Electronic Notes in Theoretical Computer Science 43 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume43.html

# Formal Verification in the Design of Gestural Interaction

G.J. Doherty [b,1] G. Faconti [a,1] M. Massink [a,1]

[a] *C.N.R.-Ist. CNUCE, Via Moruzzi 1, I56100, Pisa, Italy*

[b] *CLRC, Rutherford Appleton Laboratory, Oxfordshire, U.K.*

**Abstract**

This paper shows how formal modelling can be used in the design of a dynamic gesture language defined by sequences of poses. It discusses two models at different levels of abstraction dealing with important usability issues of the language such as ambiguity and overlap in the recognition of gestures. An approach to make the language more resilient to intermediate poses is evaluated based on a timed extension of the model. A tool providing model checking for hybrid automata is used to perform systematic and automatic analysis.

## 1 Introduction

Many 3D CAD and Virtual Reality applications require user interfaces that embody concepts like spatial depth to facilitate navigation and manipulation of objects in three dimensional virtual space. Traditional two dimensional input devices do not directly support these kinds of interaction. Three dimensional input devices have been developed to better fit these requirements, such as flying mice, the space ball and data gloves. More recently, computer vision based techniques have been developed to extract 3D information from interpretation of images acquired by means of digital cameras [4].

For these kinds of applications, we would like the human operator to be able to concentrate on the task in hand rather than on an intermediate input device. This requires that the interaction be perceived as continuous, and that it operates in accordance with the expectations of the user. An application which aims to achieve this by means of a 3D gestural interaction language is described in [3]. However, the design of a gesture language poses several interesting usability issues. Some of these issues have been discussed in [7] where an analysis of the gesture language is made based on a formalisation

[1] Partially supported by the TACIT research network under the European Union TMR programme, contract ERB FMRX CT97 0133.

in Modal Action Logic of a part of the cognitive theory Interacting Cognitive Subsystems (ICS) [1]. The analysis examins how delay in feedback, similarity between gestures and uncertainty about the state of the system may confuse the user.

In this paper, we discuss the gesture language, as described in [2,3], for interaction in 3D virtual space. However, the form of analysis we present can be applied to a more general class of issues, and is also independent of the gesture recognition technology used. In fact, the problems encountered in the analysis of the system can be dealt with at a level of abstraction that effectively hides the details of any specific technology that might be employed for the interaction.

The interface proposed in [3] allows the user to perform gestures by means of sequences of both static poses, performed during dynamic hand location, and dynamic hand poses, performed with static hand location. Hence, gestures are either of class SPDL (static hand posture, dynamic hand location) or of class DPSL (dynamic hand posture, static hand location) according to the classification described in [9].

The design of a gesture language forms an interesting example for the application of formal modelling because of its limited (but easy to extend) complexity and new aspects. We do not intend to criticise the language proposed in [3], but rather show an approach that may be of help in this area of interface design. There are a number of issues that have to be dealt with in the design of a proper dynamic gesture language. Some of them are purely language related issues, others have their roots in human factors.

**Language concerns:-**

- **Ambiguity.** One of the problems of the gesture language is that when using only informal reasoning it is not so easy to make sure that there is no ambiguity in gesture recognition. In [7] it has been shown that in one of the proposed gesture languages [3] such an ambiguity exits, i.e. there are possible series of postures that may lead to the recognition of more than one gesture at a time. Detecting ambiguities may not be that easy in general, however. A formal model and automatic verification tools may be helpful in finding critical situations.
- **Overlap.** Another problem is that there might be some overlap between the gestures. This could lead to the partial recognition of one gesture during the recognition of another gesture. This way the recognition of a gesture following another may occur much sooner than expected by the user, leading to confusion.

**Human performance concerns:-**

- **Resilience.** In general, humans are unable to repeat a posture or gesture in exactly the same way. To deal with this phenomenon the recognition process allows a certain freedom in the way the user can perform the

posture. It is clear that when there is little freedom, relatively many poses would be interpreted as an undefined posture. The consequence is that it is hard for the user to perform the series of poses required for the intended functionality. On the other hand, when there is much freedom, the probability for partially overlapping postures increases, leading to other problems in the recognition.

