for sensor modules that provide training data, whenever:

- The sensor module has no model data.
- The sensor module has model information, however no classifier can be found that is compatible with that model.

Thus, a default trainer is employed for generating models that can be subsequently used by an appropriate (default) classifier. In this case, the field 'Model information' of the sensor module database may contain suggestions for classification methods that are suited for that particular sensor. The framework can then take into account these suggestions in finding a suitable default trainer and a classifier (training is performed only once for a given sensor module).

#### 4.6. The composition module

The primary task of this module is to detect composite gestures, and to avoid ambiguity in the gestural information which is being provided to applications. For illustration, consider the example given at the beginning of this section, and look at the 'applause' gesture. This can be seen as a type of simultaneous waiving of the two hands (two elementary gestures). When Clara had only the phone, she assigned the waiving of the right hand to the navigation command 'Next'. Also, she assigned the waiving of the left hand to the navigation

command 'Previous' (because it is similar to 'Next', but in the opposite direction). Having both the watch and the phone, Clara assigns the gesture 'applause' to the command 'Last'. If ambiguities are not addressed, the gesture API will trigger both commands 'Next' and 'Previous' at the same time.

In this case, the composition module receives the information that the two elementary gestures have been executed simultaneously. By looking at the 'physical object' information, one can detect that the two gestures involve the same object (a human). Also, the Gesture Library contains a gesture that is composed of the two elementary gestures (in this case, the 'applause' gesture). Thus, the composed gesture will be issued to the interested applications, instead of the component gestures.

Gesture composition based on the containment relation among physical objects can be achieved by representing the relation as a tree, where a vertex  $v$  is a descendant of a vertex  $u$  if and only if the object represented by  $u$  contains the object represented by  $v$ . In other words,  $v$  is a physical component of  $u$ . The leaves of the tree represent the 'smallest' objects to which the sensors are attached, where elementary gestures may be generated. Upon occurrence of simultaneous elementary gestures corresponding to distinct leaves of the same tree, composite gestures are found by looking at common ascendants of the leaves and searching the Gesture Library for objects corresponding to these ascendants.

As sensors appear in the system, the containment trees are constructed according to the information about the physical objects that is provided by the sensor modules. The containment relation is dynamical: an object can change 'ownership' on the run. A change in this respect is reported by the corresponding sensor module and leads to a change in the corresponding tree. For illustration, consider the above example. Fig. 7(b) illustrates the representation of Clara's mobile phone when the phone is in her right hand. When the sensor module corresponding to the watch is created, it will send the representation depicted in Fig. 7(a). The composition module will merge the two elementary trees into the tree shown in Fig. 7(d). Consider now that Clara moves the phone from the right hand to the left hand. Assuming that the phone sensor module is able to detect this change, the module changes its 'physical object' information to the one depicted in Fig. 7(c), which leads to changing the 'Clara' tree in the composition module to the one depicted in Fig. 7(e).

The Composition module may implement other relations among objects. Consider, for example, the 'physical attachment' relation. When artifact A is placed on top on artifact B and both are manipulated as a single object, the individual gestures coming from each of the two artifact may need to be composed. The operation of the Composition module is essentially based on the specification of composite gestures in the Gesture Library. For each composite gesture, the Gesture Library must specify the relation among the objects involved. The sensor modules must use the standard object description specified in the Gesture Library. This allows the composition module to detect the relevant relations among objects with

