Dynamic Gesture Recognition Using Neural Networks; A Fundament for Advanced Interaction Construction

Klaus Boehm, Wolfgang Broll, Michael Sokolewicz

ZGDV – Zentrum fuer Graphische Datenverarbeitung e.V, Darmstadt, Germany

ABSTRACT

Interaction in Virtual Reality environments is still a challenging task. Static hand posture recognition is currently the most common and widely used method for interaction using glove input devices. In order to improve the naturalness of interaction, and thereby decrease the user–interface learning time, there is a need to be able to recognize dynamic gestures.

Dynamic Gesture Recognition (DGR) is difficult for various reasons. The large variations in the speed of execution of various phases of a gesture is one such reason. Another is the quality and positions of the physical properties describing a gesture themselves. These problems are then exaggerated by the differences which arise when various people attempt the same gesture, as well as when the same person attempts repeated executions of the same gesture. Other factors effecting the difficulty of DGR are the emotional state of the person doing the gesture and the accuracy of the input device used. And finally, a large amount of data has to be processed in real time because of large variances in the length of time to execute a gesture.

In this paper we describe our approach to overcoming the difficulties of DGR using neural networks. Backpropagation neural networks have already proven themselves to be appropriate and efficient for posture recognition. However, the extensive amount of data involved in DGR requires a different approach. Because of features such as topology preservation and automatic-learning, Kohonen Feature Maps are particularly suitable for the reduction of the high dimensional data space which is the result of a dynamic gesture, and are thus implemented for this task.

1. INTRODUCTION

Current state–of–the–art graphics workstations allow us to create almost photorealistic images in real time. In addition the multimedia technology enables the integration of sound and video. Virtual reality makes use of these capabilities in order to give the user the sensation of being part of an environment in which (s)he is able to manipulate virtual objects directly. The key issue for the user therefore is to explore virtual environments rather than learning how to do something. To make this easy for both casual and professional users, the actions possible in the artificial environment must be similar to the respective actions in the real world. For example, grabbing an object should resemble the physiological action in reality. In addition users must receive an expectable behavior naturally presented as feedback from the objects in the virtual environment. For example, releasing an object should be followed by the natural fall of that object guided by the laws of the gravitational force.

Graphical 3D scene output technology and the usage of various media including audio (speech, sound and music) and video (still and moving image) are already relatively mature fields. What is lacking are paradigms, techniques and toolkits for well–integrated 3D input styles, and their relationship to the environmental behavior. This is the field where we are mainly concentrating our research. Our goal is to develop user interfaces which require a minimal learning time and a small amount of perceptual power to understand the interaction techniques. See [10] and [11] for discussion about human factors of 3D interaction and [14,21] and [22] for examples of advanced interactions.

2. GESTURE BASED INTERACTION

Gestures are body movements which are used to convey some kind of information from one person to another. Usually this information is easily understood by other humans (at least of the same cultural origin[19]). Although the execution of gestures is not very exact in its nature, in most cases the meaning of them is deterministic. In human-computer-interaction gestures can be used to convey information from the user to the computer system. In this case, however, the information given by the gesture has to be fixed (it should still be possible to make ”more or less” –types of gestures, but the meaning of these gestures must be exactly defined because of the deterministic nature of the computer input).

2.1. Static gestures (postures)

The use of postures is very common in a lot of VR–Systems. Mostly the users wear some kind of glove and can control actions of the VR–System with hand–postures. The pointing posture has been well established for navigation (Fig. 1).

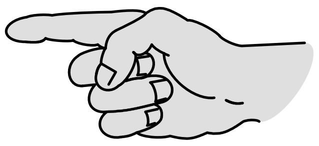


Figure 1: Pointing gesture

Published at SPIE Conference Electronic Imaging Science & Technology, San Jose California, USA, Feb. 1994.

We were able to experience the usability of static gestures with our VR–Toolkit GIVEN (**G**esture–based **I**nteractions in **V**irtual **EN**vironments) and achieved the following results (for more detail, see [20]),

- Hand and arm get tired after a long continuous period of use
- It is difficult to make exact manipulations in the space
- Force feedback (which is missing in our dataglove) is essential for more ”reality” (especially for certain tasks such as grabbing an object)
- The environmental interaction and object manipulation is considered easy and intuitive in most parts
- It helps to have metaphors related to each gesture. For instance, increasing the speed of navigation is related to ”pressing the accelerator pedal” in a car, i.e. the speed increases when the thumb is brought near to the palm.

2.2. Dynamic gestures

Static gestures do not cover the full potential of gesture–based interaction. We assume that for a lot of interactions the naturalness and the intuitivness could be improved when using dynamic gestures. This is supposed to be true at least because the process of interaction is dynamic in both worlds – in the real and in the virtual world. What we understand by dynamic gestures are the movements of the fingers in addition to the trajectory of the hand in a sequence of time steps.

Therefore we focus currently with our research on dynamic gestures. We have to find ”good” translations for actions represented by these gestures. In addition new recognition methods for dynamic gestures have to be developed and investigated. The purpose of this paper is to describe our approach to dynamic gesture recognition.

2.3. Future challenges

We believe that the use of dynamic gestures is one step further in the direction of *more intuitive and more natural* man–machine–interfaces. However, in our opinion the optimal interaction from the human factor point of view should not require any remembrance from the user. The user should know how to perform actions just by his/her (life)experience.

The opening of a door in a virtual environment might be performed as follows:

1) detecting the collision between the cursor e.g. a virtual hand and the door–handle
2) grabbing the door–handle or pushing down the handle
3) either pushing or pulling the door–handle for actually opening the door.

