15

Self-organized Evaluation of Dynamic Hand Gestures for Sign Language Recognition

Maximilian Krüger[1], Christoph von der Malsburg[2], and Rolf P. Würtz[1]

[1] Institut für Neuroinformatik, Ruhr-Universität, 44780 Bochum, Germany.
maximilian.krueger@neuroinformatik.rub.de,
rolf.wuertz@neuroinformatik.rub.de
[2] Frankfurt Institute for Advanced Studies, Max-von-Laue-Str. 1, 60438 Frankfurt a. M., Germany.
malsburg@fias.uni-frankfurt.de

15.1 Introduction

Human-computer interaction (HCI) is entering our everyday life. We are welcomed by robot guide-bots in Japan [2] and play computer games using Nintendo's nunchucks [1]. Nevertheless, the revolution is not finished and computer vision is still under development [21]. In this paper we present an organic computing approach to the recognition of gestures performed by a single person in front a monocular video camera.

Visual gesture recognition has to deal with many well-known problems of image processing, like camera noise, object tracking, object recognition and the recognition of a dynamic trajectory. Thus, a gesture recognition system has to show robust feature extraction and adaptation to a flexible environment and signer. It requires properties of an organic computing system, with different autonomous modules cooperating to solve the given problem.

Sign language is a good playground for gesture recognition research because it has a structure, which allows to develop and test methods on sign language recognition first before applying them on gesture recognition. Thus, here we restrict ourselves to working on signs of the British Sign Language (BSL) and concentrate on their manual part.

We have to consider that the projection of the 3D scene onto a 2D plane results in loss of depth information and therefore the reconstruction of the 3D-trajectory of the hand is not always possible. Also the position of the signer in front of the camera may vary. Movements like shifting in one direction or rotating around the body axis must be kept in mind, as well as the occlusion of some fingers or even a whole hand during signing.

Despite its constant structure each sign shows plenty of variation in time and space. Even if the same person performs the same sign twice, small changes in speed and position of the hand will occur. Generally, a sign is affected by the preceding and subsequent sign, an effect called *coarticulation*. Part of the coarticulation problem is that the system has to be able to detect sign boundaries automatically, thus the user is not required to segment the sign sentence into single signs.

Our approach follows the principles of organic computing [10]. We divide problems into different subtasks that are solved by autonomous subsystems. All subsystems are working on-line and therefore can help each other or can flexibly adapt to new situations. Integrating information from different sources, like hand shape, position and their temporal development present, beside the coordination of these processes, the main challenge for creating a recognition system. Our subsystems will autonomously solve part of the problem using organically inspired techniques like *democratic integration* for information merging, *bunch graph matching* for face/object recognition and a modified parallel *hidden Markov model* (HMM) for the recognition of the dynamic trajectories. Each of these techniques learns its knowledge from examples according to the organic computing approach. We explicitly separate gesture recognition into two main processes: feature extraction, which includes localization and tracking of body parts and the recognition process, which uses the selected features. Both processes will be performed during the performance of the sign.

To realize different autonomous units, their environment and communication between them in a software framework we designed a multi-agent system (MAS). Agents show self-x properties like dynamical adaptation to a changing environment (self-healing), perception of their environment and the capability to rate their action (self-reflection).

The structure of the paper is as follows: Section 15.2 gives an overview of previous work in the field of sign language recognition and motivates our ambition to use organic computing. The following section 15.3 describes the multi-agent system architecture, in particular the constructed agents and their use of organic computing methods. Visual tracking is presented in section 15.4 and our approach to sign language recognition in section 15.5. A description of the experiments undertaken and their results can be found in section 15.6. Finally, in section 15.7 conclusions are drawn and future work is outlined.

15.2 Related work

Sign languages, designed to be used by deaf people, are visual languages. They can be characterized by *manual* (hand shape, hand orientation, location and motion) and *non-manual* (trunk, head, gaze, facial expression, mouth) parameters. In this work, we concentrate on manual features and investigate *one-handed* signs performed by the dominant hand only, and *two-handed* signs, which can be performed symmetrically or non-symmetrically.

Sign language recognition (SLR) has to solve three problems, first the reliable tracking of the hands, second robust feature extraction, and third the interpretation of the temporal feature sequence. In the following we present the approaches to these problems that have inspired our work.

Starner and Pentland [13] analyze sign gestures performed by one signer wearing colored gloves. After color segmentation and the extraction of position and shape of the hands their recognition is based on a continuous sequence of signs that are bound to a strict grammar using trained hidden Markov models. Bauer and Krais [3] introduce an HMM-based continuous sign language recognition system by splitting the signs into subunits to be recognized. Image segmentation and feature extraction are simplified by using colored gloves with different colors for fingers and palm. The extracted sequence of feature vectors reflects the manual sign parameters. The same group has built another recognition system that works with skin color segmentation and builds a multiple tracking hypothesis system [24, 20]. They are using HMM as well and extract geometric features like axis ratio, compactness and eccentricity of the hands segmented by skin color.

Instead of colored gloves Vogler and Metaxas [18] use 3D electrical tracking of the wrists. They propose a parallel HMM algorithm to model gesture components and recognize continuous signing sentences. Shape, movement, and location of the right hand along with movement and location of the left hand are represented by separate HMM channels, which are trained with relevant data and features. For recognition, individual HMM networks are built in each channel and a modified Viterbi decoding algorithm searches through all the networks in parallel. Path probabilities from each network that went through the same sequence of words were combined. Tanibata et al. [14] proposed a similar scheme where output probabilities from HMMs modeling the gesture data from right and left hand, were multiplied together for isolated word recognition in the Japanese Sign Language.

The group around Richard Bowden [4, 11, 7] structures the classification mode around a linguistic definition of signed words. This enables signs to be learned reliably from just a handful of training examples. Their classification process is divided into two stages. The first stage generates a description of hand shape and movement using skin color detection. This level of feature is based directly upon those used within sign linguistics to document signs. Its broad description supports generalization and therefore significantly reduces the requirements of further classification stages. In the second stage, Independent Component Analysis (ICA) is used to separate the channels of information from uncorrelated noise. Their final classification uses a bank of Markov chains to recognize the temporal transitions of individual words/signs.

All the presented work is very inspiring and has different interesting approaches to the problems of sign language recognition. Most of these systems are working offline, meaning they collect the feature sequence and do their recognition when the gesture is already performed.

