

Gestural interaction in the pervasive computing landscape

A. Ferscha, S. Resmerita IEEE

Pervasive computing has postulated to invisibly integrate technology into everyday objects in such a way, that these objects turn into *smart things*. Not only a single object of this kind is supposed to represent the interface among the “physical world” of atoms and the “digital world” of bits, but a whole landscapes of them. The interaction with such technology rich artefacts is supposed to be guided by their affordance, i.e. the ability of an artefact to express the modality of its appropriate use. We study human gesticulation and the manipulation of graspable and movable everyday artefacts as a potentially effective means for the interaction with smart things. In this work we consider *gestures* in the general sense of a movement or a state (posture) of the human body, as well as a movement or state of any physical object resulting from human manipulation. Intuitive “everyday”-gestures have been collected in a series of user tests, yielding a catalogue of generic body and artefact gesture dynamics. Atomic gestures are described by trajectories of orientation data, while composite gestures are defined by a compositional gesture grammar. The respective mechanisms for the recognition of such gestures have been implemented in a general software framework supported by an accelerometer-based sensing system. Showcases involving multiple gesture sensors demonstrate the viability of implicit embedded interaction for real life scenarios.

Keywords: pervasive computing; embodied interaction; tangible interfaces; gesture recognition; orientation tracking

Interaktion durch Gesten im Pervasive Computing.

Pervasive Computing hat sich zum Ziel gesetzt, Technologie quasi „unsichtbar“ in Gegenstände des alltäglichen Lebens zu integrieren. Dabei geht es nicht nur darum, dass einzelne Gegenstände eine Schnittstelle zwischen der physischen Welt der Atome und der digitalen Welt der Bits bilden, vielmehr zählt die Gesamtheit dieser „Smart Things“ und wie sie miteinander interagieren.

Die Autoren untersuchen die menschliche Gestik sowie die Manipulation von beweglichen Gegenständen als mögliche effektive Maßnahme zur Interaktion mit den „Smart Things“. In diesem Beitrag erstreckt sich die Bezeichnung von „Gesten“ sowohl auf Gesten im allgemeinen Sinn als Bewegung oder Zustand (Haltung) des menschlichen Körpers als auch auf die Bewegung oder den Zustand eines physischen Gegenstands aufgrund von menschlicher Manipulation. In einer Reihe von Tests wurden Gesten des alltäglichen Lebens gesammelt, die einen Katalog von generischen Körperbewegungen bzw. Bewegungen von Objekten ergeben haben. Einzelgesten werden als Sequenzen von Orientierungsdaten dargestellt, während zusammengesetzte Gesten durch ein eigenes Regelwerk beschrieben werden. Die entsprechenden Mechanismen wurden in ein Softwaresystem integriert. Sensormodelle zur Erfassung komplexer Gestik weisen bereits auf die Machbarkeit von Systemen zur implizierten eingebetteten Interaktion in Anwendungen hin.

Schlüsselwörter: Pervasive Computing; gestenbasierte Interaktion; gegenständliche Schnittstelle; Gestenerkennung

Eingegangen am 6. November 2006, angenommen am 23. November 2006
© Springer-Verlag 2007

1. Introduction

Computing devices are already now pervading into everyday objects, in such a way that users do not notice them anymore as separate entities. Appliances, tools, clothing, accessories, furniture, rooms, machinery, cars, buildings, roads, cities, even whole agricultural landscapes increasingly embody miniaturized and wireless – thus invisible – information and communication systems, establishing information technology rich socio-economic systems with the potential of radically changing the style of how we perceive, create, think, interact, behave and socialize as human beings, but also how we learn, work, cultivate, live, cure, age as individuals or in societal settings.

Prospective advances in microprocessor-, communication- and sensor/actuator-technologies envision a whole new era of computing systems, seamlessly and invisibly woven into the “fabric of everyday life”, and hence referred to as pervasive computing. Their services will be tailored to the person and the context of their use. After the era of keyboard and screen interaction, a computer will be understood as secondary artefact, embedded and operating in the background, with its complete physical environment acting as inter-

face (primary artefact). Pervasive computing aims at interaction with digital information by manipulating physical real world artefacts as “graspable interfaces”, by simultaneously involving all human senses, and by considering interaction related to the semantics of the situation in which it occurs.

The future pervasive computing senses and controls the physical world via many sensors and actuators, respectively. Applications and services will therefore have to be greatly based on the notions of context and knowledge, they will have to cope with highly dynamic environments and changing resources, and will need to evolve towards a more implicit and proactive interaction with humans. Communication must go beyond sending information from one fixed point to another, considering situations where devices cooperate and adapt spontaneously and autonomously in the absence of any centralized

Ferscha, Alois, Univ.-Prof. Dr., Institute of Pervasive Computing, Johannes Kepler University of Linz, Altenberger Straße 69, 4040 Linz, Austria (E-mail: ferscha@soft.uni-linz.ac.at); Resmerita, Stefan, Univ.-Ass Dipl.-Ing. Dr., University of Salzburg, Department of Computer Sciences, Jakob Haringer-Straße 2, 5020 Salzburg, Austria

control. A vast manifold of small, embedded and mobile artefacts characterize the scenarios envisaged by pervasive computing. The challenges are related to (i) their ubiquity, (ii) their self-organisation and interoperation, (iii) their ability of perceiving and interpreting their situation and consequently (iv) adapt the services they offer, the different modes of user interaction with those services.

1.1 Gestural interaction

Human gesticulation as a modality of human-machine interaction has been widely studied in the field of Human-Computer Interaction. With the upcoming pervasive and Ubiquitous computing research field, the explicit interaction with computers with mouse, keyboard and screen in the WIMP metaphor has given way to a more implicit interaction involving all human senses. As an important part of this tendency, gestures and movements of the human body represent a natural and intuitive way to interact with physical objects in the environment. Thus, manipulation of objects can be regarded as a means of intuitive interaction with the digital world. This paradigm underlies the research on Tangible User Interfaces (TUIs) (Ullmer, Ishii, 2000). Embodied interaction (Dourish, 2001; Fishkin, Moran, Harrison, 1998) aims at facilitating remote control applications by providing natural and intuitive means of interaction, which are often more efficient and powerful compared to traditional interaction methods. TUIs couple physical representations (e.g. spatially manipulable physical artefacts) with digital representation (e.g. graphics and sounds), making bits directly manipulable and perceptible by people (Fitzmaurice, Ishii, Buxton, 1995; Holmquist, Redström, Ljungstrand, 1999). In general, tangible interfaces are related to the use of physical artefacts as representations and controls for digital information (Ullmer, Ishii, 2000).