Although this procedure is similar to the real world, from the technical point of view the realization of this relatively simple interaction is much more sophisticated than it seems at first. Two examples may underline the difficulties: first, the collision detection must be very precise in order to realize a collision between hand and door handle which is adequate for grabbing conditions. Second, if the user grabs the door handle and pushes the door, the door cannot follow the movement of the user directly because the door is usually fixed on a wall with two hinges.

The control of such kind of interactions could be simplified if the context of the interaction is taken into account. For instance, if the system knows based on the body movement of the user that he/she wants to open the door the system can react in a ”smarter” way. The body movement can be considered as a dynamic gesture which is performed unconsciously by the user in order to execute an interaction. Therefore the result of dynamic gesture recognition can be used as one input channel for the context information.

Thus, we believe that the research done in the field of dynamic gesture recognition will play a significant role for the advanced interaction construction in the future.

3. FUNDAMENTALS OF THE KOHONEN MAP

The following chapter describes some fundamentals of the Kohonen Feature Map (KFM) which we use for our approach to gesture recognition, a detailed description of the KFM can be found in [15,16].

The KFM is a self-organizing network which is basically trained without supervision. The input patterns are organized in a topological structure. This structure is represented by the weight vectors between neurons of the KFM, where the relationships between different patterns are preserved.

The Kohonen Map is a two-layer network. The first layer receives the input data, the second (competitive) layer is entirely connected to the first layer (Fig. 2). In the second (usually two-dimensional) output layer only one single neuron is active at a time. The spatial distance of two neurons reacting on different input patterns is the measure for the similarity of the two patterns.

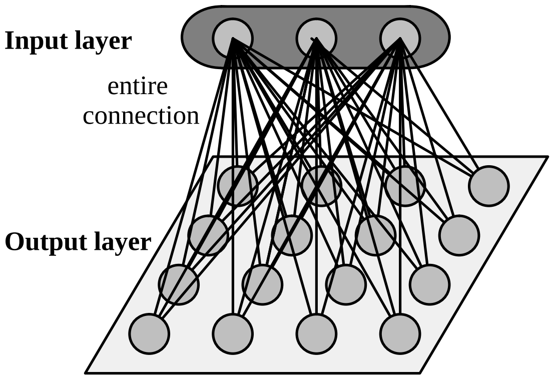


Figure 2: Kohonen Feature Map

The training of the network is done by presenting data vectors to the input layer. Based on the training rules the values of the connection weight vectors of the competitive neurons are changed. The connection weight vectors are initialized with random values. The competitive neuron with the minimal distance to the input vector is then activated. A distance could be defined for instance as the Euclidean distance d_i between the input vector **x** and the connecting weight vector $\mathbf{m_i}$ with

$$d_i = \|\boldsymbol{x}-\boldsymbol{m_i}\| = \sqrt{\sum_{j=1}^{N}(x_j-m_{ij})} \tag{1}$$

while N is the dimension of the data,

$$d_c = \min_i(d_i) \tag{2}$$

The neuron c with the minimal distance d_c will be activated.

The learning process is performed by the modification of the connection weight vectors from a subset of output neurons. After identifying the active output neuron for the current input vector, a modification of the connection weight vectors in the proximity $N_c(t)$ of the active neuron c is performed. The modification of the weights will be done in such a manner that the distance to the input vector decreases (Equation 3).

$$m_{ij}^{(t+1)} = m_{ij}^{(t)} + \alpha^{(t)}(x_j^{(t)}-m_{ij}^{(t)}) \tag{3}$$

Where $\alpha^{(t)}$ represents the time–dependent learning rate.

The size of the proximity $N_c(t)$ is reduced with the training time t. Fig. 3 shows the reduction of the size for an example of rectangle proximity.

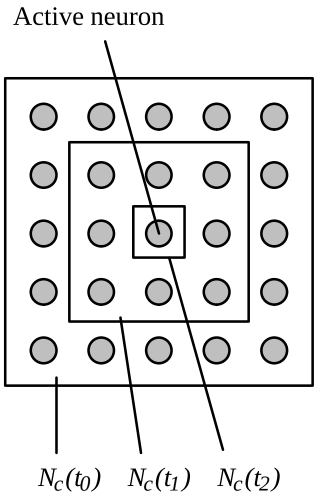


Figure 3: Time–dependant proximity

After the learning process (which can also be called ”clustering”) each neuron in the output layer corresponds to a cluster of the input data space.

In order to use the KFM for classification we have to map a class to each cluster and therefore to each output neuron. A common way for mapping between output neurons and classes is to use recorded training samples as input. Since the classes of these samples are defined, the labeling can be performed by a majority voting of each stimulated neuron.

This can be performed by an interactive selection of training areas and by a majority voting of each stimulated neuron. This process is called *labeling.*

The goal during the organization process was not to find optimal placements of the decision boundaries. Thus error classifications may occur based on this. Proceeding from the self–organization the weight vectors can be modified, according to their class affiliation, in a way that they approach the decision boundaries. This method is called *Learning–Vector–Quantization* (LVQ). In contrast to non–supervised self–organization the weight vectors are not modified according to topological aspects, but the two weight vectors nearest to the input pattern are examined in connection to their class relation each time [12].

4. INPUT PROCESSING

This chapter discusses the general problems concerning the input data for DGR as well as the realized methods to deal with these problems during training, classification and recording of training samples.

4.1. Input data

The major problems of dynamic gesture recognition (DGR) are the unknown data types, the variance and the quality of input data.

4.1.1. Unknown type of input data

The input data for gesture recognition is widely unknown. Since gestures may last several seconds all input data of this time interval has to be considered for DGR. Since the interdependences (correlation) between the input sensors are unknown and an extended examination of the input data space would have been too expensive, all input sensors were used. So the number of data which have to be processed is up to 200 times as high as it is for posture recognition. However only a small part of this data is significant for the recognition of a certain gesture. The distribution of the input data in the high dimensional input data space is unknown.