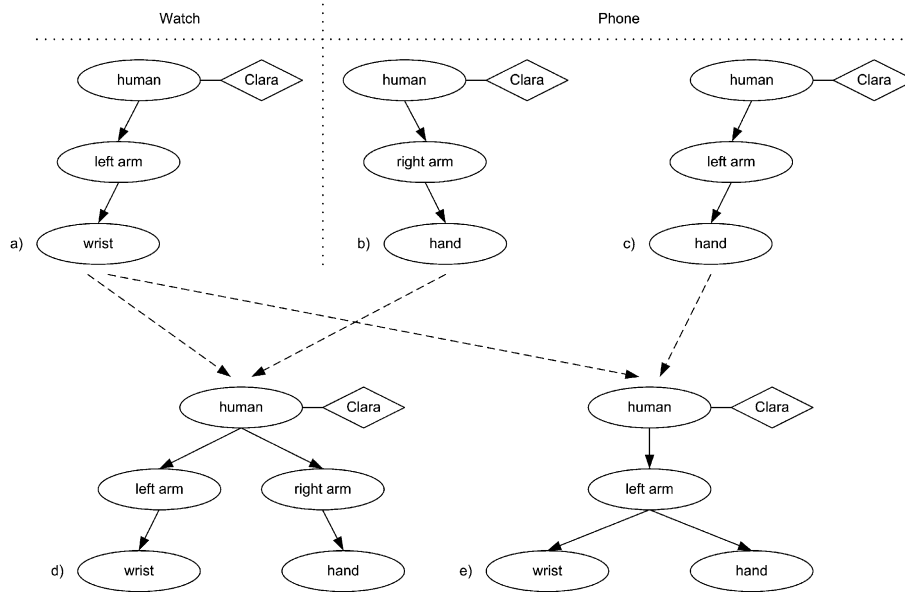


Fig. 7. Dynamic composition of gestures.

sensors and sensor modules from different producers, and to accommodate the dynamics of these relations.

5. Experimental results

An initial implementation of the framework described in Sections 3 and 4 has been achieved on a PC. This implementation was then employed to illustrate the use of two artifacts for controlling a media player by orientation sensing. To deal with the management of the framework entities, we have chosen an OSGi compliant platform that is freely available [17].

5.1. Sensors

Our initial set choice of sensors was motivated by the observations regarding orientation sensing in Sections 1 and 2. Thus, we use the InertiaCube3 sensor from Intersense, which has the following characteristics [6]:

- 3 degrees of freedom (yaw, pitch and roll) with full 360° in all three axes.
- Integration of 9 sensing elements (gyroscopes, accelerometers and magnetic field sensors) in order to provide absolute drift-free orientation data.
- Update-rate of up to 180-Hz and angular rate of up to 1200 °/s.
- Accuracy of 1° RMS at 25 °C and angular resolution of 0.01° RMS.
- RS232 serial interface.

In order to have inertial sensors embedded in artifact, it is necessary to ensure autonomous operation and wireless

communication. We use Bluetooth for wireless data transmission, achieved by the OEM Serial Port Adapter 33i from connectBlue [4]. This enables devices with RS232 to communication via a class 1 Bluetooth module, which has a range of up to 100 m. For autonomous operation, we use an accumulator with 860 mAh at a nominal voltage of 6 V, which enable at least 2 h of operation. As different working voltages are required for the sensor and the adapter, a voltage regulator has been integrated. The whole sensing hardware has a weight of about 130 g and fits in a rectangular box of dimensions 85 × 55 × 20 mm (length × width × height), which can be, for example, a cigarette box. The block diagram and the physical realization can be seen in Fig. 8.

5.2. Framework implementation

5.2.1. The sensor module

The sensor module that corresponds to the above sensor incorporates different functionalities and information that allow the sensor to be used for detection of all three types of gestures in the Gesture Library. The functional parts are detailed below.

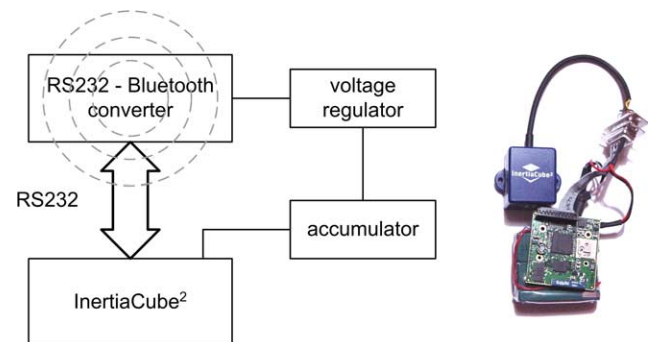


Fig. 8. Hardware setup [20].