In this work, a *gesture* is considered in the general sense of a movement or a state (posture) of the human body, as well as a movement or state of any physical object (artefact) resulted from human manipulation. Clearly, the expressive power of gestures in this general sense makes the gesture itself a significant candidate for intuitive interaction. To encourage application developers to consider gestural input for their applications, one needs common specifications of gestures and independent implementations of gesture providers. To this end, we propose an application independent Gesture Library (GLib), and show how GLib can be implemented and employed for embodied interaction. The GLib is an open set of gestures, which is also quite comprehensive for the purpose of implementing applications of embodied interaction. The GLib contains atomic and composite gestures, where the former are modelled from sensor data, and the latter are defined by a gesture grammar.

We also present several applications of the GLib that demonstrate the versatility of the library as a tool for achieving embodied interaction. These applications involve hand gestures, gestures of handy artefacts that are often manipulated, and gestures of artefacts that are only occasionally manipulated.

Several gestural command sets and guidelines for obtaining such gestures have been proposed in the literature (Lenman, Bretzner, Thuresson, 2002; Nielson et al., 2003), while other research work has established the related software frameworks for implementing embodied user interfaces (Koleva et al., 2003; Greenberg, Fitchett, 2001). In these papers, each of the proposed gesture set is application-dependent. It is argued in (Nielson et al., 2003) that gesture sets should be application-dependent. In contrast, we propose an application-independent collection of gestures and an implementation of a gesture provider that is able to work with multiple applications (at the same time). In (Westeyn et al., 2003), a rule-based grammar for describing sequences of gestures is presented. In our case, we also consider parallel and overlapping dynamic gestures. While gesture sensing and classification is subject to a large body of literature, the usage of inertial sensing has only recently received attention, with the

main focus on linear acceleration (Benbasat, Paradiso, 2001; Amft, Junker, Trster, 2005). Inertial orientation sensors have been considered for gesture detection in (Lementec, Bajcsy, 2004), where a recognition algorithm for a specific set of gestures is presented. In contrast, we evaluate here standard algorithms which are independent of the types of gestures that should be recognized.

2. Gestures

While many authors consider that gestures should be application-dependent, to our knowledge there are few that propose application-independent gestural command sets (Benbasat, Paradiso, 2001). 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. We consider that the mapping from gestures to application commands is application-specific. In this section, we introduce a comprehensive, yet open, Gesture Library (GLib). An abstract gesture in GLib is formally represented by the tuple

$$G = ((Object), (Name), [Parameters]),$$

where *(Object)* is a generic denomination of the object that performs the gesture (e.g., hand, artefact type), *(Name)* is a gesture identifier that is unique within the scope of the same object type, and *[Parameters]* is a (possibly empty) list of gesture attributes. For example, a parameter for a rotation gesture is the angle of rotation.

We distinguish between atomic and composite gestures. A gesture is called *atomic* if the information model of the gesture is strictly related to the sensor data, which completely characterize the gesture. This model is implementation-dependent: it may consist of representations of labelled feature sets (e.g., Support Vectors (Burgess, 1998)), generative assumptions (e.g., Markov transition probabilities), or algorithmic, rule-based, or logical descriptions of the gesture in terms of sensor data. A *composite gesture* is modelled only in terms of other gestures in GLib, as an expression containing gestures related by connectors. The expression must be well-formed, according to a gesture grammar.

2.1 Atomic gestures

The gestures presented in this section have been obtained empirically, by several experiments carried out in a laboratory setting, where various people were asked to perform simple hand gestures and object manipulations. Wireless inertial orientation sensors were used to track gesture data. A snapshot from these experiments is shown in Fig. 1(a), with the corresponding virtual 3D model depicted in Fig. 1(b). A detailed presentation of these experiments and their results is being reported elsewhere.

For simplicity, the following atomic gestures are described in an informal way and in some cases pictures are used for illustration. While we acknowledge the importance of having a formal description model for gestures, we do not address this issue here. Most of the gestures have a common parameter: the duration of the gesture in milliseconds. Unless otherwise specified, each of the following hand gestures starts from an initial position where the upper arm lies alongside the body.

1. $G_1 = (RightHand, Take, duration)$. In the initial position, the forearm is horizontal and pointing to the front, the hand is open, with the palm facing up. When performing the gesture, the upper arm stays always relaxed alongside the body. 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°. The palm should be always in continuation of the forearm. Thus,

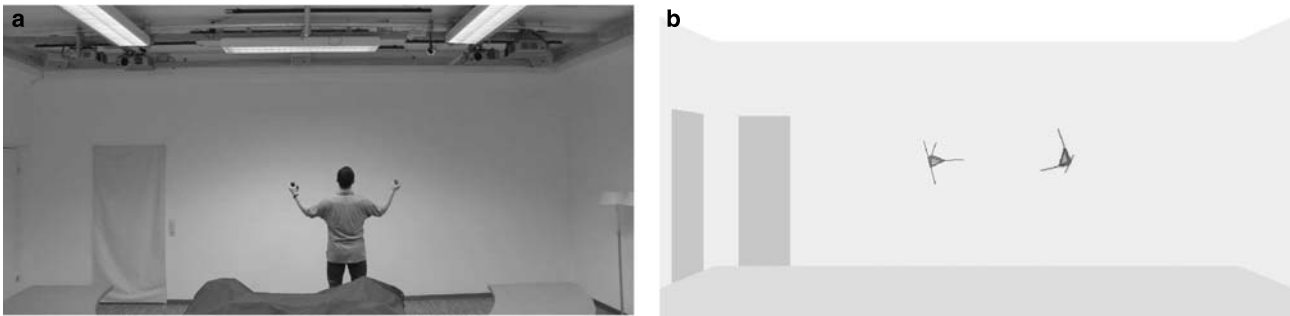


Fig. 1. Gesture evaluation. (a) Gesture experiments, (b) Virtual model of the lab and sensors

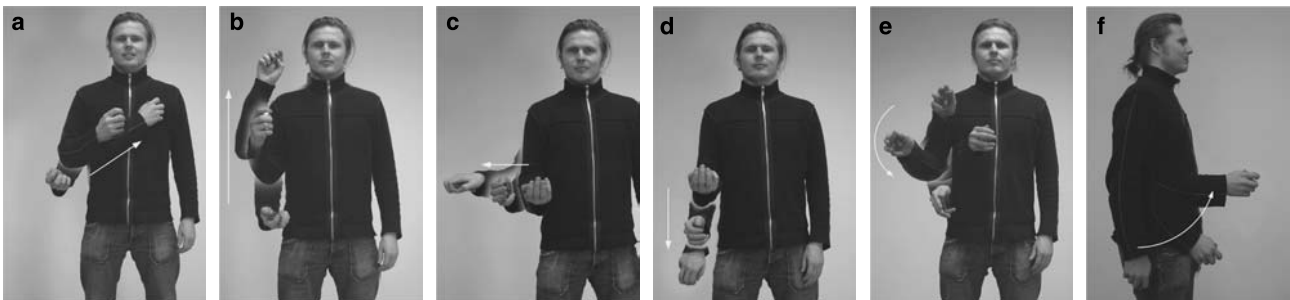


Fig. 2. Hand gestures. (a) Take, (b) Commit, (c) Throw, (d) Leave, (e) Circle, (f) Balance

the palm will actually touch the body at the end of the gesture. This gesture is presented in Fig. 2(a).