In our approach we divide the problems into different subtasks that are solved by autonomous subsystems. Instead of color tracking we use self-organizing multi-cue tracking for the different body parts. Like in the pa-

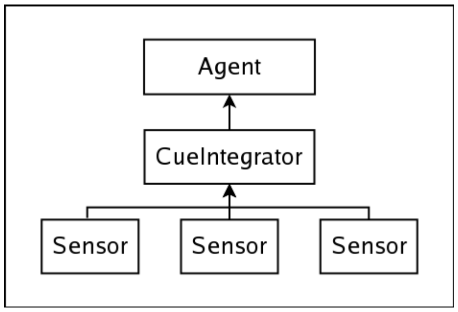

Fig. 15.1. An agent is based on three modules provided with as much flexibility as possible. The interface to the environment and the communication are included in the agent class. There is one cueIntegrator, a module that integrates and interprets the information provided by the sensors.

pers above we use an HMM approach for the temporal recognition, but we extended the idea of HMM by introducing self-organization properties.

15.3 System architecture

Organic computing systems consist of autonomous and cooperating subsystems. We build on a multi-agent system (MAS) developed earlier [9] as a framework for our task. The system consists of three base classes of objects, the **environment**, the **blackboard**, and the **agent**. While environment and blackboard are realized as singleton objects [5], there can be a multitude of different agents. These agents handle tasks ranging from coordination of subprocesses, tracking of an image point, up to the recognition of human extremities.

The information about the world is supplied by the **environment**. Based on the desired functionality of visual tracking and recognition, the environment provides access to image sequences, e.g., the current original color image and its processed versions, the gray value image and the difference image between two consecutive video frames.

Communication within the system is done via the **blackboard**. A message can be quite complex (e.g. carry an image) and has a defined lifetime. Each agent can write messages onto the blackboard and read the messages other agents have posted. Thus, the message handling allows the creation of new agents with specific properties, the change of properties and also the elimination of agents during run-time.

The **agent** is the most interesting entity, it shows the following self-x properties. Agents are autonomous and aware of their state. They perceive their surrounding, to which they can adapt and they communicate with other entities. To implement this behavior, agents have three layers, see figure 15.1. The top layer, called *agent* handles the communication, the fusion center, called *cueIntegrator*, merges the information supplied by one or more *sensors*. Perception of the surrounding is twofold. On the one hand there is message handling via the blackboard, on the other hand an agent can receive information from its sensors, which filter incoming data. Based on the obtained information the agent reaches a decision about further actions.

Gesture recognition is split into the subproblems of object tracking and recognition (object and gesture). Each subproblem is solved by one or more agents. Hence teamwork and a observer/controller architecture are essential. There are three main classes of agents, tracking agents, agents for recognition and agents for control.

We designed **tracking agents** whose task is to follow an object. These agents merge different visual cues like color, texture, movement, etc. Cue fusion is done using democratic integration [16]. This technique offers a self-organized, flexible and robust way of tracking and will be explained in section 15.4. Agents that provide world knowledge stored in the system are called **recognition agents**. This includes knowledge for face recognition and static hand gesture recognition. Training the recognition agents, i.e. learning world knowledge from examples is also a crucial task requiring organic computing methods see [23]. As the system should act independently from user interaction **controlling agents** are responsible for solving the conflicts that might occur during execution.

15.4 Visual tracking

Visual tracking of head, left and right hand is done by a cooperation of globally and locally acting agents that are organized in a hierarchical network [9]. A global working agent scans the image for regions of interest, defined by skin colored and moving blobs. A controlling agent supervises the tracking. It collects the region of interest messages, checks whether they are already tracked and if not instantiates a new tracking agent. The visual appearance of the hand is a function of several factors, which hand classifier and tracking agents have to take into account, including pose, lighting, occlusion and intra/inter-signer variations.

Object tracking is performed by tracing an image point on the object. Tracking agents take on this task by scanning on the new frame the local surrounding of the last target position of the previous image. Hence they are called local agents. Due to lack of robustness of single cues, tracking should not rely on a single feature, thus each tracking agent integrates the results of four different information sources, namely pixel template, motion, motion prediction, and color, each realized in a sensor. The agent's cueIntegrator calculates the result as a weighted average of saliency maps derived from the different sensors. The result is fed back to the sensors and serves as the basis for two types of adaptation. First, the weights of the sensor are adapted according to their agreement with the overall result. Second, sensors are allowed to

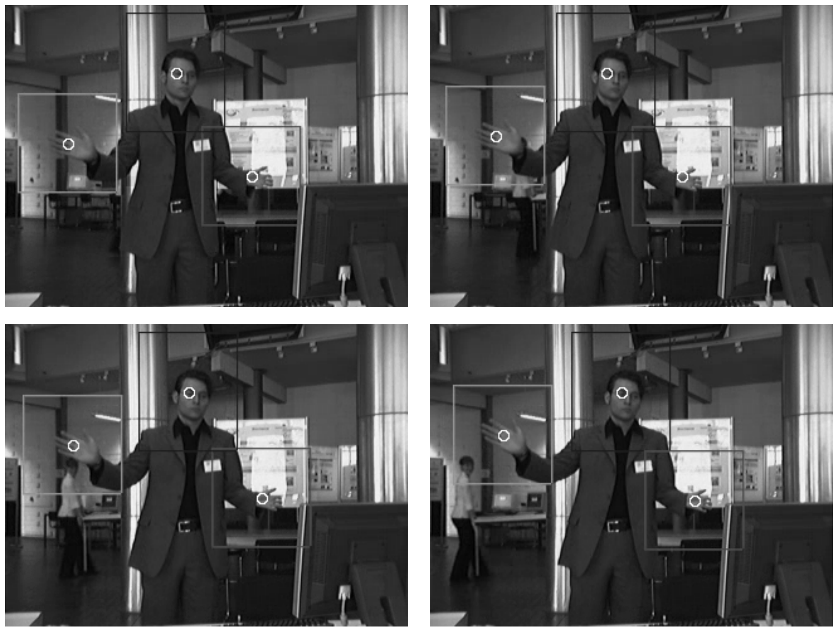
Fig. 15.2. In this tracking sequence head and hands were found. The identity of the objects is visualized by the gray value of the rectangles, which delineate the search region of each tracking agent. Moving skin color in the background is ignored.

adapt their internal parameters in order to have their output better match the determined collective result. This integration scheme is called *democratic integration* [16] and will be explained below.

After tracking the object on the current frame, the tracking agent evaluates its success and posts a message containing the actual position, the contour and an image of the target. This information is passed to the recognition agents trying to identify the object. The face recognition agent, for instance, performs face detection using bunch graph matching [22]. Once a face has been found left and right hands are determined via their position relative to the face. The tracking agent adds its identification to its messages, see figure 15.2. To further support identification we added two recognition agents to identify the static hand gesture. The first recognition agent matches a gallery of learned bunch graphs on the image to identity the texture of the static hand gesture [17]; the second one matches the contour against a gallery of learned contours.