4.1.2. Variance of input data

At the execution of a gesture there is large variance not only in the duration of the gesture but also relating to the curve described by the hand. For most gestures the direction (especially within the horizontal plane) in which they are executed and the size of the described curves are irrelevant. Since there is usually only a small number of training samples (which are not representative for all feasible types of a certain gesture) those variations very often cause an incorrect recognition of a gesture. It usually is impossible to consider all these variations during the learning phase since the execution of a given gesture depends on the person who executes it as well as their emotional state.

4.1.3. Quality of input data

Another large problem in gesture recognition is the quality of input data, the number of input training samples and whether they are representative for the gestures. With the currently used input devices (VPL–Dataglove, VT–Cyberglove and 6D trackers), input data of two identically executed gestures can differ widely. Moreover in some cases spasmodic changes of the sensor inputs as well as unintentional interdependences among the sensors can be observed (the sensor value of the finger bending changes when the fingers are spread). The input data of two executions of the same gesture may vary widely because of the bad repetition accuracy of the input devices. Another problem is that many people tend to omit the beginnings and endings of gestures, so these parts cannot be used to trigger the recognition.

4.1.4. Preprocessing

An essential improvement of the data quality can be achieved by preprocessing the input data. Disregarding the calibration of the input device, we can distinguish between two different kinds of preprocessing. On the one hand the immediate connecting and processing of the sensor input

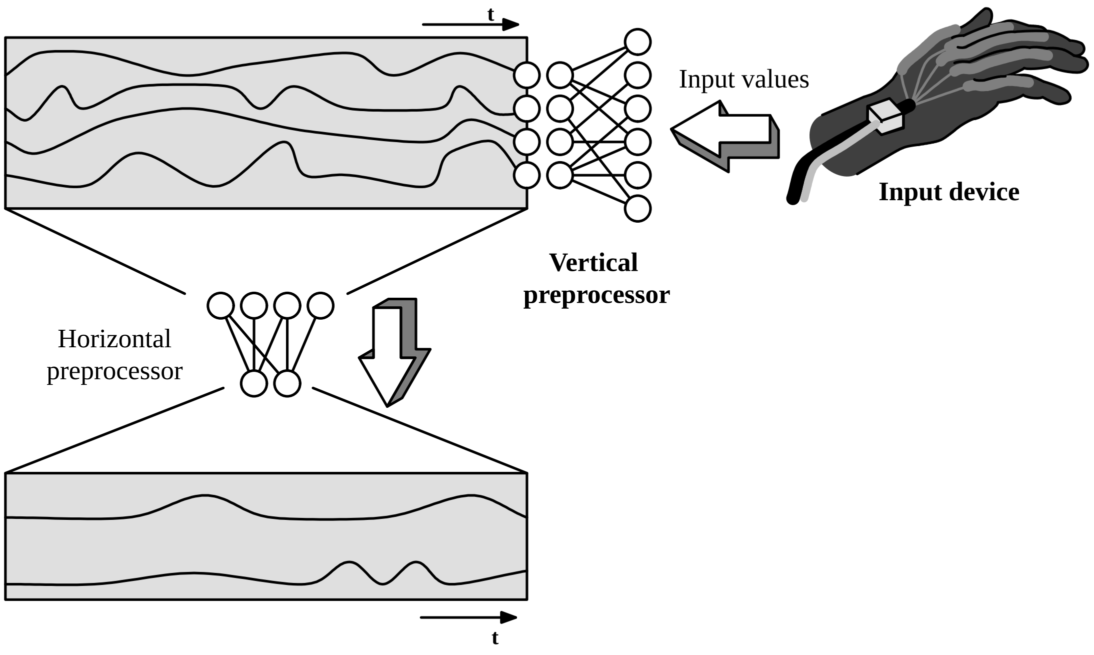


Figure 4: Preprocessing

at each discrete time step (vertical preprocessing) (e.g., scaling, converting radians to degrees), on the other hand the transformation and connection of input data of a certain period, i.e., a certain number of time steps (horizontal preprocessing) (Fig. 4). Examples for horizontal preprocessing are filters (e.g., low pass and high pass) and derived data (e.g., relative instead of absolute data and speed). In order to have a universal and flexible preprocessing mechanism, a transformation pipeline, which can be configured, was developed. Based on a library of transformation algorithms, the input data streams can be connected hierarchically to new data streams.

4.1.5. Recording of training samples

Before starting the training procedure, suitable training samples have to be recorded. As an input device several types of data gloves (see 4.1.3.) where used. The angles of the fingers and the hand as well as the orientation and position of the hand are recorded at discrete time steps. We realized two different methods for recording gestures: training samples may be recorded supervised (by a second person) or they can be recorded automatically. Automatic recording requires definite and well defined starting and ending postures of a gesture to identify the gesture. In order to find the exact triggering point a neural network which already has been used for posture recognition [20] can be applied. However most of the real gestures do not have definite starting and ending postures (especially the ending posture is very often used within the gesture, which leads to a premature suspension of the recording). Users tend to omit these postures or do not perform them very exactly. Independent of the used recording method, the quality of the training samples can be enhanced considerably by additional manual clipping (Fig. 5).