- $G_2 = (\text{RightHand}, \text{Commit}, \text{duration})$. The initial position is the same as the above. 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° , then stop. When the forearm is about vertical, the palm should face to the front. The palm should be always in continuation of the forearm. This gesture is presented in Fig. 2(b).
- $G_3 = (\text{RightHand}, \text{Throw}, \text{duration})$. Rotate the forearm horizontally to the right as much as conveniently possible. At the same time, rotate the wrist about the forearm counterclockwise with about 180° . When the forearm is maximally to the right, the palm should face down. 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. 2(c).
- $G_4 = (\text{RightHand}, \text{Leave}, \text{duration})$. Move the forearm down with about 90° . At the same time, rotate the wrist about the forearm counterclockwise with about 90° . 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. 2(d).
- $G_5 = (\text{RightHand}, \text{CircleRight}, \text{duration}, \text{direction})$. The initial position is the same as for G_1 , except that now the palm is oriented to the left. Rotate the forearm about the elbow such that the tip of the hand describes one complete circle in a vertical plan, then stop. The initial movement of the hand must be towards the right. Fig. 2(e) illustrates this gesture. The parameter *direction* can be either *clockwise* or *counterclockwise*.
- $G_6 = (\text{RightArm}, \text{BalanceDownBackForth}, \text{duration})$. Initially, the hand is extended to the back as shown in Fig. 2(f). Bring the hand to the front by rotating the upper arm, as shown in the figure.
- $G_7 = (\text{LeftArm}, \text{BalanceDownBackForth}, \text{duration})$. This is defined similarly with the previous gesture, but for the left hand.
- $G_8 = (\text{RightHand}, \text{Pick}, \text{duration})$. Initially, the forearm and the hand are horizontal, with the palm oriented down. Rotate the wrist about the forearm with about 180° clockwise, i.e., bring the palm upwards, then stop.
- $G_9 = (\text{RightArm}, \text{BalanceUpLeftRight}, \text{duration})$. In the initial position, the forearm is extended to the front and up, such that the angle between the upper arm and forearm is about 60° . Moreover, the elbow is rotated about the upper arm with approximately 45° clockwise, such that the forearm is in the exterior of the body. While the exact orientation of the hand is not relevant for this gesture, let us consider that the hand is in continuation of the forearm, with the palm oriented to the left. Rotate the elbow about the upper arm with 90° counterclockwise, so as to bring the forearm pointing to the interior of the body, then stop. This gesture can be seen in Fig. 4(a) for the right hand.
- $G_{10} = (\text{LeftArm}, \text{BalanceUpRightLeft}, \text{duration})$. This gesture is defined similarly to G_9 , but for the left arm and it can be seen in Fig. 4(a) for the left hand.
- $G_{11} = (\text{RightArm}, \text{RotateHorizontal}, \text{duration}, \text{direction}, \text{angle})$. Initially, the forearm is in horizontal position. Rotate the elbow about the upper arm and keep the forearm horizontal. The parameter *direction* can be *clockwise* or *counterclockwise* and the parameter *angle* specifies the angle (in degrees) of rotation.
- $G_{12} = (\text{RightHand}, \text{Shake}, \text{duration})$. Initially, the hand lies alongside the body. Raise the forearm to the front and shake it twice, as when shaking hands with somebody.
- $G_{13} = (\text{RightHand}, \text{Hike}, \text{duration})$. Initially, the hand lies alongside the body. Raise the forearm in the lateral plane and shake it twice, then put it back to the initial position.
- $G_{14} = (\text{RightHand}, \text{Knock}, \text{duration})$. Initially, the forearm is raised in the lateral plane such that the palm faces the front. The forearm is rotated ahead (i.e., to the front) for about 45° , then is retreated. The movement is similar to knocking on a door.
- $G_{15} = (\text{RightHand}, \text{Salute}, \text{duration})$. Initially, the hand lies alongside the body. Raise the hand until the tip of the thumb touches the right side of the forehead, then put the hand back into the initial position.