- **Accurate and timely feedback.** Accurate, clear and timely feedback plays an important role in user interfaces. Especially when there is a tight coupling between user and computing system, such as in interaction based on gesture recognition. As has been shown in [7], delay in feedback on the portion of a gesture being recognised can lead to confusion and oscillating behaviour. Also confusion about the mode in which the computing system is operating can occur when the reaction of a computing system appears not to be in line with user's expectations.
- **Accidental slips of action.** Some classes of postures and gestures may be more prone than others to be badly formed or misrecognised. It should be made sure that such accidents do not lead to serious consequences.

This multitude of factors can make it rather hard to develop gesture languages that are a pleasure to use. Moreover, often the usability of a gesture based interface is validated a posteriori, i.e. after all implementation work has been done and a prototype is available. Finding the exact cause of usability problems at that stage may turn out to be very difficult. First of all because there are so many factors that may have contributed to the problem. Secondly, because statistical approaches are used to recognise poses and gestures and therefore repeated experiments may produce different results for every test. Thirdly because there is a natural variation in human performance.

The above problem could be handled better if a way could be found to analyse the factors separately. One way in which this could be done is to perform analysis on models of the gesture interface, focusing on different aspects in isolation. Two of these aspects are ambiguity and overlap. This is the part we will focus on in this paper.

The modelling technique we use is that of (hybrid) automata using varying degrees of the capabilities of the model checking tool HyTech [11,12]. This technique allows us to describe a relatively simple automata model of the gesture recognition process and perform automatic and systematic verification of ambiguities and overlap in the gesture language. We show how the model can be helpful for the improvement of the gesture language. As a next step we show how a timed extension of the model can be used to examine the effect on the recognition performance of the system when allowing for intermediate, non-specified, postures between specified postures of a gesture.

This paper discusses only the first preliminary models of gestural interaction. We intend to develop extended models which can take into account the statistical performance of both users and gesture recognition systems and also timed aspects of feedback. Analysis of these models may give us useful clues

about problems in the usability of the system. Models of this kind will require timed and stochastic modelling techniques that have recently become available as a result of research in protocol verification. The current work can be seen as a necessary first step towards the final goal of more comprehensive models of gestural interaction that can provide information about usability and performance in the design phase of an interface, so before the construction of a complete prototype.

In the next section we describe in more detail the gesture language as originally proposed. Section 3 shows an automaton model and analysis of a subset of the language illustrating the detection of ambiguity and overlap by means of reachability analysis. Section 4 discusses improvements to the language and gives a model and analysis of the complete language. In Section 5 and 6 human factors related issues and further research are discussed. Section 7 draws a number of conclusions.

## 2 The dynamic gesture language

In [3] a gesture language is described for navigation and manipulation of virtual objects in a 3D virtual space. The interface makes use of a dataglove [15] that registers the position of finger joints at a rate of about 60Hz. The trajectory of the hand is derived by interpolating over time the location of the centers of mass of each captured pose in a sequence. It should be noted that the rate of input of glove data is particularly demanding for computer resources if gesture recognition has to be done in real-time, largely exceeding the rates currently affordable with standard equipment using computer vision techniques, which generally do not exceed 14 frames per second.

The application subject to analysis in this paper allows the definition of a gesture language in an initialisation phase separate from the actual gesture recognition. In this phase the user "shows" the system the different poses that appear in the language. The characteristic features of the individual postures are captured and memorised. Subsequently, the captured poses may be composed into sequences, each one defining one gesture. After this initialisation the application is ready for gestural interaction using the defined language.

Typical features that are used to characterise postures are the orientation of the hand and bending values of finger joints. For some of the dynamic poses, the trajectory of the movement is also specified. When the gesture language has been specified, recognition is performed by a Gesture Recognition Machine. Its algorithm operates by means of a number of parallel processes, each one specialised for recognising one of the gestures, that independently use the incoming data. When a gesture is recognised by one of the processes, this is notified to the application so that the system can react appropriately.

### *2.1 The language*

The proposed language consists of gestures for navigation, selecting and querying an object, zooming, gripping, including rotation and moving of an object, and a gesture indicating to exit from the application. The following description is taken from [3].