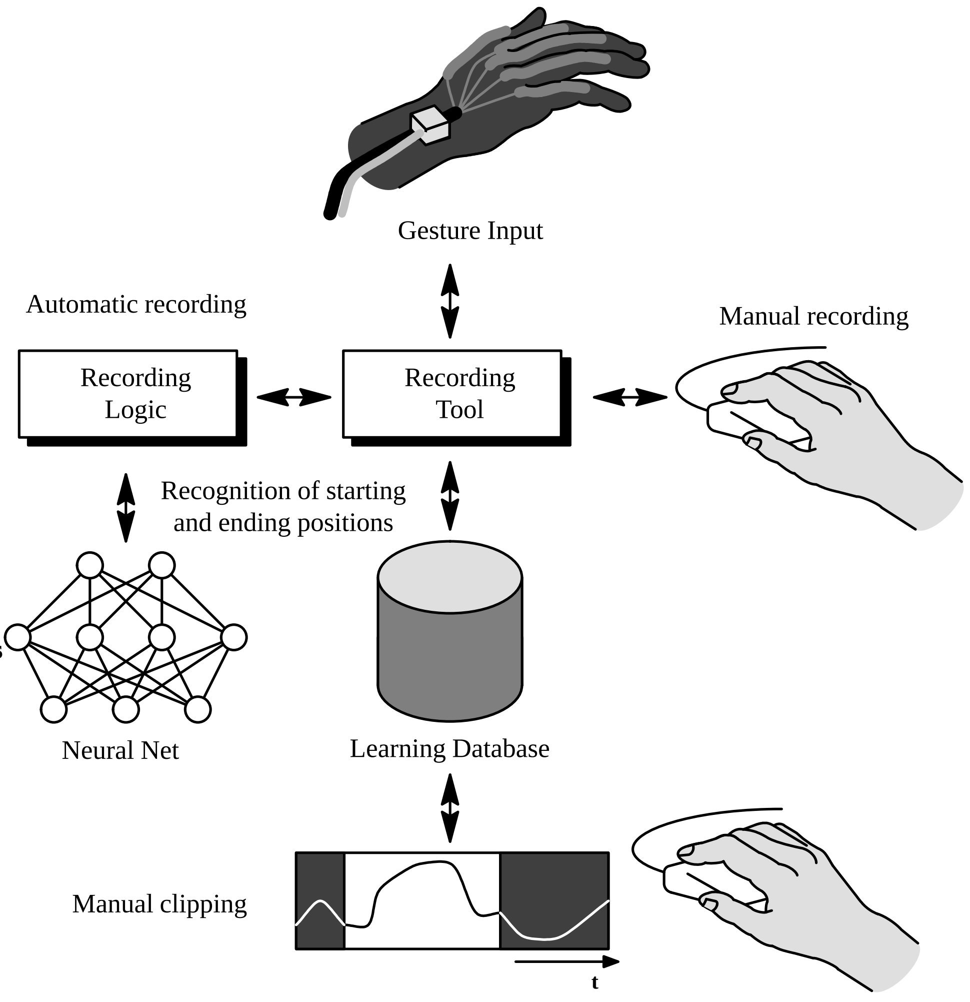


Figure 5: Recording procedure

5. OUR APPROACHES

Since the data space has a very high number of dimensions (one dimension for each input sensor and the time = 30 dimensions) – at which a high correlation rate between some of these dimensions can be suspected – the KFM seems to be the best choice for data reduction. Another important advantage of the KFM is that there is no need for extensive examinations of the data space, because of the unsupervised self–organization mechanism of the KFM. Using a universal implementation of the KFM, which was originally limited to two dimensions in the output layer, it is possible to achieve higher dimension (more complex) projections.

For DGR two different approaches were realized: one approach based on direct mapping by the KFM and another approach based on dividing the gestures into parts.

5.1. Direct mapping approach

5.1.1. Training and recognition

The training of the KFM usually is divided into four phases. During the training the recorded samples are presented to the KFM in random order. Since the input sensors were recorded at discrete points only, the input layer of the KFM represents a matrix of the dimensions:

[number of input sensors] $\times$ [number of time steps] (Fig. 6).

The first training phase serves as the coarse structuring of the KFM, whereas the second phase is used for fine tuning. In these two phases there is no classification, but only a clustering of the input data. The third phase is the labeling of the clusters to the corresponding gesture classes. The fourth phase is used for an optimization of the connection weights with LVQ.

When the KFM is used for recognizing (classifying) gestures, the input data is stored temporally in a buffer that is presented to the KFM. The classification is achieved by referring to the gesture class, which was used for the labeling of the corresponding (active) cluster.

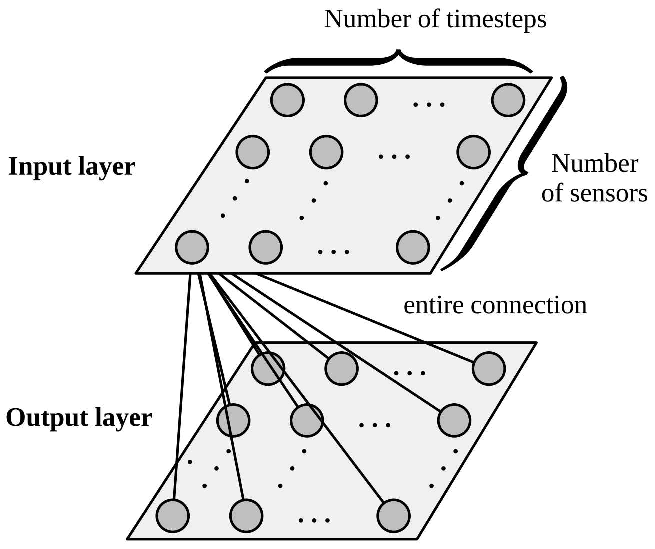


Figure 6: Kohonen Feature Map

5.1.2. The problem of variation in the speed of execution

Using the approach described above the main problem is the large variation in the speed of execution. On the one hand there are very large differences in the length of two different gestures, on the other hand the time of execution of a certain gesture may vary depending on the person. This brings up the question of the suitable (and optimal) size of the buffer (time window). How many input values of each sensor should be presented to the KFM at each time step?