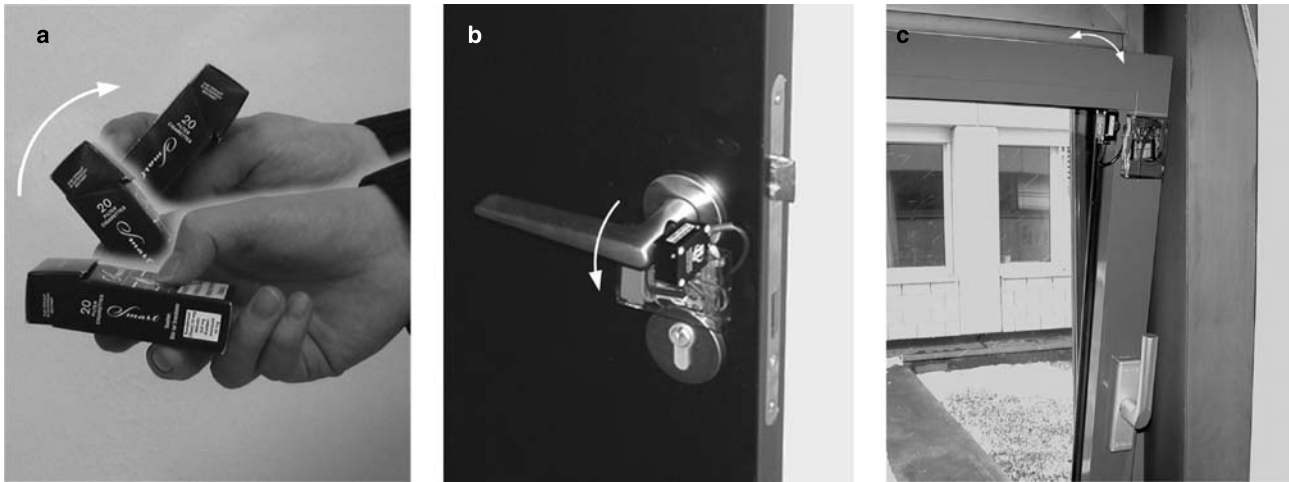


Fig. 3. Artefact gestures. (a) Box shake, (b) Handle press, (c) Window rotate

16. $G_{16} = (\text{SixFaceBox}, \text{Shake}, \text{duration}, \text{face_up})$. Holding the box in hand, shake it twice in the vertical plane. The gesture is illustrated in Fig. 3(a). The parameter *face_up* is a number from 1 to 6, specifying which of the box's faces is oriented upwards.
17. $G_{17} = (\text{SixFaceBox}, \text{Flip}, \text{duration}, \text{initial_face}, \text{final_face})$. Flip the box such that before the flip the face number *initial_face* is oriented upwards and after the flip the face number *final_face* is oriented upwards. The values of these parameters are numbers between 1 and 6 such that $\text{initial_face} \neq \text{final_face}$.
18. $G_{18} = (\text{SixFaceBox}, \text{Rotate}, \text{duration}, \text{direction}, \text{angle}, \text{face_up})$. A box oriented with face number *face_up* upwards is rotated with *angle* degrees in the direction specified by *direction* (which can be either *clockwise* or *counterclockwise*), about a vertical axis that is orthogonal to *face_up*.
19. $G_{19} = (\text{Cylinder}, \text{Rotate}, \text{duration}, \text{direction}, \text{angle}, \text{face_up})$. A cylinder object is defined by two planar surfaces (denoted by 1 and 2) and a circular surface. A cylinder oriented with face number *face_up* upwards is rotated with *angle* degrees in the direction specified by *direction* (which can be either *clockwise* or *counterclockwise*), about an axis that is orthogonal to the planar surfaces.
20. $G_{20} = (\text{Chair}, \text{Rotate}, \text{duration}, \text{direction}, \text{angle})$. A chair is rotated about a vertical axis with *angle* degrees in the direction specified by *direction*, which can be either *clockwise* or *counterclockwise*.
21. $G_{21} = (\text{Window}, \text{Rotate}, \text{duration}, \text{direction}, \text{angle})$. A window is rotated with *angle* degrees, in the direction specified by *direction*, which can be *opening* or *closing*. This is shown in Fig. 3(c).
22. $G_{22} = (\text{Window}, \text{Closed})$. A window is closed. As opposed to all the above gestures, this is a static gesture. It describes a specific state of an object of type *Window*.
23. $G_{23} = (\text{Window}, \text{Open})$. A window is open.
24. $G_{24} = (\text{DoorHandle}, \text{Press})$. A door handle is pressed, as shown in Fig. 3(b).
25. $G_{25} = (\text{DoorHandle}, \text{Release})$. A door handle is released. This is the opposite of the previous gesture.
26. $G_{26} = (\text{Door}, \text{Rotate}, \text{duration}, \text{direction}, \text{angle})$. This is defined similarly to G_{21} .
27. $G_{27} = (\text{Door}, \text{Closed})$. A door is closed.
28. $G_{28} = (\text{Door}, \text{Open})$. A door is open.

To refer to a gesture with a fixed parameter, the value of the parameter is mentioned in the index of the gesture. For example, Fig. 2(e) shows the gesture $G_{5(\text{direction}=\text{clockwise})}$.

For each of the atomic gestures G described above there exists a reverse atomic gesture, denoted by \overline{G} , defined as follows.

Let $x, y \in \{\text{Clockwise}, \text{CounterClockwise}\}$ such that $x \neq y$.

- ▶ For each of the hand gestures G_1 – G_{10} , the reverse gesture is obtained by exchanging the initial and final hand and arm postures and executing the reverse movement.
- ▶ $\overline{G_{5(\text{direction}=x)}} = G_{5(\text{direction}=y)}$.
- ▶ $\overline{G_{i(\text{direction}=x, \text{angle}=a)}} = G_{i(\text{direction}=y, \text{angle}=a)}$, where $i \in \{11, 20\}$ and $a \in [0, 360)$.
- ▶ $\overline{G_{i(\text{direction}=z, \text{angle}=a)}} = G_{i(\text{direction}=u, \text{angle}=a)}$, where $i \in \{26, 21\}$, $a \in [0, 360)$, and $z, u \in \{\text{Opening}, \text{Closing}\}$ such that $z \neq u$.
- ▶ $\overline{G_{17(\text{initial_face}=m, \text{final_face}=n)}} = G_{17(\text{initial_face}=n, \text{final_face}=m)}$, where $m, n \in \{1, \dots, 6\}$, $m \neq n$.
- ▶ $\overline{G_{i(\text{direction}=x, \text{angle}=a, \text{face_up}=k_i)}} = G_{i(\text{direction}=y, \text{angle}=a, \text{face_up}=k_i)}$, where $a \in [0, 360)$, $i \in \{18, 19\}$, $k_{18} \in \{1, \dots, 6\}$, and $k_{19} \in \{1, 2\}$.
- ▶ $\overline{G_{24}} = G_{25}$, and $\overline{G_{25}} = G_{24}$.
- ▶ $\overline{G_{27}} = G_{28}$, $\overline{G_{28}} = G_{27}$, $\overline{G_{22}} = G_{23}$, and $\overline{G_{23}} = G_{22}$.

Notice that for an atomic gesture G it holds that $\overline{\overline{G}} = G$.

2.2 Composite gestures and gesture grammar

We present in this section definitions of composite gestures based on combining atomic gestures via composition operators, which are defined in terms of timing of the involved gestures. To describe the relations, let us denote the times (expressed in milliseconds) when gesture G starts and ends by $t_s(G)$ and $t_e(G)$, respectively. The operators are defined as follows.

- ▶ **Sequence:** $G_1 \rightarrow G_2$ if $t_e(G_1) \leq t_s(G_2)$ and there is no gesture G such that $t_e(G_1) \leq t_s(G)$ and $t_e(G) \leq t_s(G_2)$.
- ▶ **Timed sequence:** $G_1 \xrightarrow{\delta} G_2$ if $0 < t_s(G_2) - t_e(G_1) \leq \delta$. That is, G_2 follows G_1 after at most δ milliseconds of time. Whenever this operator is used, δ must be specified as a parameter of the corresponding gesture.
- ▶ **Repeat:** $*G = G \rightarrow *G$.
- ▶ **Timed repeat:** $(*\delta)G = G \xrightarrow{\delta} (*\delta)G$.
- ▶ **Parallel:** $G_1 \parallel G_2$ if $|t_s(G_1) - t_s(G_2)| \leq \varepsilon$ and $|t_e(G_1) - t_e(G_2)| \leq \varepsilon$. The parameter ε must be fixed in the implementation of the GLib (as opposed to δ , which is specified as a gesture parameter).
- ▶ **While:** $G_1 \prec G_2$ if $[t_s(G_1), t_e(G_1)] \subset [t_s(G_2) + \varepsilon, t_e(G_2)]$, or $[t_s(G_1), t_e(G_1)] \subset [t_s(G_2), t_e(G_2) - \varepsilon]$.
- ▶ **Overlap:** $G_1 \Delta G_2$ if $t_s(G_2) \in [t_s(G_1), t_e(G_1)]$ and $t_e(G_2) > t_e(G_1) + \varepsilon$, or if $t_s(G_2) > (t_s(G_1) + \varepsilon, t_e(G_1))$ and $t_e(G_2) > t_e(G_1)$.