**Navigation.** The application starts performing a navigation task when the "Index" posture is recognised. A rotation of the hand changes the point of view of the 3D scene. When the pose is released, the navigation task ends. The "Index" pose consists of a fist with only the index finger stretched.

**Picking.** During navigation, when an object, or a set of objects, are reached, they can be selected by performing the posture "Pistol". The "Pistol" pose is formed by a fist with the index finger and the thumb stretched and with the thumb in the upward direction.

**Querying.** Also this task can be performed during navigation. When an object is reached, its content can be visited by performing a "Qmark" posture, which is also the final posture of the gesture. The "Qmark" pose is similar to the index pose, but with the index finger bent halfway.

**Zooming.** The gesture starts by a "Flat" pose performed with the back of the hand turned towards the user. Zooming in occurs when the hand is moved away from the user, zooming out when the hand is moved towards the user. The task is ended when any other gesture is recognised.

**Gripping.** This gesture starts when a "Fist" posture is recognised. The object is grabbed and can be rotated or moved by means of rotation or moving of the closed hand. When the "Fist" pose is no longer recognised, the gripping task ends.

**Exit.** The exit gesture is similar to an Italian good-bye wave. This consists of opening and closing the hand, with the palm to the right of the user, repeated twice.

Fig. 1 gives a graphical representation of three of the gestures. Zooming is presented at the first line of images, grasping at the second and a shorter version of exit at the last line. The "Any" pose indicates the end of a gesture and may coincide with the first pose of another gesture. A summary of the characteristics of the gestures is given in Table 2.1.

### *2.2 Feedback*

A graphical image of the user's hand is shown as a natural feedback to the movements of the user's "real" hand. It reflects both position and shape of the hand. The graphical hand changes colour or shape when a gesture is recognised. Specific colours and shapes may be associated with each task.

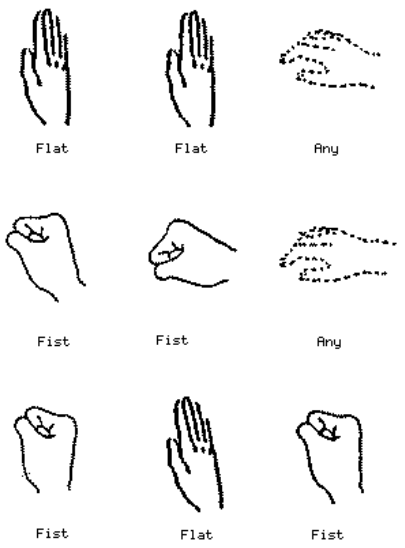


Fig. 1. Some gestures of the gesture language; zoom, grip and exit.

### *2.3 Gesture Recognition*

Gesture recognition takes place by matching the incoming data from the data glove with the specified models of the postures and gestures. Each gesture has an associated process that tries to match the data with the model. When the data can be matched with the first posture, the process is becoming "active" and tries to recognise the remaining postures. When a sufficient number of postures are recognised, the related task can be activated until the data from the glove match the final posture of the gesture. The processes keep track of the history of the gesture and update a pointer to the currently expected posture. A parameter sets the maximal number of consecutive mismatched postures that is considered to be acceptable.

## 3 Ambiguity and Overlap

The first model we will consider models a subset of the gesture language. We model the *zoom*, the *grip* and the *exit* gesture recognition processes as separate automata using a graphical version of the language HyTech. For details on the syntax and semantics of this language and its associated tool for reachability checking we refer to [11,12]. In this section we will explain the semantics informally while developing the models for the gesture language.

All three considered gestures consist of series of hand-poses of two kinds; flat and fist. All other hand-poses are recognised as being of the kind *other*. This simple model allows us to formalise the problem of ambiguity and overlap that may occur in the gesture language and how these problems can be detected automatically by means of model checking techniques.

| Gesture | Posture Model | Orientation | Trajectory |
|---|---|---|---|
| Navigation | Index; | Any | Any |
| | Any | Any | Any |
| Picking | Index; | Any | Any |
| | Pistol; | Any | Any |
| Querying | Index; | Any | Any |
| | Qmark | Any | Any |
| Zoom in | Flat; | BT, FU | To-Usr |
| | Any | Any | Any |
| Zoom out | Flat; | BT, FU | From-Usr |
| | Any | Any | Any |
| Gripping | Fist; | Any | Any |
| | Any | Any | Any |
| Exit | Flat; | BR, FU | Any |
| | Fist; | BR, FU | Any |
| | Flat; | BR, FU | Any |
| | Fist | BR, FU | Any |