Since there might be skin colored moving blobs in the background of a real-world setting, which are not connected to a hand or the head, agents that track an unknown object over a period of time will delete themselves. This self-healing of the global system is also enforced if the agent is not content with its tracking results. After this analysis the tracking continues.

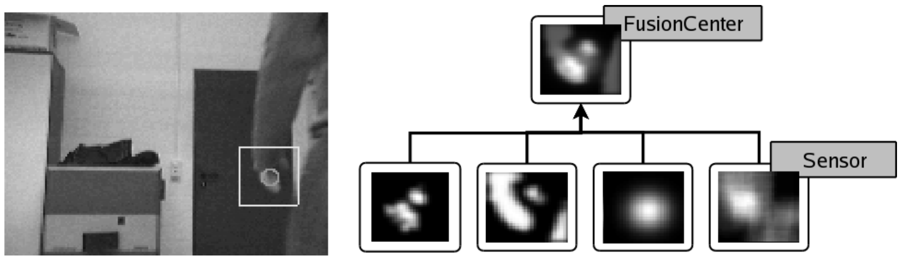

Fig. 15.3. Tracking agent in use, on the left we see the tracking result marked with the circle. The rectangle shows the border of the agent's search region. On the right we see the similarity maps created by the different sensors, from left to right: color, motion, motion prediction and pixel template. The fusion center shows the result of the information integration.

Using *democratic integration* the different cues, namely color, motion, motion prediction and pixel template, are integrated to agree on one result. After this decision each cue adapts toward the result agreed on. In particular, discordant cues are quickly suppressed and re-calibrated, while cues having been consistent with the result in the recent past are given a higher decision weight in future.

Integrating information using democratic integration relies on two assumptions. First, the cues must be statistically dependent, otherwise there is no point in trying to integrate them. Second, the environment must exhibit a certain temporal continuity. Without that, any adaptation would be useless.

As shown in figure 15.3 all cues are working on the two dimensional search region of the agent. Each sensor i provides a similarity map $\mathsf{M}_i(x,t)$ at time t, that shows the image similarity at each coordinate x with an agent-specific and adaptable prototype template $\mathsf{P}_i(t)$. To integrate the similarity maps to an overall map $\mathsf{R}(x,t)$, they are weighted and summed up

$$\mathsf{R}(x,t) = \sum_i r_i(t)\mathsf{M}_i(x,t), \tag{15.1}$$

The weights $r_i(t)$ are part of the self-controlling of each sensor and will further be called reliability. The reliabilities are normalized such that $\sum\limits_i r_i(t) = 1$. The target position $\hat{x}(t)$ is found by scanning the overall similarity map for the maximal entry

$$\hat{x}(t) = \arg\ \max_x \left\{\mathsf{R}(x,t)\right\}. \tag{15.2}$$

To rate its action each tracking agent analyzes the similarity value at the target position $\hat{x}(t)$. If the value is above a threshold, the image point has been found, otherwise, the tracking agent has failed to track it. Using the information of the similarity value and the target position each sensor is able to update its reliability and to adapt its prototype to the new situation. The adaptation depends on the tracking result — if the target was found in the image the prototype adapts towards the actual parameters at the target position $\hat{x}(t)$. Otherwise it adapts towards its initial values. We refer to [16] for a complete description of the update strategies.

Equation (15.2) has proven to work well on small objects [16, 8] with a unimodal similarity map. Larger objects can create a multimodal similarity map with more than one peak. Hence we modified the search of the target position by thresholding the map and from the remaining peaks we calculate the center of gravity. The new target position might be located outside the object or might be a bad point to track, but our experiments showed that this was not the case for different tracking scenarios and that tracking became more stable.

15.5 Recognition

In the previous section we presented the MAS system for visual tracking of head and hands. Tracking agents provide position and static hand gesture information of the object for nearly each frame. In this section we describe the subsystem that collects the information about the trajectory of left and right hand and the corresponding static hand gesture. The information for sign language recognition is merged by using an extended self-organizing hidden Markov model architecture. Recognition needs to be stable and robust enough tho deal with the changes in speed and position of a hand, which will even occur if the same person is performing the same sign twice. Hidden Markov models (HMMs) can solve these problems. Their ability to compensate time and amplitude variations of signals has been amply demonstrated for speech and character recognition. Before we discuss our approach to recognition using an extension to HMM we review the aspects of HMM theory relevant to this paper.

15.5.1 Theory of hidden Markov models

This section briefly discusses the theory of hidden Markov models (see figure 15.4 as an example of a left-right architecture). It follows the classic paper by Rabiner [12], which we recommend for a more detailed description of this topic. A hidden Markov model is characterized by the following:

1. N, the number of states in the model. We denote the individual states as $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$, and the state at time t as q_t.
2. M, the number of distinct observation symbols per state, i.e. the discrete alphabet size. The observation symbols correspond to the physical output of the system being modeled. We denote the individual symbols as $\mathbf{V} = \{v_1, v_2, \ldots, v_M\}$. If the observation is in a continuous space, M is replaced by an interval of possible observations.

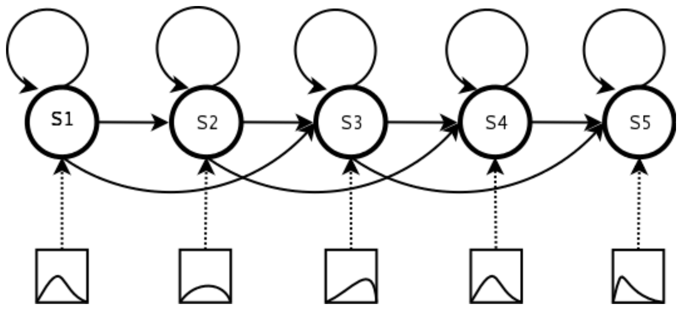

Fig. 15.4. HMM with the left-right (Bakis) topology, typically used in gesture and speech recognition. The solid lines denote the transition probabilities and set $a_{ij} = 0 \quad \forall \quad j < i \quad \wedge \quad j > i+2$. The dotted line connects a continuous observation distribution to the belonging state (circle).