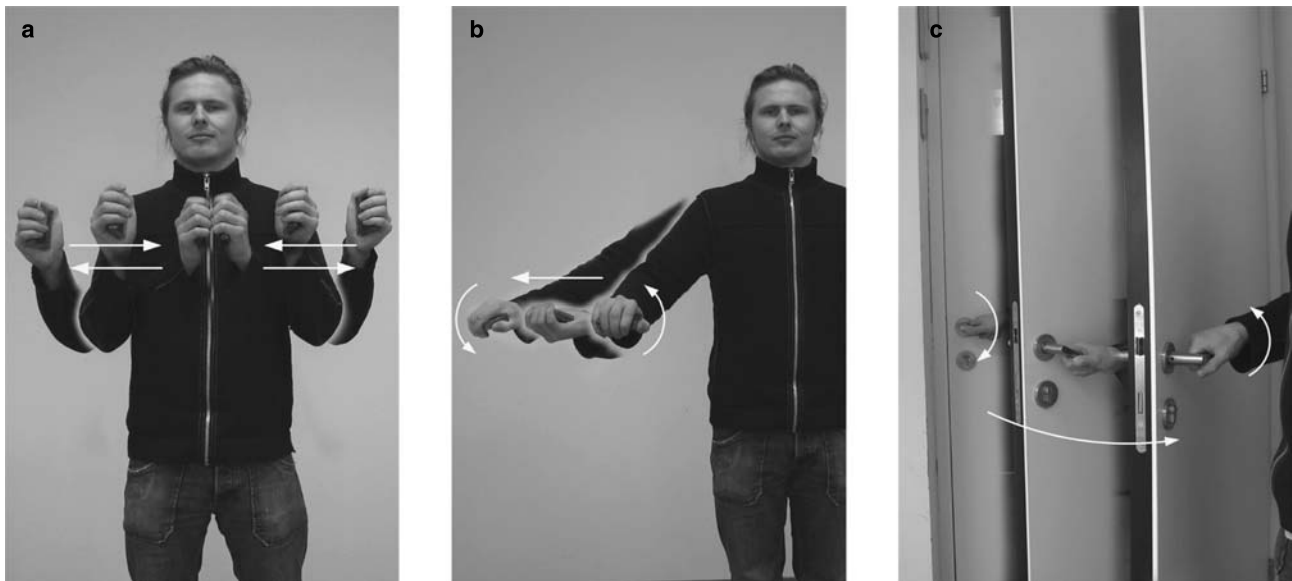


Fig. 4. Composite gestures. (a) Applause, (b) Move, (c) Door opened

- ▶ **Reverse:** The reverse of G is denoted by \overline{G} and is defined as follows:
 - ▶ If G is an atomic gesture, then \overline{G} is defined as shown in Sect. 2.1.
 - ▶ If $G = G_1 \rightarrow G_2$, then $\overline{G} = \overline{G_2} \rightarrow \overline{G_1}$.
 - ▶ If $G = G_1 \xrightarrow{\delta} G_2$, then $\overline{G} = \overline{G_2} \xrightarrow{\delta} \overline{G_1}$.
 - ▶ If $H = *G$, then $\overline{H} = *\overline{G}$.
 - ▶ If $H = (*\delta)G$, then $\overline{H} = (*\delta)\overline{G}$.
 - ▶ If $G = G_1 \parallel G_2$, then $\overline{G} = \overline{G_1} \parallel \overline{G_2}$.
 - ▶ If $G = G_1 \prec G_2$, then $\overline{G} = \overline{G_1} \prec \overline{G_2}$.
 - ▶ If $G = G_1 \Delta G_2$, then $\overline{G} = \overline{G_2} \Delta \overline{G_1}$.

A composite gesture must specify the relation between the object to which the gesture is associated (the “composite” object) and each of the objects that perform the atomic gestures which form the composed gesture (the “atomic” objects). The ensuing composite gestures require a containment relation: the composite object must physically contain each of the atomic objects. In particular, the composite object can be the same as each of the atomic objects.

29. $G_{29} = (Human, Applause)$. $G_{29} = (*500)((G_9 \xrightarrow{500} \overline{G_9}) \parallel (G_{10} \xrightarrow{500} \overline{G_{10}}))$. This gesture is illustrated in Fig. 4(a).
30. $G_{30} = (Human, Walk)$. $G_{30} = *((G_6 \parallel \overline{G_7}) \xrightarrow{500} (\overline{G_6} \parallel G_7))$.
31. $G_{31} = (RightHand, MoveRight)$. $G_{31} = G_8 \rightarrow (* (G_{11}(\text{direction} = \text{Clockwise}, \text{angle} = 1))) \rightarrow \overline{G_8}$. This gesture is illustrated in Fig. 4(b).
32. $G_{32} = (RightHand, MoveLeft)$. $G_{32} = \overline{G_{31}}$.
33. $G_{33} = (RightHand, RollOn)$. $G_{33} = (*300)G_{5(\text{direction} = \text{Clockwise})}$.
34. $G_{34} = (RightHand, RollBack)$. $G_{34} = \overline{G_{33}}$.
35. $G_{35} = (Door, Opening)$. $G_{35} = (G_{27} \prec G_{24}) \rightarrow (* (G_{26}(\text{direction} = \text{Opening}, \text{angle} = 1)))$.
36. $G_{36} = (Door, JustOpened)$. $G_{36} = G_{35} \rightarrow \overline{G_{24}}$. This gesture is illustrated in Fig. 4(c).
37. $G_{37} = (Door, Closing)$. $G_{37} = (G_{28} \prec G_{24}) \rightarrow (* (G_{26}(\text{direction} = \text{Closing}, \text{angle} = 1)))$.
38. $G_{38} = (Door, JustClosed)$. $G_{38} = G_{37} \rightarrow \overline{G_{24}}$.

The time parameters specified in these definitions have been experimentally determined by observing the corresponding gestures as performed by various persons (see Sect. 3.1).

3. GLib implementation and applications

An implementation of the GLib is a software module that is able to recognize and to provide a specified set of instances of the abstract

gestures defined in the library. An instance of an abstract gesture is represented by specified values of the parameters of the gesture.

We have achieved a flexible implementation of the GLib, called *Gesture provider*, with the structure shown in Fig. 5. An application must register to be notified about occurrences of instances of gestures in GLib. When registering, the application can specify parameter values for gestures in order to narrow down the set of desired gestures. The GLib was implemented in Java, using OSCAR, an OSGi compliant framework that allowed the gesture provider to be used by an arbitrary, dynamically changing number of applications. We modelled all the gestures in GLib in terms of orientation sensor data. The models for box, cylinder, door, window are simple state machines defined over suitable discretizations of the orientation space. The hand gesture modelling and classification are discussed in Sect. 3.1. The composition module implements a simple parser for the grammar, where each composite gesture is represented as a tree of operators. A history of gestures is maintained here, and every time a new atomic gesture occurs all the operator trees are checked with the new gesture and the gestures in the history.