Whereas large buffer sizes lead to an insufficient recognition of short term gestures (since the user usually will do arbitrary movements before and after performing a gesture), short buffer sizes come along with problems for long term gestures, because the gesture is not presented to the KFM at once (only parts of the gesture) (Fig. 7).

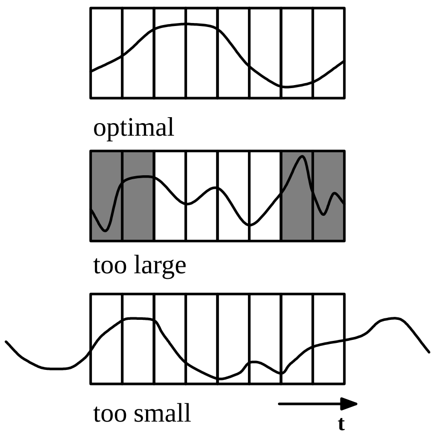


Figure 7: Buffer size

5.1.3. Creating time intervals

To solve these problems and the problem of individual execution speed, it seems to be a good approach to present several sized buffers to the KFM at each time step. This requires a determination of minimal and maximal execution times for all gestures. The maximal duration of a gesture determines the size of the buffer. Starting with the whole buffer, shrinking parts of it are presented to the KFM one after another, always using only the last (most relevant) data of the buffer. The smallest part of the buffer presented to the KFM corresponds to the gesture with the shortest execution time which can still be recognized (Fig. 8).

It is possible to use one KFM for each time interval, in which case the size of the input layer would refer to the length of the interval (length of the buffer part). This can cause recognition problems if the execution time of a gesture differs widely from the execution time of the training samples (only one part of the gesture is presented to the trained KFM and the whole gesture is presented to another KFM, which was not trained on this pattern). Another possibility is standardizing the intervals presented to the KFM to the size of the input layer. This leads to an independence between the number of time steps (length of the interval) and the size of the input layer of the KFM, which allows us to use a smaller input layer from the beginning. Thus we used this alternative in our realization.

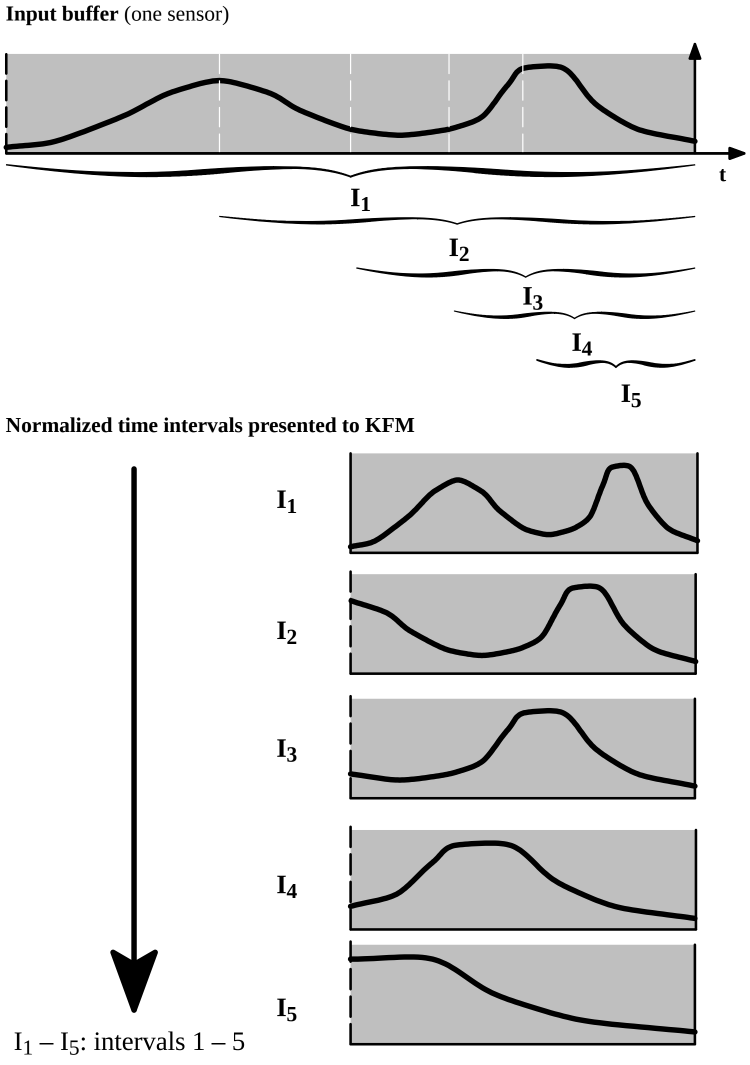
5.1.4. The problem of hyper sensitivity

During the labeling of the KFM clusters there are usually several clusters (especially between the real class areas) which do not belong to any gesture. They are labeled as the refusing class (these clusters were never activated during the labeling of the KFM). During gesture recognition (classification) these clusters (when they become active) are interpreted as arbitrary movements. However the KFM very often recognizes gestures even when the user does not seem to do anything. Especially during short term recognition (short time interval presented to KFM), the KFM is almost constantly recognizing a gesture since the extracted data with a lack of distinguishing details is much closer to one of the trained gestures than it is to one of the patterns of the refusing class.

5.1.5. Suppressing the classification

One possibility to omit hyper sensitivity was already mentioned before: by presenting large intervals to the KFM first, long gestures (with many more details) are recognized before short (and sometimes wrong) gestures. It is possible to improve this approach, so short gestures will not be recognized until it is guaranteed that they are not the beginning of a larger (longer) gesture. It is even possible to ensure the recognition by demanding that no part of a short gesture may be a part of long gesture. The resulting problem is obvious: the recognition of almost every gesture (except that of the buffer length) is delayed for the execution time of the longest gesture (maximal time interval). Even if this time is reduced to a short interval (so only very short gestures can be recognized) there will be a break in interaction coming along with a loss of intuitivness for the user.