3. The transition probability distribution $\mathsf{A} = \{a_{ij}\}$ where

$$a_{ij} = P\left(q_{t+1} = S_j \mid q_t = S_i\right), \quad 1 \leq i, j \leq N \tag{15.3}$$

   and

$$\sum_j a_{ij} = 1. \tag{15.4}$$

   Assuming that the state transition probability a_{ij} from state S_i to state S_j only depends on the preceding state (first order Markov process). For the special case that any state can reach any other state in a single step, we have $a_{ij} > 0$, for all i, j. For other types of HMM, we would have $a_{ij} = 0$ for one or more i, j pairs.
4. The observation probability distribution $\mathsf{B} = \{b_j(k)\}$ in state j, where

$$b_j(k) = P\left(v_k \text{ at } t \mid q_t = S_j\right), \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \tag{15.5}$$

   and

$$\sum_k b_j(k) = 1. \tag{15.6}$$

   The observation probability distribution can be discrete or continuous.
5. The initial distribution $\boldsymbol{\pi} = \{\pi_i\}$ where

$$\pi_i = P\left(q_1 = S_i\right), \quad 1 \leq i \leq N. \tag{15.7}$$

A complete specification of a HMM consists of two model parameters (N and M), the specification of observation symbols, and the specification of the three probabilistic measures A, B and $\boldsymbol{\pi}$. For convenience, we use the compact notation

$$\boldsymbol{\lambda} = (\boldsymbol{\pi}, \mathsf{A}, \mathsf{B}) \tag{15.8}$$

to indicate the complete parameter set of the model.

Due to their doubly stochastic nature HMMs are very flexible and became quite famous in the gesture recognition community (section 15.2). The art of HMM design lies in the specification of their topology of allowed transitions, and the features to be observed. There are three basic problems of interest that must be solved for the HMM to be useful in real-world applications:

Evaluation problem: Given the observation sequence $\mathbf{O} = O_1, O_2, \ldots, O_T$, and the model $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \mathsf{A}, \mathsf{B})$, how do we efficiently compute $P(\mathbf{O} \mid \boldsymbol{\lambda})$, the probability of generating the observation sequence given the model?

Decoding problem: Given the observation sequence $\mathbf{O} = O_1, O_2, \ldots, O_T$, and the model $\boldsymbol{\lambda}$, how do we choose a corresponding state sequence $\mathbf{Q} = q_1, q_2, \ldots, q_T$ which is meaningful in some sense (i.e., best "explains" the observations)?

Estimation problem: How do we train the HMM by adjusting the model parameters $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \mathsf{A}, \mathsf{B})$ to maximize $P(\mathbf{O} \mid \boldsymbol{\lambda})$?

The standard way to recognize a gesture out of a set G is to train a HMM $\boldsymbol{\lambda}_g$ for every single gesture $g \in G$ and after the observation sequence is recorded, start the calculation of $P(\mathbf{O} \mid \boldsymbol{\lambda}_g)$ for every HMM $\boldsymbol{\lambda}_g$ (see section 15.5.3.1). The solution of the *evaluation problem* is used for recognition where the model $\boldsymbol{\lambda}_g$ which produces the highest probability of describing the observation sequences

$$g = \arg\max_{g} P(\mathbf{O} \mid \boldsymbol{\lambda}_g) \tag{15.9}$$

is deemed to be the recognized gesture.

15.5.2 Organic modification of the hidden Markov model

The topology of our HMMs is an extension of the Bakis model as seen in figure 15.4 and will be further explained in section 15.5.3.1.

The observed features (which we will call observations from now on) are provided by the tracking module described in section 15.4, namely position, texture and contour of left and right hand. Each kind of observation has a particular degree of uncertainty, the position can vary on the object, texture and contour might not be accurately determined due to blurring or erroneous segmentation. Thus, we use the organic computing principle that distributed information is advantageous for robustness and split the observations into different channels. This parallel HMM (PaHMM) structure had been used by Vogler [18] and Tanibata [14], who divided the observations for left and right hand and trained a HMM for each hand. The independence of the channels has been demonstrated in [19]. Consequently, we go one step further give the system better generalization power if every observation has its own HMM, instead of putting them into one observation vector. Therefore, in our system a gesture g will be represented by six channels that separate the position $\mathbf{y}$ relative to the head, texture $\boldsymbol{\tau}$ and contour $\mathbf{c}$ for left and for right hand. Another point is that if we assume that parallel HMMs model the parallel

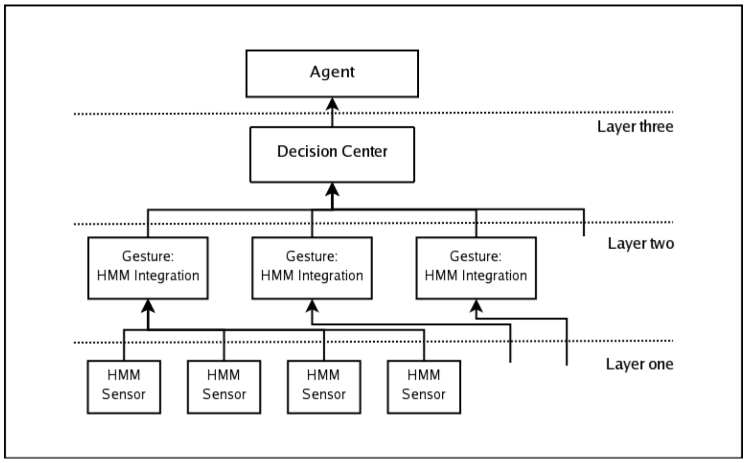

Fig. 15.5. Layout of the recognition agent. It is hierarchically organized, at the bottom we have the HMM sensor modules, four of them for each gesture. They collect the information and calculate their observation probability. The HMM sensor results are merged in the Gesture HMM Integration modules, of which there is one for each learned gesture. The Decision Center decides about the most probable gesture.

processes independently, they can also be trained independently, and do not require consideration of the different combinations at training time.

15.5.3 The HMM recognition agent

Administration of gesture recognition is hosted in the recognition agent. During a recognition cycle the agent starts collecting the observations. Gesture recognition is done from bottom to top (see figure 15.5 and algorithm box 1). Layer one consists of the trained HMM channels for each gesture. In terms of our MAS they are called HMM Sensors. They independently calculate the actual observation probability and pass this information to layer two. The Gesture HMM Integration modules in layer two represent the learned gestures. Each module fuses the information of its channels to compute a decision about the probability that the performed gesture is similar to the one represented by the Gesture HMM Integration module. Finally, in the Decision Center of layer three, the results of the Gesture HMM Integration modules are compared to determine which gesture is the most probable.