Table 1
Summary of gesture language specification

### *3.1 Automata model*

The zoom gesture starts when a flat-pose is recognised. This pose can last as long as needed, during which the zoom remains activated. Zooming ends when a fist or another pose is recognised. If we assume that the poses are generated by a human wearing a data-glove that produces raw data about the position of all finger joints at a regular rate, say 60Hz, and the pose-recognition is performed in real-time without significant delay, we can model pose-generation as a series of flat, fist and other poses at a regular rate. Random series of this kind can be modelled as the behaviour of the automaton *Gesture* in Fig. 2. This automaton has only one location, that is also its starting location, and three transitions. The transitions are labeled by the names of the postures and guards and assignments to a clock variable $t$. Clocks in HyTech increment automatically in a continuous way. So every time the variable $t$ reaches the value 1 the automaton can perform any of the three transitions non-deterministically, generating a fist, flat or other posture. When a transi-

tion is performed, the clock is reset to 0 and the automaton starts from its initial location. The new value of $t$ after the transition is indicated by $t'$.

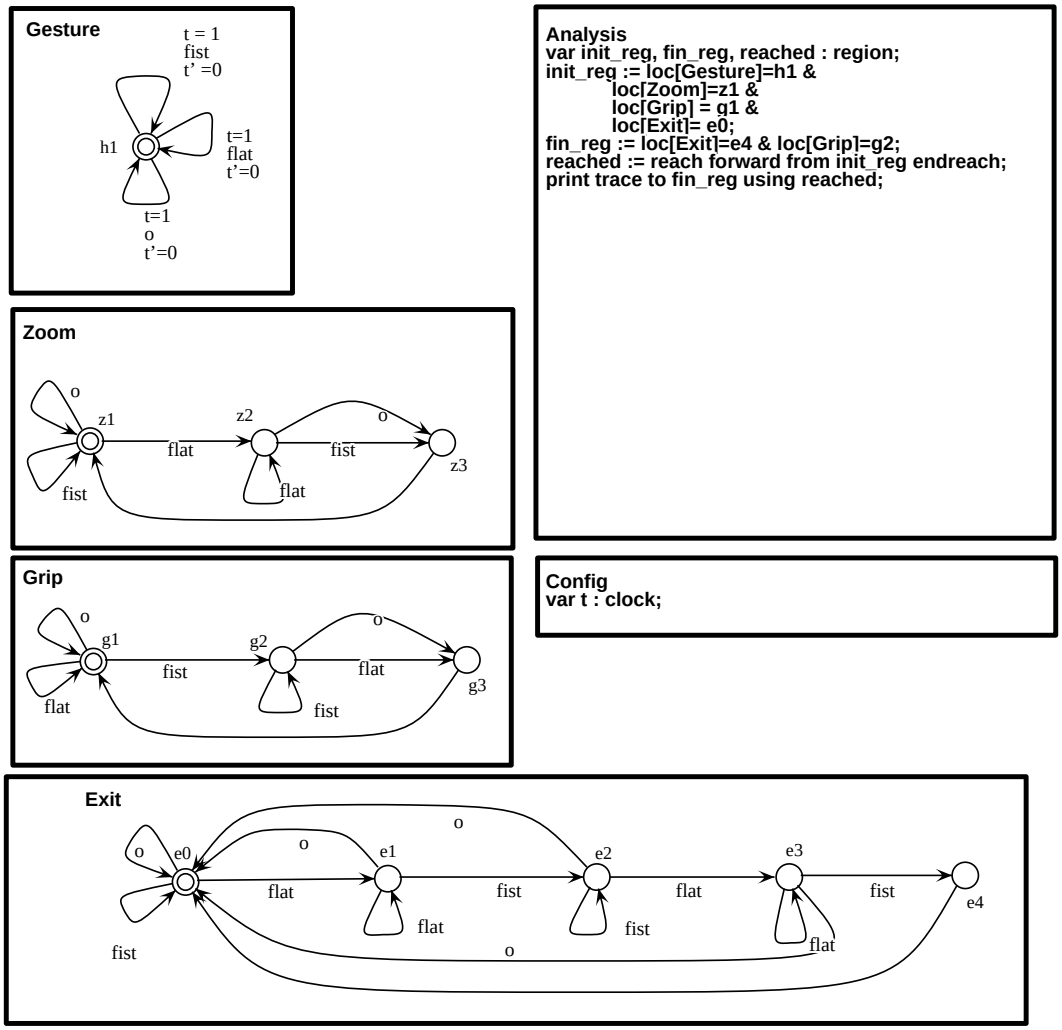


Fig. 2. Gesture recognition

A model of the zoom gesture is given by the automaton *Zoom* in Fig. 2. This automaton has three locations and *z1* is the initial location. *Zoom* can accept many fist and other postures in the initial location, without becoming active. However, when a flat posture is recognised, it goes to the next location and zooming becomes active. Zoom remains active as long as it stays in location *z2*, i.e. as long as flat poses are recognised. The recognition of a pose different from flat (so fist or other) results in the disactivation of zooming and a return to the initial location of the automaton.

The grip gesture starts when a fist pose is recognised and is active until a flat or other pose is recognised. The automata model is presented in Fig. 2 as automaton *Grip*.

The gesture for exiting from the gesture recognition is a bit more complex. In the literature on the gesture language two different versions can be found.

One being a series of three poses fist, flat, fist, as illustrated in Fig. 1, the other a series of 4 poses flat, fist, flat, fist. The latter is modelled by the automaton *Exit* in Fig. 2.