Another possibility is the use of threshold values. Threshold values are an appropriate method to avoid hyper sensitivity, although the extensive use of them restricts the universality of recognition. To solve this problem, it is necessary to use one threshold value for each interval length instead of one value for the KFM. The shorter the time interval is, the higher the threshold value should be.



Input buffer (one sensor)
t
I_1
I_2
I_3
I_4
I_5
Normalized time intervals presented to KFM
I_1
I_2
I_3
I_4
I_5
$I_1 – I_5$: intervals 1 – 5

Figure 8: Creating time intervals

The third possibility to avoid hyper sensitivity is to use special training data representing arbitrary movements. This data should be recorded from short and simple movements as well as resting positions. Due to the self–organization of the KFM it is not necessary to consider more than one special class for the classification of this data.

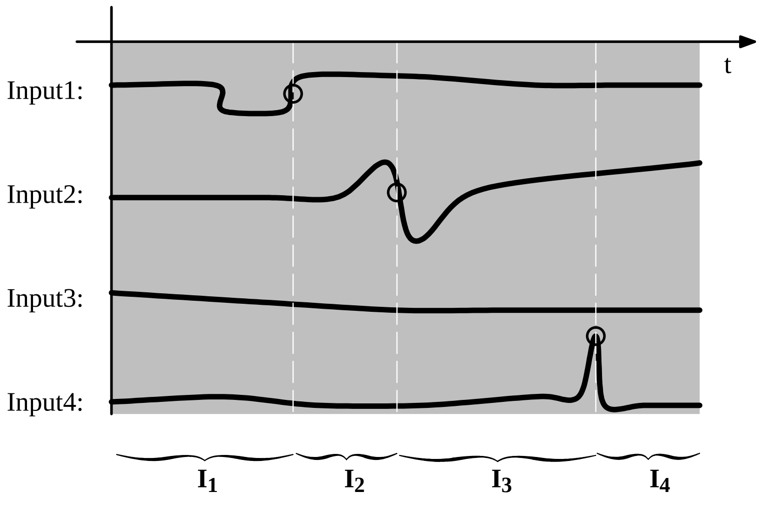
5.2. Dividing into gesture parts

Another approach implemented is based on dividing the gestures into parts. First the gesture parts are recognized and finally they are composed into the whole gesture.

5.2.1. The dividing procedure

In general there are two different methods to obtain gesture parts: On the one hand the gestures that should be recognized have to be analyzed to define a set of gesture parts. All gestures must consist of these parts. Thus it is not possible to train any gestures, which are not (or not completely) composed of these gesture parts. For that the set has to be extended. On the other hand it is possible to generate gesture parts according to a number of (abstract) rules. The simplest example for a rule is the division into equidistant time intervals: for each discrete time step the input data is classified as a posture (this is not a real posture, since it does not make any sense on its own). Since gestures are in general not a sequence of certain postures (see 4.1. and 5.1.2.) this is not suitable for a universal dividing method.

A universal dividing mechanism which can be configured freely and which is based on a large number of dividing criteria (such as minimum, maximum, threshold values, time–out mechanisms, etc.) is used for that (Fig. 9). This mechanism can be applied on any kind of data stream (i.e., input data or data streams resulting from horizontal preprocessing (see 4.1.4.).



I_1 – I_4: intervals 1 – 4

Figure 9: Dividing gesture inputs

5.2.2. Training and clustering

For this approach the KFM is used for recognition of gesture parts only. Since these parts are not known before (i.e., the gesture class of the training data in unknown), the KFM can be used for clustering only. The training consists of two phases, in which the gesture parts of all training gestures are presented in randomized order to the KFM. Our realization allows to present the input data of the gesture part as well as derived data (obtained from the gesture part data by the horizontal preprocessor). The preprocessor already allows a data reduction, so that it is possible to present only some significant values of the gesture part to the KFM.

The presentation of the input data (or derived data) to the KFM results in a sequence of active clusters (output neurons). If the gesture is very short or simple (or if it is a posture) only one cluster will become active.

5.2.3. Combining the gesture parts

For the actual recognition of the gestures it is necessary to combine the gesture parts to one single gesture again. Since one gesture may be represented by different sequences of active clusters, a second neural net seems to be the best choice to combine them.

As input for the second NN the sequence of active clusters of the KFM is used. Each cluster is represented by its index. Since the index numbers of two clusters do not give any information about their similarity (their distance within the output layer of the KFM) a better approach is to use the coordinates of the clusters (according to the dimensions of the output layer) instead.

Since the number of gesture parts (i.e., the length of the cluster sequence) may vary widely (depending on the gesture, its execution and the criterion used for the partition), a special presentation method for the second NN has to be found. One possibility (in analogy to the first approach) is to present sequences of different length to the NN. Another possibility is to use a recurrent NN combined with a ’short’ gesture part window (Fig. 10).

Although the recognition method used for this approach tends less to hyper sensitivity, the methods described above (see 5.1.5.) to avoid this (late recognition, threshold values and training of arbitrary movements) improve the recognition quality as well.