Each of the classification modules and the composition module were implemented as OSGi bundles, providing gestures as services. The applications were also developed as bundles using the services provided by the GLib implementation. To this end, a Gesture API was developed based on the definitions in GLib. Before presenting the implemented applications, we take a closer look at our implementation of the hand gesture classification.

3.1 Classification of atomic gestures based on inertial orientation sensing

Due to technological advancements like MEMS technology, high performance orientation sensors are getting smaller and smaller, which makes it possible to integrate them in smart artefacts scattered around the human user. This plethora of orientation sensing in the environment will enable a human to use such small artefacts as enablers for gestural interaction, by grasping an artefact and then making hand gestures (with the artefact in hand). More generally, orientation sensors can be attached to various parts of the human body, and then be used for activity recognition.

We are interested in evaluating how hand gestures in our Gesture Library can be recognized using only orientation sensing from a sensor embedded into a smart artefact which can be grasped by

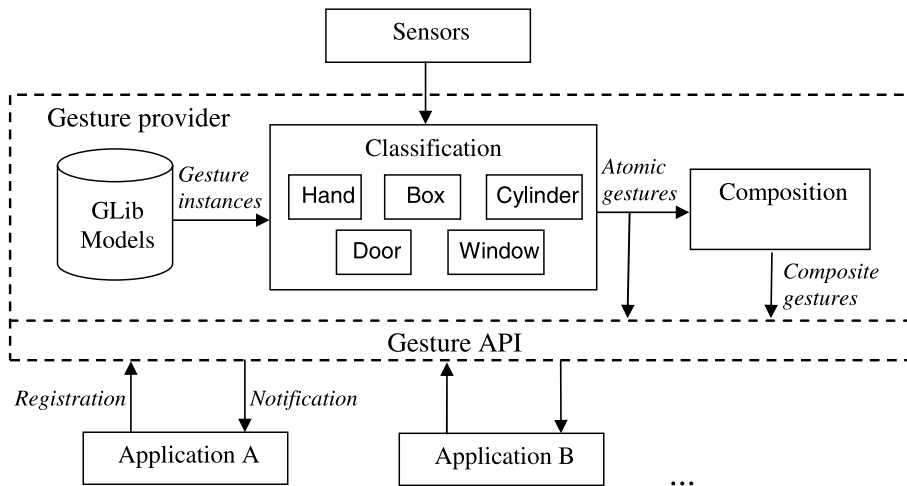


Fig. 5. The gesture provider and its usage

the human hand. To our knowledge, this has not been studied before. We present in this section a performance evaluation of two types of classification methods: K-nearest neighbors (Knn) and Support Vector Machines (SVM) (Burges, 1998) on two orientation-specific sets of features: gesture coordinates in the Euler space and in the Euclidean space, respectively.

3.1.1 The Sensor

We used in our experiments the wireless InertiaCube3 sensor from Intersense Inc, which provides Euler angles (yaw, pitch and roll) as a full representation of the sensor's orientation in 3D with respect to a fixed (Earth-related) global referential frame. The sensor is embedded into a cube that can be easily grasped by hand. A cube gesture is represented as a temporal sequence of points in the Euler space.

A particular aspect of the Euler representation of orientation is the Euler singularity, which is a 180° jump of yaw and roll values when the pitch value reached ± 90 . The reader is referred to (Lementec, Bajcsy, 2004) for a more detailed description of gesture modelling using Euler angles. As noted in (Lementec, Bajcsy, 2004), it might be useful to eliminate this strong nonlinearity from the input data for the purpose of gesture classification. Our observation is that this depends on the classification algorithm. As we shall see in the sequel, the jumps in the Euler angles can be good gesture discriminators when combined with certain classification methods.

We also use a second representation of gestures, in the form of a temporal sequence of the global coordinates (i.e., the coordinates in the fixed frame) of a specified point that is attached to the sensor's frame (a point that rotates together with the sensor). For simplicity, we consider that the sensor frame has the same origin as the fixed frame and we take the point [1, 1, 1] in the sensor's frame. In this case, the transformation from Euler angles yaw(Y), pitch(P) and roll(R) to global Euclidean coordinates is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c_p \cdot (c_y - s_y) + s_p \\ c_r \cdot (s_y + c_y) + s_r s_p \cdot (c_y - s_y) - s_r c_p \\ s_r \cdot (s_y + c_y) + c_r s_p \cdot (s_y - c_y) + c_r c_p \end{bmatrix},$$

where $c_p = \cos(P)$, $s_p = \sin(P)$, and c_y, s_y, c_r, s_r are defined similarly with respect to Y and R, respectively. Thus, a gesture is represented in Euclidean coordinates as a (timed) trajectory on the unit sphere.

3.1.2 Gesture detection

We consider hand gestures as hand movements made at reasonable high paces (Benbasat, Paradiso, 2001). Gesture segmentation is per-

formed on three data sequences, obtained by differentiating the Euclidean representation along the three axes, thus obtaining velocity projections, and applying a moving average filter to each velocity. If the value of the filtered velocity on at least one of the axes exceeds a certain threshold value thr_{start} , then a gesture beginning is detected.

Thereafter, when all three filtered velocities are under another threshold value thr_{end} for a certain window of time dt_{end} , then the end of the gesture is detected. The threshold values, the filter parameters, and the size of the moving window were determined experimentally. They are as follows: $thr_{start} = 0.8$, $thr_{end} = 0.5$, and $dt_{end} = 0.5$ s. The gesture segmentation procedure is illustrated in Fig. 6 for the Take gesture.

3.1.3 Features

The two sets of features considered for classification are the Euler and Euclidean representations of gestures, respectively. As noted earlier, our choice of the Euler representation is motivated by the fact that the Euler singularity in the data may act as a good discriminator. On the other hand, the Euclidean representation is a smooth trajectory on the unit sphere, which poses regularity properties that may be exploited in classification. There are two main disadvantages of taking the segmented sensor data as the feature vector: the feature vector may have high dimension, and it has a variable dimension. The former issue was addressed by considering classification algorithms with relatively low computational complexity. To deal with the latter, we considered the following options:

- ▶ Projecting all the feature vectors to a space of common dimension, which is determined in the training phase. We considered the trivial transformation obtained by linear interpolation, mainly because it was not expected to introduce significant errors at least for the case of the Euclidean representation.
- ▶ Computing certain distances as measures of similarity between vectors of different dimensions. Two such distances are considered: the Dynamic Time Warping (DTW) distance (Shimodaira et al., 2001), and an "Euclidean projected" distance (EP). Dynamic time warping finds the optimal alignment between two time series by stretching or shrinking them along the time axis. The Euclidean projected distance between two vectors is obtained by transforming the higher dimension vector to a vector of the same dimension as the lower dimension vector, then taking the Euclidean distance between the projected vector and the lower dimension vector. To formally define the EP distance between feature vectors v_i and v_j , let us first denote the dimension of v_i by $|v_i|$. We use $v_{i \rightarrow j}$ to