Data acquisition and part of the pre-processing is done by the vendor software, which comes as a library that must be installed on the PC. This library provides the Euler angles or the quaternion representation of the sensor's absolute orientation. (Our preprocessing software eliminates the Euler singularity as explained in Section 2.)

Data segmentation refers to detection of the starting and ending points of a gesture. This step depends on which type of gesture is being monitored. Since this sensor was intended for usage with all three types, the segmentation method is selected at runtime, depending on the object to which the sensor is attached:

- **Hand:** As noted in [8] and the references therein, a continuous hand gesture consists of a series of qualitatively different kinematical phases such as movement to a position, hold, and transitional movement. We considered the sequence preparation-stroke-hold-retraction. The stroke is distinguished by a peaking effort, which translates into high angular velocities. The hold phase is defined by low velocity during a short time. When multiple gestures are executed immediately one after another, the retraction can be either very short or completely eliminated. In order to obtain velocities, we derivate the orientation input data. For the purpose of gesture recognition, movement of human hands can be sampled with sufficient accuracy at relatively low frequencies. In our case, a gesture is sampled at approximately 10 Hz. On the other hand, we use a higher sampling rate (50 Hz) to compute the hand (angular) velocity, in order to be able to detect a gesture from its inception, i.e. beginning of the stroke phase. If changes in the input coordinates are slow in time, and then at least one of the coordinates starts changing faster than a certain threshold, then the gesture sampling begins. If velocities become low on all three coordinates, and stay low for more than two seconds, it is considered that the gesture ended and the system is back in the preparation phase. Moreover, to allow a hand gesture to be recognized regardless of the person's heading, the gesture date is shifted by the initial yaw angle.
- **Artifact held in hand:** The segmentation is performed the same as for the hand.
- **Detached artifacts.** In this case, every movement of the artifact is a potential gesture. As opposed to the above cases, every movement of the artifact is intentional. The user can change the position of the artifact in the environment, however any significant change in the orientation data triggers the segmentation algorithm. Thus, a gesture in this case represents any variation of the orientation (above a certain threshold) between two steady states. A steady state is represented by no variation or very small variations lasting at least half a second.

The sensor module contains information about the artifact to which the sensor is attached. The model information is represented as a set of models obtained from training sample

data corresponding to each of the gestures described in Section 4. The training was done in accordance to the classification method described below.

### 5.2.2. Classification module

We considered the Support Vector Machine approach for gesture classification. In this respect, we used the library LibSVM [3], which provides functions for training, calibration, and classification.

Since at this stage the Gesture Library does not specify composite gestures, no Composition module was used.

### 5.2.3. Gesture API

This interface allows applications to register for listening to gesture events. An interested application is notified whenever a gesture is recognized. An application may specify criteria for selection of gestures in which the application is interested. For example, an application may want to be notified only about hand gestures. The Gesture API delivers a gesture in the format defined by the Gesture Library specification, together with the identification of the object that performed the gesture.

### 5.3. Application

The application considered here is a media player that accepts as input commands like: play, stop, pause, increase/decrease volume, next/previous song, next/previous playlist,