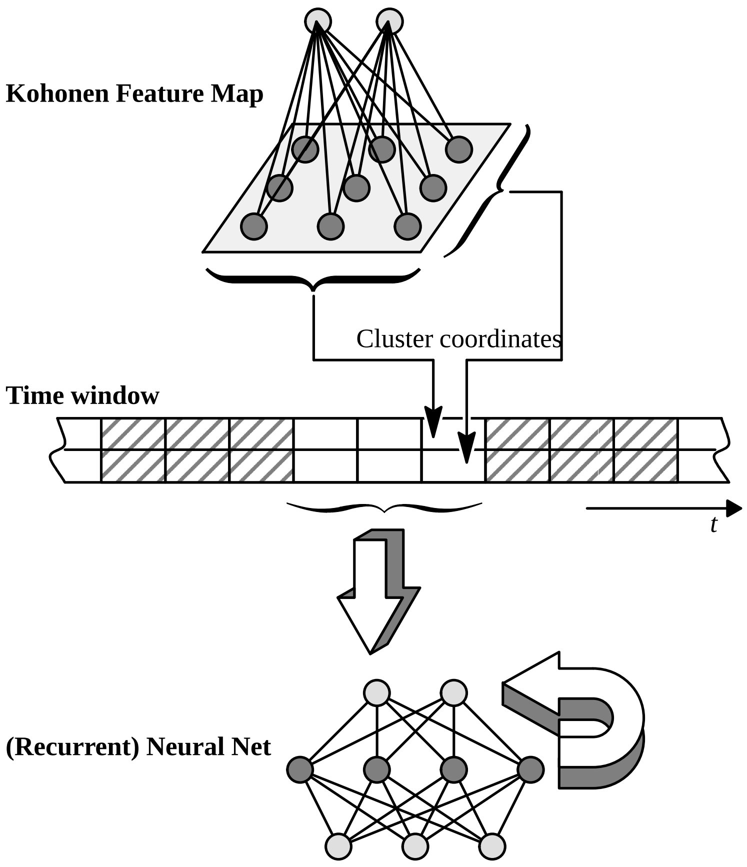


Figure 10: Combing gesture parts

5.3. Results

Realizing these two approaches we got the following results:

Both approaches allow real gesture recognition and have to deal with the same basic problems, such as insufficient training and input data. For real (universal) gesture recognition the number of training samples as well as the size (and dimensions) of the NN have to be extended. At the moment this is a problem of machine power, since the DGR has to be performed in real time to guarantee a natural (intuitive) interaction.

6. RESULTS

The two approaches discussed in Chapter 5 are implemented in the system *Gesture Tool* (for implementation details see [5]). The results we achieved will be presented in this chapter.

With this implementation we have shown that both approaches work for dynamic gesture recognition. The first approach is more general and therefore would also be adequate for classification and recognition of other time dependent data.The second approach is more specific for gesture recognition. For this approach it must be possible to extract break points for gesture parts from the input data. The extraction is based on criteria such as extreme values which have to be defined in advance.

In the first approach there is no data reduction during the preprocessing phase. Therefore a great amount of data is passed to a large input layer (e.g., 20 input sensors, sampling rate 30 Hz, maximum gesture length 3 s results in $20 \times (30 \times 3) = 1800$ inputs). The number of distinguishable classes depends highly on the size and dimension of the output layer. The experience showed us that networks configured as in the example above and even with a 2D–output layer (8×8) did not perform in real time an a workstation (SUN Sparc 10). A configuration which was suitable for 10 gestures was achieved by reducing the sampling rate to 10Hz and the number of input sensors to 15.

Due to the more specific nature of the second approach, it is from the performance point of view better than the first. The reason for this is the division of the input data stream into small intervals. This requires only a small input layer of the KFM, which can be processed in real time. This enables us to use a 3D–output layer which is able to represent more complex topologies. Since the layers of the second NN can be very small (dimensions of the KFM output layer $\times$ max. number of gesture parts or width of the time window ≤ 30 for the input layer and number of gestures for the output layer) its performance is not relevant to the recognition time.

However, for using the recognition output of our system *Gesture Tool* as an input to a VR–System we need to improve the speed of the implementations, especially in the first approach. Although the recognition is done in real-time, there is still a remarkable lag between execution of gestures and recognition. In fact the highly dimensioned networks which are appropriate for this problem need a remarkable amount of hardware power.

The main advantage of our approaches compared to others is that we do not make any limitation concerning the gestures. Most other approaches require a well defined start and end posture. In order to avoid these limitations we need either a sophisticated preprocessing technique or huge networks.

7. RELATED WORK

Since the invention of the dataglove people have been trying to use it to recognize postures and gestures. These other projects could generally be classified into two groups: those which incorporate neural nets and those which do not.

An example of a non-neural-net approach was the work done by M. Bordegoni at the Rutherford Appleton Laboratory[3]. In this system certain postures would be denoted as the start and end of a gesture. The teaching phase consisted of creating and storing a range of sensor values at certain time intervals over the ”life” of the gesture; since the gesture was repeated many times, the ranges of sensor values at the time intervals would acquire a minimum and maximum. Then during the recognition phase the system would continually try to match the current posture to any of its gestures' start postures and, after one was found, would keep checking the sensor data at the time intervals to make sure it all stayed within the range expected at that relative interval.

This led to a simple, fast gesture recognition system, which was rather successful in the laboratory. The disadvantage however is that the beginning and end postures had to be rigorously set and remembered by the users: if the start posture was not correct the gesture was never started, and if the end posture was not executed correctly, the whole gesture was lost.

Under the heading of neural network approaches falls the work done by M. Ala-Rantala[1]. This system was originally set up as a neural network for static posture recognition, and was very successful in this respect. In order to apply the system to gesture recognition, the number of input neurons was increased by **n** so that the same amount of information used to recognize postures could be stored over a continually-shifting ”time window” of **n** intervals; this information was then fed into the neural network.

While it enjoyed limited success, this approach led quickly to some problems. Because of the lack of sophisticated preprocessing, recognition of the gesture is very dependent on its execution time: if the same gesture is done more quickly there is less of a chance that it will be recognized.

An example of a project ranging into the very complex neural network area was the work done by K. Murakami and H. Taguchi[18]. This approach used recurrent neural networks, feeding the ”hidden layer” back into itself as input neurons in order to retain a longer history of sensor data. One may also wish to look into the project done by F. Camplani at the University of Mailand in Italy[7] for an example of a yet more complex approach.