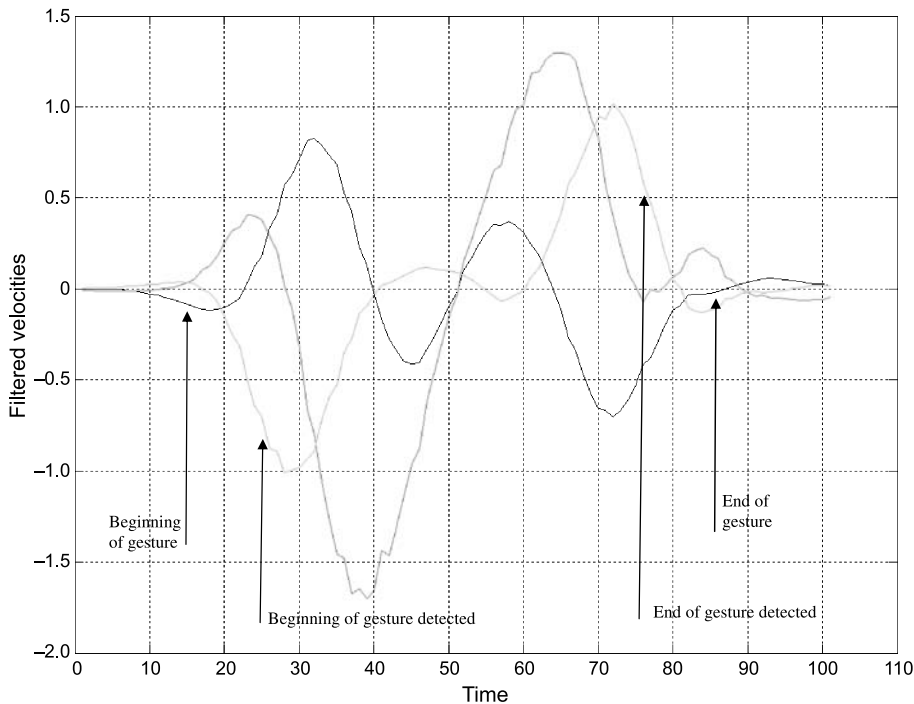


Fig. 6. Gesture segmentation

represent the vector obtained by interpolating v_i at $|v_j|$ equidistant points in time such that $v_{i \rightarrow j}(1) = v_i(1)$ and $v_{i \rightarrow j}(|v_j|) = v_i(|v_i|)$. Given two vectors a and b of the same dimension, we denote by $d(a, b)$ the sum of the Euclidean distances between the individual elements of the same indices in a and b . The EP distance is

$$d_{EP}(v_i, v_j) = \begin{cases} d(v_i, v_{j-i}), & \text{if } |v_i| \leq |v_j| \\ d(v_{i-j}, v_j), & \text{if } |v_i| > |v_j| \end{cases}$$

3.1.4 Classification methods

We have chosen to evaluate the K-nearest neighbors (Knn) and Support Vector Machines (SVM) methods, due to their computational performances and their lack of assumptions on the processes that generate the gesture data. The only input needed to classify an unknown feature sequence is a set of labelled feature sequences (training data). In the Knn approach, a similarity distance is computed between the unknown sequence and the labelled sequences. The new gesture is classified with the label which occurs most in the k labelled sequences that are closest to the new sequence by the given distance. The main idea of the two-class SVM is to first map the training data into a higher dimension space and then compute a separating hyperplane in this space, which defines a decision boundary used to classify unknown data. The mapping is implicitly achieved by means of a kernel function. The separating hyperplane is an optimal one, ensuring maximal margins (distances between the hyperplane and the closest training data). We used the RBF kernel in our experiments, where the kernel function is

$$k(v_i, v_j) = e^{-\gamma(\|v_i - v_j\|^2)},$$

where γ is a model parameter that is empirically determined (here $\gamma = 0.126$).

In general SVM multi-class classification is based on some combination of multiple two-class classifications. We employed the "one-against-one" approach, where, for n classes, one constructs $n(n-1)/2$ two-class classifiers and each one trains data from two distinct classes. Then, for a test vector, a voting procedure is employed to select one of the pairwise-determined classes: the one that gets more votes. If there is more than one class with a maximal number of votes, then the class with the smallest index is chosen. (Empirical

evidence for the precision of gesture recognition for both Euler and Euclidian feature vector representations has been reported elsewhere.)

3.2 Applications

Several applications have been developed that use the gesture services provided by our GLib implementation. In this section, we present some of the applications that are integrated into a smart home environment. The artifacts involved are a swivel chair, a cube, and a soft drink can, each having an embedded wireless InertiaCube orientation sensor. The devices that are controlled by gestures are a TV (controlled by chair and cube gestures), a music player (controlled by chair, cube, and can gestures) and a telephone application (controlled by cube and hand gestures). This layout is shown in Fig. 7.

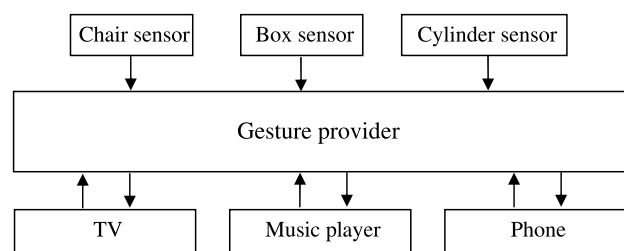


Fig. 7. Applications of the GLib

The chair orientation is used to control the position of a TV screen on the wall such that the screen is always in front of the user. The cube is employed to control the TV, as shown in Fig. 8(a). Flipping gestures are used to select among the possible controls (e.g., channel, contrast) and cube rotation gestures are employed to set values for the selected control. By orienting the chair towards another wall, a music player starts up and the user can control its volume with a Red Bull can, as depicted in Fig. 8(b). When an incoming phone call is received, a flip of the cube displays the name of the caller on the screen and pauses the current player (TV or music). The call is accepted with a *Take* hand gesture, as shown in Fig. 8(c). To close



Fig. 8. Application snapshots. (a) TV control with chair and cube, (b) Music player control with cylinder, (c) Accept a phone call, (d) Terminate the call

the call, a *Throw* gesture is issued, as seen in Fig. 8(d). To see how the GLib is used by applications, consider, for example, the TV application. This registers to be notified of the following gestures: G_{17} , $G_{18}(\text{angle}=3)$, $G_{20}(\text{angle}=1)$. Thus, if the cube is flipped in 0.5 seconds from face 1 to face 5 upwards, then the application is notified by the gesture provider with the gesture $G_{17}(\text{duration}=500, \text{initial_face}=1, \text{final_face}=5)$. Then, if the cube is turned clockwise by 10° at an angular speed of $30^\circ/\text{second}$, the application receives three consecutive gestures $G_{18}(\text{duration}=100, \text{direction}=\text{Clockwise}, \text{angle}=3, \text{face_up}=5)$.