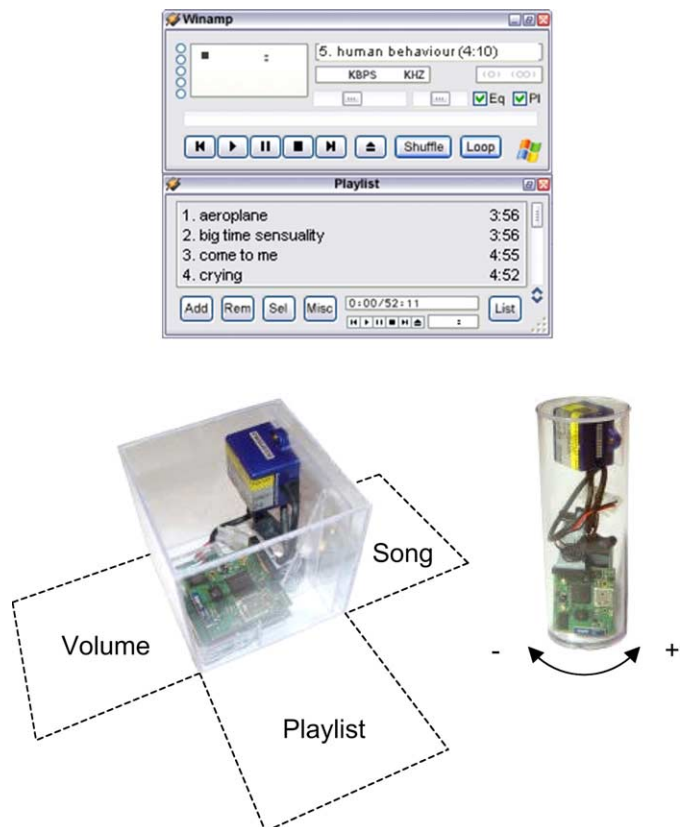


Fig. 9. Media player control.



Fig. 10. Artifacts with orientation sensors.

etc. These commands are mapped to gestures of two artifacts: a box and a cylinder, as depicted in Fig. 9.

The face of the box that is upwards indicates whether the player is stopped, paused, or it plays a song. When the ‘play’ face is up, its horizontal orientation (the value of the Yaw angle) indicates which commands are issued when rotating the cylinder. In this case, the box acts like a selector with four positions: volume, song, playlist and seek.

## 6. Conclusions and future work

Due to technological advancements like MEMS technology, in the near future orientation sensors are expected to be so small, that they can be integrated in smart objects that can be almost continuously worn (e.g. wrist watches) or manipulated (e.g. mobile phones), as well as larger objects that are occasionally moved by a human (e.g. bottles).

This plethora of orientation sensing in the environment will enable a human to use intuitive gesture-based interaction for remote control of various applications and devices. In this context, to achieve interoperability among different sensors, and among different applications, it is necessary to provide application developers and sensor manufacturers with relevant common specifications of information structures and operational requirements.

In this paper, an application-independent set of gestures was presented. Then, a general framework for gesture recognition was specified, that is independent of sensor technology and classification methods. An implementation and an application of this framework were described.

We intend to employ the above system for investigating the use of various artifacts with different shapes, like the ones shown in Fig. 10.

The work presented in this paper can be further pursued in several directions. Clearly, achieving a comprehensive library of gestures is a challenging endeavor, which involves considering also position sensing. Moreover, additional effort is needed to obtain a fully functional implementation of the gesture framework, especially for small computational platforms like mobile phones.