15.5.3.1 Layer one: modified HMM

Starting at the sensor layer we are mainly interested in solving the evaluation problem mentioned above. Before we present our extensions of the HMM idea, we want to outline the Forward-Backward approach to calculate $P(\mathbf{O} \mid \boldsymbol{\lambda})$ and motivate our modifications.

Forward-backward algorithm

The forward-backward algorithm [12] solves the evaluation problem by calculating the $P(\mathbf{O} \mid \boldsymbol{\lambda})$ using the forward variable $\alpha_t(j)$, which is defined as

$$\alpha_t(j) = P(O_1, O_2, \ldots, O_t, q_t = S_j \mid \boldsymbol{\lambda}), \tag{15.10}$$

the probability of the partial observation sequence $O_1, O_2, \ldots, O_t$ and state S_j at time t, given the model $\boldsymbol{\lambda}$. The forward variable is solved inductively, as follows:

1. Initialization:
$$\alpha_1(j) = \pi_j b_j(O_1), \quad 1 \le j \le N. \tag{15.11}$$
2. Induction:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \quad 1 \le t \le T-1 \quad 1 \le j \le N. \tag{15.12}$$
3. Termination:
$$P(\mathbf{O} \mid \boldsymbol{\lambda}) = \sum_{i=1}^{N} \alpha_T(i). \tag{15.13}$$

The backward variable $\beta_t(i)$ is calculated in a similar manner and is defined as the probability of the partial observation sequence $t+1$ to the end, given state S_i at time t. Each of these variables or a combination of them can be used to solve the *evaluation problem.*

Using the forward-backward algorithm and especially the multiplication of probabilities in equation (15.12), recognition would not be robust to:

1. a missing observation,
2. a wrongly classified static hand gesture that was not in the training data[3],
3. the observation sequence taking longer than the learned ones.

Problem one can be solved by having perfect tracking and perfect classification of static hand gestures. The other two problems become less crucial when collecting more training data. But it is our aim to develop a system that works robustly in a self-organized way under real-world conditions, the conditions of sparse data.

HMM sensor

The flexibility of the HMM depends on the training data, which determines the transition probabilities a_{ij} and the observation probability distributions $\mathsf{B} = \{b_j(k)\}$ of the model. Under real world conditions the HMM will be confronted with unknown, not learned, observations or variations in the dynamics caused

[3] Using a discrete observation distribution, a zero will be returned for the observation probability of a symbol that has not been learned for the particular state.

by, e.g., missing tracking information, blurring, etc. To face these problems we split up the doubly probabilistic method of the HMM by introducing a self-organized transition between the states. In our approach we are using a strict left-right model with $\pi_1 = 1, \pi_i = 0$ for $i \neq 1$ and $a_{ij} = 1$ if $j = i + 1$, $a_{ij} = 0$ else. The number of states N is equal to the longest learning sequence of the gesture's training set. Instead of the transition probability matrix A where the transitions are learned, the a_{ij} are replaced by a weighting function $w_t(u)$. Equation (15.12) will become

$$\alpha_{t+1}(j) = \alpha_t(i) \underbrace{a_{ij}}_{=1} w_t(u) b_j(O_{t+1}), \tag{15.14}$$

and therefore equation (15.13) becomes

$$P(\mathbf{O} \mid \boldsymbol{\lambda}) = \alpha_T(N). \tag{15.15}$$

Equation (15.14) allows the HMM to perform on-line gesture recognition and computes its probability on every frame.

The weighting function of each channel is Gaussian

$$w(u) = \exp\left(-\frac{u^2}{2\sigma}\right), \tag{15.16}$$

where $u = [0, \infty]$ is a measure of uncertainty. Starting with a maximal certainty of $u = 0$ at the beginning of the recognition, the modified HMM (mHMM) checks whether the received observation O_t is presented in the observation distribution $b_i(O_t)$ of the actual state i. If the result is not satisfactory, i.e., below a recognition threshold, the mHMM can pass the observation to the next state $i+1$, check again, and pass it further to state $i+2$ if necessary. If the observation does not even match at state $i+2$ it will be ignored. Each of these transitions is punished by increasing the uncertainty u and thus lowering the weighting function. To reinstall its certainty, the mHMM recovers with every recognized observation by decreasing u. If the observation has been recognized the system switches to the next state.

To come back to our HMM recognition agent, the task of the HMM sensor (layer one in figure 15.5) is to calculate its weighted observation probability $w_t(u)b_j(O_t)$ for the actual frame.

15.5.3.2 Layer two: gesture HMM integration

Each learned gesture g has a Gesture HMM Integration unit in layer two. By merging the information of its six sensors the Gesture HMM Integration unit computes the quality Q_g of the gesture g matching the observation sequence.

Due to the nature of the observation, the HMM sensors have different probability distributions. The position information is stored in a continuous

Algorithm 1: Recognition is hierarchically organized using three layers. The characteristic of each layer is its information integration. Layer one, the HMM Sensor, compares the received observation with its observation probability function. Layer two comprise the HMM Integration unit of each learned gesture and integrates the information received from layer one. The top layer compares the results from the HMM Integration units. The Decision Center determines the most probable gesture and manages the inhibition.

```
while not at end of gesture sequence do
    /* ******************************************************** */
    /* Layer one: HMM sensor                                    */
    /* ******************************************************** */
    foreach HMM sensor do
        calculate observation probabilities;
    end
    /* ******************************************************** */
    /* Layer two: HMM Integration unit                          */
    /* ******************************************************** */
    foreach HMM Integration unit do
        compute $\varrho$ to fuse the information of position, texture and contour;
        calculate the actual quality $Q_a$;
        update the overall quality $Q_g$;
        control the activation using $Q_g$, $\xi_{start}$ and $\xi_{stop}$ ;
    end
    /* ******************************************************** */
    /* Layer three: Decision Center                             */
    /* ******************************************************** */
    if $\exists$ HMM Integration unit that reached its $\zeta_{min}$ then
        choose HMM Integration unit with highest $Q_g$
          as current winner;
    end
    if $\zeta_{winner} == 1$ then
        reset all HMM Integration units;
    end
    else
        /* inhibit all gestures                                 */
        search for the maximal quality $Q_{max}$;
        foreach HMM Integration unit do
            subtract $Q_{max} from\ Q_g$;
        end
    end
end
```

Result: last winner will be chosen as recognized gesture.

distribution using Gaussian mixtures. The information of the static hand gesture, the identified most similar contour and bunch graph, are stored separately using a discrete probability distribution, which is realized as histogram based on the appearance of the elements of the contour alphabet or bunch graph alphabet, respectively.