8. FUTURE WORK

Our future work will be directed at the improvement of the speed of recognition in order to reduce the gap between execution of gestures and recognition. The success of the recognition in both approaches depends basically on the preprocessing method used, the network configuration and the classification parameters, such as threshold values and maximal gesture length. Therefore research has to be conducted in order to find an optimal or at least acceptable solution. For the evaluation of the system a user test has to be performed.

In the future we also want to investigate dynamic gestures from the human factors point of view. Therefore we have to find meaningful translations for gestures to their functionality. This investigation will be performed by connecting the developed *Gesture Tool* to our VR-Toolkit GIVEN[2].

9. ACKNOWLEDGEMENTS

The authors would like to thank Professor Dr.–Ing. José Encarnação for his encouragement and support in making this work possible. We would also like to thank Dr. Markus Groß and Frank Seibert for their insight and comments on classification methods and Neural Networks and we thank Dr. Markus Gross for his support in the process of writing this paper. Furthermore, we thank Constance Belz and Werner John for proof reading and commenting on this paper.

10. REFERENCES

1. Ala-Rantala,M., ”NeuroGlove: An Interactive Test Environment for Hand Posture and Gesture Recognition Using Dataglove and Neural Networks,” (unpublished) *Master’s Thesis*, Technical University Darmstadt, Darmstadt: 1992.

2. Böhm,K., Hübner,W., Väänänen,K., ”GIVEN: Gesture Driven Interactions in Virtual Environments – A Toolkit Approach to 3D Interactions,” *Interfaces to Real and Virtual Worlds*, pp. 243–254, Montpellier, France, March 1992.

3. Bordegoni,M., *Dynamic Gesture Machine*, Rutherford Appleton Laboratory, Science and Engineering Research Council, 1992.

4. Bordegoni,M., Hemmje,M., ”A Dynamic Gesture Language and Graphical Feedback for Interaction in a 3D User Interface,” in *COMPUTER GRAPHICS forum*, Hubbold,R.J., Juan,R., Volume 12, No. 3, Blackwell Publishers, 1993.

5. Broll,W., ”Konzeption und Entwicklung eines Werkzeuges zur Analyse und Klassifikation dynamischer Gebärden unter Verwendung Neuronaler Netze,” *Master’s Thesis*, Technical University Darmstadt, Darmstadt: 1993.

6. Busch,Ch., Groß,M., ”Interactive Neural Network Texture Analysis and Visualization for Surface Reconstruction in Medical Imaging,” in *COMPUTER GRAPHICS forum*, Hubbold,R.J., Juan,R., Volume 12, No. 3, Blackwell Publishers, 1993.

7. Camplani,F., ”Gesture Recognition with neural nets and DataGlove,” *Master’s Thesis*, Dipartimento di Sciennze dell ’Informazione Universita’ Statale degli Studi di Milano, Milano: 1992.

8. Encarnação,J.L., Eckardt,D., ”Entwicklungstendenzen in der Mensch–Maschine–Kommunikation,” *Meeting of INFINA’89*, 1989.

9. Fels,S.S., Hinton,G.E., ”Building Adaptive Interfaces with Neural Networks: The Glove–Talk Pilot Study,” *INTERACT’90 Proceedings*, pp. 683–688, Elsevier, 1990.

10. Foley,J.D., Wallace,V.L., Chan,P., ”The Human Factors of Computer Graphics Interactive Techniques,” *IEEE Computer Graphics and Applications*, November 1984.

11. Foley,J.D., ”Interfaces for Advanced Computing,” Scientific American, pp. 127–135, October 1987.

12. Groß M., Seibert F., ”Neural Network Image Analysis for Environmental Protection”, in Informatik aktuell *Visualisierung von Umweltdaten 1991*, pp. 31–42, Springer Publishing Company, Berlin, 1992.

13. Groß, M., ”Physiological Aspects of Human Vision and Computer Graphics,” *Tutorial for Eurographics’91*.

14. Holloway,R., Fuchs,H., Robinett,W., ”Virtual–Worlds Research at the University of North Carolina at Chapel Hills as of February 1992”.

15. Kohonen, T.**,** *Self–Organization and Associative Memory,* 3. Aufl. Berlin: Springer–Verlag, 1989.

16. Kohonen, T., ”The Self–Organizing Map,” *Proceedings of the IEEE.* Volume 78, No. 9 (September 1990): 1464 – 1480.

17. Lippmann, R. P**.,** ”An Introduction to Computing with Neural Nets,” *IEEE ASSP Magazine*, pp. 4–22, April 1987.

18. Murakami, K., Taguchi, H., ”Gesture Recognition using Recurrent Neural Network,” *Proceedings of the ACM SIGCHI,* Human Interface Laboratory, Fujitsy Laboratories LTD, Kawasaki, 1991.

19. Rammel,G., ”Gebärde – Gebärdensprache,” *Die Gebärdensprache: Versuch einer Wesensanalyse,* Carl Marhold Verlagsbuchhandlung, Berlin, 1974.

20. Väänänen,K., Böhm,K., ”Gesture Driven Interaction as a Human Factor in Virtual Environments – An Approach with Neural Networks,” in *Virtual Reality Systems***,** Earnshaw R., Gigante M., Jones H., pp. 93–106, Academic Press, 1993.

21. Venolia,D., ”Facile 3D Direct Manipulation,” *INTERCHI ’93*.

22. Zeleznik,R.C., Herndon,K.P., Robbins,D.C., Huang,N., Meyer,T., Parker,N., Hughes,J.F., ”An Interactive 3D Toolkit for Constructing 3D Widgets,” *Computer Graphics Proceedings*, Annual Conference Series, 1993.