## References

- [1] T. Baudel, M. Beaudouin-Lafon, Charade: remote control of objects using free-hand gestures, *Communications of the ACM* 36 (7) (1993) 28–35.
- [2] A.Y. Benbasat, J.A. Paradiso, An Inertial Measurement Framework for Gesture Recognition and Applications. *Gesture Workshop, 2001 LNAI* 2298, pp. 9–20.
- [3] Chih-Chung Chang, Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [4] connectBlue Inc. Internet: <http://www.connectblue.se/>.
- [5] P. Hamette, P. Lukowicz, G. Tröster, T. Svoboda, FingerMouse: A Wearable Hand Tracking System, *UBICOMP Conference Proceeding, ETH Zürich, September, 2002*.
- [6] InterSense Inc. InertiaCube2 Product Manual. Internet: <http://www.isense.com/products/prec/ic2/index.htm>.
- [7] H. Ishii, B. Ullmer, Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, *ACM Proceedings of CHI97, March, 1997*.
- [8] S. Kettebekov, R. Sharma, Toward Natural Gesture/Speech Control of a Large Display, *The 8th IFIP Working Conference on Engineering for Human-Computer Interaction, 2001, LNCS 2254, 2001 pp. 221–234*.
- [9] M. Koelsch. Vision Based Hand Gesture Interfaces for Wearable Computing and Virtual Environments. PhD Thesis, University of California at Santa Barbara, 2004.
- [10] Chan-Su Lee, Sang-Won Ghyme, Chan-Jong Park, KwangYun Wohn, The Control of Avatar Motion Using Hand Gesture, *Virtual Reality Software and Technology, VRST 1998, November 2-5, Taipei, Taiwan, ACM, 1998*.
- [11] J.C. Lementec, P. Bajcsy, Recognition of Arm Gestures Using Multiple Orientation Sensors: Gesture Classification, *Seventh International IEEE Conference on Intelligent Transportation Systems, Washington, DC, October 3–6 2004; 965–970*.
- [12] S. Lenman, L. Bretzner, B. Thuresson. Computer Vision Based Hand Gesture Interfaces for Human-Computer Interaction. Technical Report TRITA-NA-D0209, 2002, CID-172, Royal Institute of Technology, Sweden.
- [13] J. Lester, B. Hannaford, ‘Are you with me?’-Using Accelerometers to Determine if Two Devices are Carried by the Same Person, *Proceedings of Pervasive 2004, LNCS 3001, 2004 pp. 33–50*.
- [14] J. Mäntyjärvi, J. Kela, P. Korpipää, S. Kallio, Enabling Fast and effortless Customisation in Accelerometer Based Gesture Interaction, *Third International Conference on Mobile and Ubiquitous Multimedia (MUM2004), College Park, Maryland, USA, 2004*.
- [15] M. Nielsen, M. Störring, T.B. Moeslund, E. Granum. A procedure for developing intuitive and ergonomic gesture interfaces for man-machine interaction. Technical report CVMT 03-01, 2003, Aalborg University.
- [16] C. Nölker. GREFIT: Ein System zur visuellen Erkennung von Handposturen, PhD thesis, Bielefeld, Germany, October 2000.
- [17] Open Service Container Architecture (OSCAR) <http://oscar-osgi.sourceforge.net/>.
- [18] V.I. Pavlovic, R. Sharma, T.S. Huang, Visual Interpretation of Hand Gestures for Human-Computer Interaction: a Review, *IEEE Transaction on Pattern Analysis and Machine Intelligence* 19 (7) (1997).
- [19] T. Pering. icube: A Tangible Interface for Mobile Interaction, *Second International Conference on Pervasive Computing (Pervasive 2004, Linz/Vienna, Austria, April 18–23, 2004), Workshop 2: Toolkit Support for Interaction in the Physical World*.

- [20] M. Reichör. TangibleControl: Universelle TUIs mittels Accelerometer. Master Thesis, University of Linz, Austria, October 2004.
- [21] P. Dourish. Where the action is: The foundations of embodied interaction. Cambridge, MA: MIT Press.
- [22] T.E. Starner, J. Weaver, A. Pentland, A. Wearable, A Wearable Computer Based American Sign Language Recognizer, IEEE International Symposium on Wearable Computers, Massachusetts Institute of Technology, October, 1997.
- [23] M. Urban, P. Bajcsy, R. Kooper, J.C. Lementec, Recognition of Arm Gestures Using Multiple Orientation Sensors: Repeatability Assessment, Seventh International IEEE Conference on Intelligent Transportation Systems, Washington, DC 3-6 (October 2004) 553–558.
- [24] J. Vehmas, S. Kallio, J. Kela, J. Plomp, E. Tuulari, H. Ailisto, EDEMO-Gesture-based Interaction with Future Environments. Accepted to HCI International 2003.
- [25] R. Watson. A Survey of Gesture Recognition Techniques. Technical Report TCD-CS-93 11, Department of Computer Science, Trinity College, Dublin, July 17, 1993.
- [26] D. Wilson, A. Wilson. Gesture recognition using the XWand. Technical Note: TR-04-32, 2002, Carnegie Mellon University.