The continuous distribution offers a more flexible way to evaluate the observation, as we have a Euclidean distance for our position observations. In our discrete feature space the concept of similarity, or distance, cannot be assumed to be Euclidean. Therefore, we use position as the basis for our recognition. The aim of sensor integration is the computation of the overall quality Q_g for the single gesture g. At the beginning or after a reset (see below), the Q_g is initialized with zero. To get rid of possible multiplications with small numbers when estimating $\alpha_{t+1}(j)$ using equation (15.14), we will work with the logarithms of the probabilities sent by the sensors and therefore obtain:

$$\alpha_{t+1}(j) = \alpha_t(i) + \log(w_t(u)b_j(O_{t+1})). \tag{15.17}$$

Thus, for every frame we receive the log probability l of the left hand position $l_{lh}(\mathbf{y})$, left hand contour $l_{lh}(\mathbf{c})$, left hand texture $l_{lh}(\boldsymbol{\tau})$ and right hand $l_{rh}(\mathbf{y})$, $l_{rh}(\mathbf{c})$, $l_{rh}(\boldsymbol{\tau})$. To calculate the actual gesture quality Q_a of the current frame, we first weight the position probabilities of the two hands and add them to Q_a

$$Q_a = w_{lh}l_{lh}(\mathbf{y}) + w_{rh}l_{rh}(\mathbf{y}). \tag{15.18}$$

Thereby we focus on the dominant hand by setting $w_{rh} = 0.7$ and $w_{lh} = 0.3$. Although position is already a good observation for gesture recognition we have to add the static hand gesture information to obtain better results. But as mentioned above, recognition of the static hand gesture might not be stable on every frame, especially when the hand is moving. Hence, we decided to integrate the bunch graph and contour information using a rewarding function ϱ. This function rewards only if position and static hand gesture information are correlated. Correlation does not necessarily mean that the mHMMs have to be in the same state i. For each hand the $l(\mathbf{c})$ and $l(\mathbf{y})$ or $l(\boldsymbol{\tau})$ and $l(\mathbf{y})$ just have to be above a threshold θ. The reward is linked to the probability for the static hand recognition $l(\mathbf{c})$, $l(\boldsymbol{\tau})$ respectively

$$\varrho(x) = (x - \theta)H(x - \theta) \ ; \quad H : \text{Heaviside step function} \tag{15.19}$$

and will be added to Q_a. After computing Q_a we update the Q_g by adding Q_a. Without the static hand gesture information the Q_g would decrease with increasing gesture length. By introducing ϱ we allow the Q_a and Q_g to become positive and therefore Q_g cannot be transferred into a probability again.

Each Gesture HMM Integration unit has two states, active and inactive. In the active state the gesture is certain that it could match the data and by increasing the states of the HMM Sensor the recognition is continuously following the incoming observations. Increase of the state of the HMM Sensor is a cue for the similarity of the learned gesture to the performed sign. A gesture becomes active if the Q_a of the first state is above the threshold ξ_{start}. Otherwise, the gesture is inactive, which means that all the connected HMM Sensors are set to their initial state and all the parameters like the uncertainty u of each sensor and the Q_g are reset to zero. An active gesture can become inactive if the Q_g drops below a threshold ξ_{stop}. ξ_{start} and ξ_{stop} have global values and allow the system to reset a gesture autonomously to restart the recognition during the performance of the sign. We developed this active/inactive mode to handle the problem of coarticulation (the frames between two gestures) and the case where we have similar beginning for one or more gestures and only the following frames will decide which gesture is performed.

15.5.3.3 Layer three: decision center

Only active gestures will receive the attention of the Decision Center in layer three. The Decision Center compares the results of the Gesture HMM Integration units and determines which gesture is the most probable so far.

The autonomy of the Gesture HMM Integration units in choosing a starting time prohibits the Decision Center from using equation (15.9) directly and declare the gesture g with the highest value of Q_g the recognized one. In that case the Decision Center would wrongly favor gestures that just started over gestures that already accumulated similarity. Thus, we coupled the recognition to the progress, the actual state, of the HMM Sensor by means of a confidence value ζ, which is computed by the ratio of the actual state of the sensor to the maximal number of states N of the mHMM. This confidence value is a measure of certainty. Only gestures that are above a threshold of ζ_{min} will be handled by the Decision Center. This minimal confidence ζ_{min} is individual for each gesture and is computed as the ratio of its shortest to its longest sequence in the training set. Out of the gestures that reached their ζ_{min} the Decision Center chooses the one with the highest Q_g to be the most probable gesture that represents the observation sequence so far. This method favors short gestures that only need a small amount of recognized frames to reach their ζ_{min}. Therefore, all gestures are inhibited by the gesture with the highest Q_g to become inactive before they reach the needed confidence value. If a gesture reaches a confidence value of one, it is deemed recognized already before termination of the sequence, and a reset signal is sent to all connected Gesture HMM Integration units.

15.6 Experiments and results

15.6.1 Sign language data

Our training and testing data consist of signs of the British Sign Language (BSL) that were kindly provided by Richard Bowden. The data is a continuous

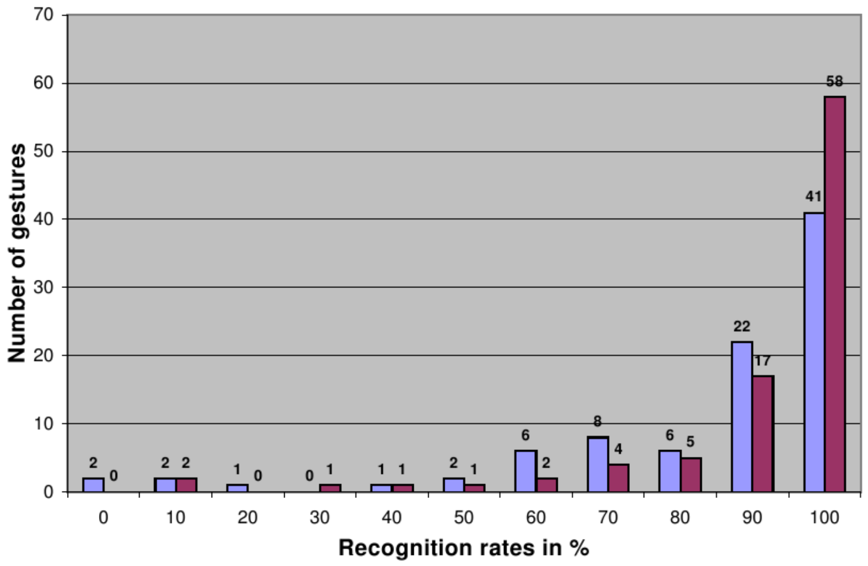

Fig. 15.6. Recognition started with the beginning of the gesture. The histogram shows the result for the experiment using only position and contour information in light color and in dark color the result when integrating position, contour and texture information.