Notice that the box sensor is also used to detect hand gestures, when the box is grasped by the hand and after the box has been flipped with a specific face upwards (say, face 1). The phone application registers for the gestures $G_{17}(\text{final_face}=1)$, $G_{17}(\text{initial_face}=1)$, G_1 , and G_3 .

4. Conclusions and future work

The vision impacting the evolution of pervasive computing is the claim for an intuitive, unobtrusive and distraction free interaction with technology-rich environments. In an attempt of bringing interaction *back to the real world* after an era of keyboard and screen interaction, computers are being understood as secondary artefacts, embedded and operating in the background, whereas the set of all physical objects present in the environment are understood as the primary artefacts, the *interface*. Instead of interacting with digital information via traditional computing means, pervasive computing aims at physical

interaction with digital information, i.e. interaction by manipulating physical artefacts via *graspable interfaces*. It links the "atoms of the physical world" with the "bits of the digital world" in such a way, that every physical artefact is considered as being both representation of and control for digital information. Manipulating physical artefacts in the physical world hence causes the manipulation of their respective associations in the digital world and vice versa.

Motivated by the expressive power of gestures as enablers of intuitive interaction, we proposed in this paper the GLib, a collection of application-independent gestures that is intended as a reference model and standardization effort towards achieving embodied interaction. The GLib contains atomic gestures of body and artefacts, that are modelled in terms of trajectories of sensor data, as well as composite gestures defined by means of several composition operators and a gesture grammar. We presented an implementation of this gestural set that is able to provide gestures to various applications by using inertial orientation sensors for detecting gestures. This allowed us to demonstrate the usage of GLib for achieving embodied interaction in several applications.

Acknowledgements

The authors wish to thank Siemens CT SE 2 (Siemens AG Munich) for their support to conduct this research and several of the experiments, as well as Clemens Holzmann and Dominik Hochreiter for their help with preparing the presentation of the paper.

References

- Amft, O., Junker, H., Trster, G. (2005): Detection of eating and drinking arm gestures using inertial body-worn sensors. Proc. of the 9th IEEE Int. Symposium on Wearable Computers (ISWC'05): 160–163.
- Bahlmann, C., Haasdonk, B., Burkhardt, H. (2002): On-line handwriting recognition with support vector machines. A kernel approach. In Proc. of the 8th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR): 4954.
- Benbasat, A. Y., Paradiso, J. A. (2001): An inertial measurement framework for gesture recognition and applications. Gesture Workshop, LNAI 2298: 9–20.
- Burges, C. J. C. (1998): A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2(2): 1–47.
- Chang, C.-C., Lin, C.-J., LIBSVM (2001): A library for support vector machines. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- Dourish, P. (2001): Where the action is: the foundations of embodied interaction. MIT Press, Cambridge.
- Fishkin, K., Moran, T., Harrison, B. (1998): Embodied user interfaces: towards invisible user interfaces. Proc. of the 7th int. conf. on engineering for human-computer interaction (EHCI'98), Heraklion, Crete, Greece, September 1998.
- Fitzmaurice, G. W., Ishii, H., Buxton, W. (1995): Bricks: laying the foundations for graspable user interfaces. Proc. of the ACM conf. on human factors in computing systems (CHI'95), Denver, Colorado, May 1995.
- Greenberg, S., Fitchett, C. (2001): Phidgets: easy development of physical interfaces through physical widgets. Proc. of ACM symposium on user interface software and technology (UIST 2001), Orlando, Florida, November 2001.
- Holmquist, L. E., Redström, J., Ljungstrand, P. (1999): Token-based access to digital information. Proc. of the first int. symposium on handheld and ubiquitous computing (HUC'99), Karlsruhe, Germany, September 1999.
- Koleva, B., Benford, S., Hui Ng, K., Rodden, T. (2003): A framework for tangible user interfaces. Proc. of the real world user interfaces workshop at the 5th int. symposium on human-computer interaction with mobile devices and services (MobileHCI 2003), Udine, Italy, September 2003.
- Lementec, J. C., Bajcsy, P. (2004): Recognition of arm gestures using multiple orientation sensors: gesture classification. 7th Int. IEEE Conf. on Intelligent Transportation Systems, Washington, D.C., October 3–6 2004: 965–970.
- Lenman, S., Bretzner, L., Thuresson, B. (2002): Computer vision based hand gesture interfaces for human-computer interaction. Technical Report TRITA-NA-D0209, 2002, CID-172, Royal Institute of Technology, Sweden.
- Nielsen, M., String, M., Moeslund, T. B., Granum, E. (2003): A procedure for developing intuitive and ergonomic gesture interfaces for man-machine interaction. Technical Report CVMT 03-01, 2003, Aalborg University.
- Shimodaira, H., Noma, K., Nakai, M., Sagayama, S. (2001): Dynamic time-alignment kernel in support vector machine. Advances in Neural Information Processing Systems 14, NIPS2001, 2: 921–928, December 2001.
- Ullmer, B., Ishii, H. (2000): Emerging frameworks for tangible user interfaces. IBM Syst 39 (3, 4): 915–931.
- Westeyn, T., Brashear, H., Atrash, A., Starner, T. (2003): Georgia Tech Gesture Toolkit: Supporting experiments in gesture recognition. Proc. of ICMI 2003, November 5–7, Vancouver, British Columbia, Canada.
- Williams, A., Kabisch, E., Dourish, P. (2005): From interaction to participation: Configuring space through embodied interaction. Proc. of the Ubicomp 2005, LNCS 3660: 287–304, September 2005.

The Authors

**Alois Ferscha**

received the PhD degree in 1990 at 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, Multi-user Cooperation, Distributed Interaction and Distributed Interactive Simulation. 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. He has been a visiting researcher at the Dipartimento di Informatica, Università di

Torino, Italy, at the Dipartimento di Informatica, Università di Genova, Italy, at the Computer Science Department, University of Maryland at College Park, College Park, Maryland, U.S.A., and at the Department of Computer and Information Sciences, University of Oregon, Eugene, Oregon, U.S.A. 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 Heinz-Zemanek Award for distinguished contributions in computer science.

**Stefan Resmerita**

received his BSc (1996) and MSc (1997) degrees in Electrical Engineering from the Technical University of Iasi, Romania. He obtained a PhD (2003) in Computer Science from the Technion-Israel Institute of Technology. He was 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). Between 2004 and 2006, he was a research associate at the Institute for Pervasive Computing, University of Linz, Austria. During the summer of 2004 he has been a visiting researcher at NASA Ames Research Center, Moffett Field, California. S. Resmerita is currently a postdoctoral researcher and university assistant at the University of Salzburg.