**Alois Ferscha** received the Mag. degree in 1984, and a PhD in business informatics in 1990, both from the University of Vienna, Austria. From 1986 through 2000 he was with the Department of Applied Computer Science at the University of Vienna at the levels of assistant and associate professor. In 2000 he joined the University of Linz as full professor where he is now head of the department for Pervasive Computing and the speaker of the JKU Pervasive Computing Initiative. Prof. Ferscha has published on topics related to parallel and distributed computing, like e.g. Computer Aided Parallel Software Engineering, Performance Oriented Distributed/Parallel Program Development, Parallel and Distributed Discrete Event Simulation, Performance Modeling/Analysis of Parallel Systems and Parallel Visual Programming. Currently he is focussed on Pervasive Computing, Embedded Software Systems, Wireless Communication, Multiuser Cooperation, Distributed Interaction and Distributed Interactive Simulation. He has been the project leader of several national and international research projects like e.g.: Network Computing, Performance Analysis of Parallel Systems and their Workload, Parallel Simulation of Very Large Office Workflow Models, Distributed Simulation on High Performance Parallel Computer Architectures, Modelling and Analysis of Time Constrained and Hierarchical Systems, Broadband Integrated Satellite Network Traffic Evaluation and Distributed Cooperative Environments, etc. Currently he is pursuing project work related to networked embedded systems, software frameworks for context computing, coordination architectures and models, wireless and mobile ad-hoc networks and sensor/actuator networks. In his application related work he has built context based application frameworks for the JKU ‘Wireless Campus’ network, public community displays with wireless remote controls (‘WebWall’), geo-enhanced, augmented reality mobile navigation systems, RFID based realtime notification systems, wearable computing and embedded internet application frameworks (‘DigtalAura’, ‘SmartCase’, ‘DigiScope’). He has been a visiting researcher at the Dipartimento di Informatica, Universita di Torino, Italy, at the Dipartimento di Informatica, Universita di Genoa, Italy, at the Computer Science Department, University of Maryland at College Park, College Park, Maryland, USA, and at the Department of Computer and Information Sciences, University of Oregon, Eugene, Oregon, USA. He has served on the committees of several conferences like PERVASIVE, UMBICOMP, WWW, PADS, DIS-RT, SIGMETRICS, MASCOTS, TOOLS, PNPM, ICS, etc. Prof. Ferscha is member of the OCG, GI, ACM, IEEE and holds the einz-Zemanek Award for distinguished contributions in computer science.



**Clemens Holzmann** is a PhD student at the University of Linz, Austria. His research interests include pervasive and ubiquitous computing, context awareness, sensor technologies and human computer interaction. He received a Dipl.-Ing. degree in 2004 in computer science from the University of Linz. Since 2003, he has been working as research associate at the Department of Pervasive Computing and studying business informatics, both at the University of Linz.



**Stefan Resmerita** received his B.Sc. (1996) and M.Sc. (1997) degrees in Electrical Engineering from the Technical University of Iasi, Romania. He obtained a Ph.D. in Computer Science (2003) from the Technion-Israel Institute of Technology. He has been a teaching and research assistant at the Department of Automatic Control and Industrial Informatics, Technical University of Iasi (1996-1998, 2003-2004), and at the Department of Computer Science, the Technion (1998-2003). Since 2004 he is a research associate at the Institute for Pervasive Computing, University of Linz, Austria. S. Resmerita has pursued research on Control Engineering, Knowledge-based Systems, Hybrid Systems Control. His current interests include Multi-agent Systems, Air Traffic Management, Embedded Systems, Software architectures for context computing, Human-Computer Interaction. He has been a visiting researcher at NASA Ames Research Center, Moffett Field, California, USA. S. Resmerita is a member of the IEEE.



**Martin Reichoer** studied computer science at the University of Linz/Austria from 1999 to 2004. He received his M.Sc. in 2004. The title of his diploma thesis, which was written at the Institute for Pervasive Computing at the University of Linz, is ‘TangibleControl - Universelle TUIs mittels Accelerometer’. Currently he works at technosert electronic in Upper Austria. His main areas of activity are software quality, certification of safety critical applications and cost estimation of hard- and software projects.