movie with ground truth information about the beginning and the end of each gesture. We have 91 different signs performed with 10 repetitions by one signer, a total of 29219 images to be processed. The sequence length ranges from 11 to 81 frames for the gestures and even within the gestures the sequence length shows differences of around 50 percent, e.g., the length of gesture “live” ranges from 18 to 41 frames. The signer is wearing colored gloves, hence for training the exact position of the hand (the center of gravity), the texture and the shape contour could be automatically determined. To calculate relative positions for the hands a bunch graph face detection was run on the images. The segmented hands allow the automatic creation of bunch graphs for left and right hand for each frame. They where clustered by matching each bunch graph on the other images and adding the image if the matching similarity was above a certain threshold. The extracted contours were clustered using standard vector quantization as described in Gray [6] to gain an alphabet of representative hand shapes. As a result we obtain observation sequences for relative hand position and static hand gesture, which are used to train the mHMM.

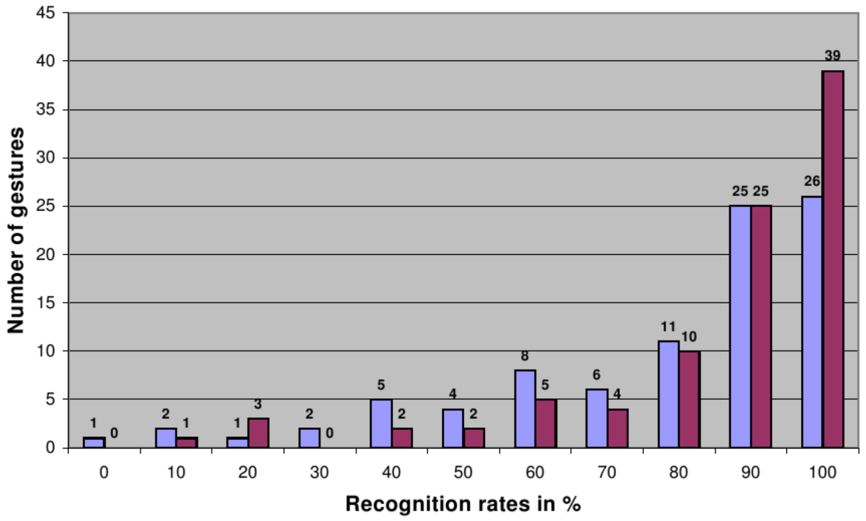

Fig. 15.7. Recognition started 10 frames in front of the the beginning of the gesture to examine the effect of coarticulation. The histogram shows the result for the experiment using only position and contour information in light color and in dark color the result when integrating position, contour and texture information

15.6.2 Experiments

The recognition experiments were performed using a leave-one-out procedure, where for the testing gesture all sequences excluding the one that is tested were used to build the mHMMs. Therefore, we perform ten recognition experiments per gesture. At the end of each performance, the final most likely gesture is deemed to be the overall recognized gesture.

We tested our system in two sets of recognition experiments. In the first set recognition starts at the known beginning of the sign, while in the second recognition experiment we included the coarticulation of a previous gesture. To simulate coarticulation we started the recognition 10 frames before the ground truth starting time.

On both sets we tested the benefit of multi-cue integration and the stability of of the system concerning missing data by running each set two times. In the first run we integrated position, contour and texture information for recognition and then we dropped the texture information and only integrated position and contour information in the second run.

The distribution of the recognition rates is shown in figure 15.6 for the given start of the gesture and figure 15.7 for the coarticulation. The results are presented in histogram style, where we plotted the recognition rate (light bars for the recognition experiment applying only position and dark bars for adding texture information) against the number of recognized gestures. For

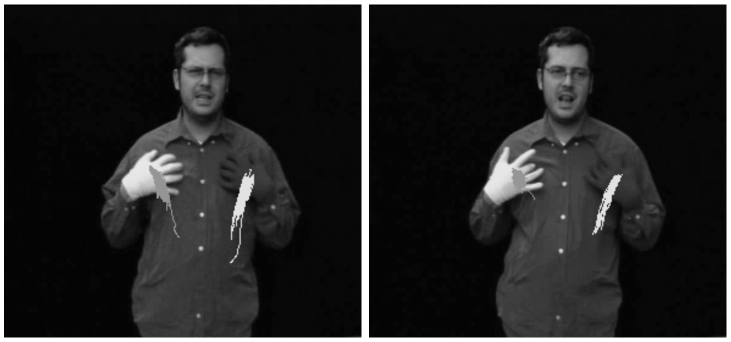

Fig. 15.8. The trajectories and static hand gestures of the signs "excited_interested" (left) and "live" (right) are very similar. Therefore, the shorter sign live dominates the recognition of the excited_interested performance. The integration of non-manual observation like a grammar or facial expression should help to differentiate between similar signs.

example, using figure 15.6 we have recognized eight gestures with a recognition rate of 70% when only using position and contour information.

Given the start of the gesture we achieve an average recognition rate of 90% if we integrate position contour and texture. The mean recognition rate is reduced to 84% if we exclude the texture information. Taking the first run we receive a mean recognition rate of 90% and higher for over the half of our gestures. By analyzing the gestures with lower recognition rates of 0 to 10%, these gestures were mainly dominated by a very similar gesture that have a lower sequence length. For example the "excited_interested" gesture is dominated by the shorter "live" gesture, that shows a very similar trajectory and similar static hand gestures as can be seen in figure 15.8. The difference of the trajectories is small compared to the inter-sign variations that can occur in other sign like the "different" and the "bat" gesture trajectories that are plotted in figure 15.9. This misclassification trap is caused by the self-organizing property of the system. All known gestures are in a loop and are waiting to become active by passing the activation threshold ξ_{start}. Therefore as seen for the low recognized gestures they are likely to be dominated by similar shorter gestures and this might be a good reason to include grammar or other non-manual observation like facial expression to future systems.

The benefit of this autonomy to start the recognition becomes obvious in our second set. Running the experiment with the same data and parameters we achieve a mean recognition rate of 85% or 78% respectively for the second run. The recognition system shows just a 5% difference between a fixed and a self-organized start of the recognition.

Comparing recognition rates for sign language recognition is a difficult task, because every group has its own data. Nevertheless we have to admit that our recognition rates are below the results of von Agris et al [20] with 97.9%

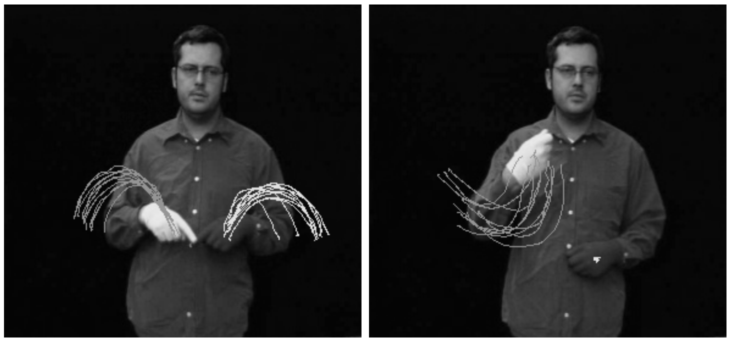

Fig. 15.9. The signs "different" (left) and "bat" (right) shown with the trajectory differences of ten repetitions by the same signer.

for a database of 153 isolated signs and Kadir et al [7] who have a recognition rate of 92% for a lexicon of 164 words. The strength of our system is the autonomy of the recognition process to handle the effect of coarticulation, which have not been investigated by von Agris and Kadir.

15.7 Conclusion

We have presented an approach to gesture recognition by organic computing technology. We built a software framework to design and test multi-agent systems. The characteristics of a multi-agent system are autonomous and cooperating units. Organic computing principles like divide and conquer, learning from examples and self-control have been used for object tracking and sign language recognition. Both systems are running simultaneously.

For gesture recognition we modified a standard HMM architecture by introducing two types of information, a more reliable channel as a basis and a weaker one. Both are integrated by using a correlation and rewarding scheme. Another innovation is the competition of the learned gestures during the recognition process. In addition to satisfying recognition results the autonomy of the system allows to handle the problem of coarticulation.

Only simple features like the position, contour and texture of the hands have been applied, we resigned to grammar or a high level description. To learn a grammar or a high level description would be an interesting challenge for future projects. In the near future we plan to integrate facial expression recognition of Tewes et al [15] as a new HMM sensor and to run the system on more data to examine its signer independence ability.

Acknowledgments

We thank Richard Bowden from the University of Surrey for providing his sign language data. Funding by the DFG in the SPP "Organic Computing" (MA 697/5-1 and WU 314/5-2) is gratefully acknowledged.

References

1. http://de.wii.com/, 2006.
2. http://www.kokoro-dreams.co.jp/english/robot/act/index.html, 2006.
3. B. Bauer and K.-F. Kraiss. Video-based sign recognition using self-organizing subunits. In *ICPR (2)*, pages 434–437, 2002.
4. R. Bowden, A. Zisserman, T. Kadir, and M. Brady. Vision based interpretation of natural sign languages. In *International Conference on Computer Vision Systems, Graz. Austria*, 2003.
5. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons, August 1996.
6. R. M. Gray. Vector quantization. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 75–100. Kaufmann, San Mateo, CA, 1990.
7. T. Kadir, R. Bowden, E. J. Ong, and A. Zisserman. Minimal training, large lexicon, unconstrained sign language recognition. In *Proceedings of the 15th British Machine Vision Conference, Kingston*, 2004.
8. O. Kähler and J. Denzler. Self-organizing and adaptive data fusion for 3d object tracking. In U. Brinkschulte, J. Becker, D. Fey, C. Hochberger, T. Martinetz, C. Müller-Schloer, H. Schmeck, T. Ungerer, and R. Würtz, editors, *ARCS 2005 – System Aspects in Organic and Pervasive Computing – Workshops Procedeedings, Innsbruck Austria, March 14–17*, pages 109–116. VDE Verlag, Berlin, Offenbach, 2005.
9. M. Krüger, A. Schäfer, A. Tewes, and R. P. Würtz. Communicating agents architecture with applications in multimodal human computer interaction. In P. Dadam and M. Reichert, editors, *Informatik 2004*, volume 2, pages 641–645. Gesellschaft für Informatik, 2004.
10. C. Müller-Schloer, C. von der Malsburg, and R. P. Würtz. Aktuelles Schlagwort: Organic Computing. *Informatik Spektrum*, 27(4):332–336, 2004.
11. E. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *Face and Gesture Recognition*, pages 889–894, 2004.
12. L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb. 1989.
13. T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *International Workshop on Automatic Face and Gesture Recognition*, pages 189–194, 1995.
14. N. Tanibata, N. Shimada, and Y. Shirai. Extraction of hand features for recognition of sign language words. In *Proceedings International Conference on Vision Interfaces*, pages 391–398, 2002.
15. A. Tewes, R. P. Würtz, and C. von der Malsburg. A flexible object model for recognising and synthesising facial expressions. In T. Kanade, N. Ratha, and A. Jain, editors, *Proceedings of the International Conference on Audio- and*

Video-based Biometric Person Authentication, LNCS, pages 81–90. Springer, 2005.
16. J. Triesch and C. von der Malsburg. Democratic integration: Self-organized integration of adaptive cues. *Neural Computation*, 13(9):2049–2074, Sept. 2001.
17. J. Triesch and C. von der Malsburg. Classification of hand postures against complex backgrounds using elastic graph matching. *Image and VisionComputing*, 20(13):937–943, 2002.
18. C. Vogler and D. N. Metaxas. Parallel hidden markov models for american sign language recognition. In *ICCV (1)*, pages 116–122, 1999.
19. C. Vogler and D. N. Metaxas. Handshapes and movements: Multiple-channel american sign language recognition. In *Gesture Workshop*, pages 247–258, 2003.
20. U. von Agris, D. Schneider, J. Zieren, and K.-F. Kraiss. Rapid signer adaptation for isolated sign language recognition. In *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, page 159, Washington, DC, USA, 2006. IEEE Computer Society.
21. C. von der Malsburg. Vision as an exercise in Organic Computing. In P. Dadam and M. Reichert, editors, *Informatik 2004*, volume 2, pages 631–635, 2004.
22. L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.
23. R. P. Würtz. Organic Computing methods for face recognition. *it – Information Technology*, 47(4):207–211, 2005.
24. J. Zieren and K.-F. Kraiss. Robust person-independent visual sign language recognition. In J. S. Marques, N. Pérez de la Blanca, and P. E. Pina, editors, *Pattern Recognition and Image Analysis, Second Iberian Conference IbPRIA, Estoril, Portugal*, volume 3522 of *LNCS*, pages 520–528. Springer, 2005.