### *3.2 Reachability Analysis*

#### *3.2.1 Ambiguity*

This simple model allows us to perform a first analysis with the tool HyTech. Since the model of the pose-generator allows for all possible series of poses consisting of fist, flat and other poses, we can automatically check whether in the current model it is possible that two gestures are recognised at the same time.

Zooming is activated when zoom is in location `z2`, grip is activated when in location `g2` and exit is performed when the exit automaton reaches `e4`. Checking for ambiguity results therefore in checking whether it is possible to reach a state in which:

- zoom is in `z2` and grip in `g2`, or
- zoom is in `z2` and exit is in `e4`, or
- grip is in `g2` and exit is in `e4`

This analysis may be performed by forward reachability analysis, starting from a state in which each automaton is in its initial location, and checking whether a state can be reached in which two gestures are recognised to be characterised by the above listed combinations. The analysis of the third combination in the list above is formulated as follows in the HyTech analysis language:

```
Analysis
var init_reg, fin_reg, reached : region;
init_reg := loc[Gesture]=h1 &
            loc[Zoom]=z1 &
            loc[Grip] = g1 &
            loc[Exit]= e0 &
            t=0;
fin_reg := loc[Exit]=e4 & loc[Grip]=g2;
reached := reach forward from init_reg endreach;
print trace to fin_reg using reached;
```

The analysis generates an example trace that shows that it is indeed possible that Exit and Grip are recognised simultaneously. Table 2 shows which (shortest) combination of user gestures can lead to this situation. The table lists the time in the first column, the vector of the location of each automaton at the given time, the value of clock $t$ and the action that has been performed to reach the next situation in the last column. Note that a next situation can also be reached by letting some time pass or by an internal transition of one automaton.

In the last line (10) of the table we can see that the state $h1.z3.g2.e4$ has been reached, indicating that *Grip* is in location $g2$ and *Exit* in $e4$. Notice also that in the trace *Zoom* has been going twice through its location $z2$, indicating that the Zoom gesture has been recognised twice. The same holds for the Grip gesture.

| Step | Time | Loc | Value t | Via |
|---|---|---|---|---|
| 0 | 0.00 | h1.z1.g1.e0 | 0 | 1.00 time units |
| 1 | 1.00 | h1.z1.g1.e0 | 1 | flat |
| 2 | 1.00 | h1.z2.g1.e1 | 0 | 1.00 time units |
| 3 | 2.00 | h1.z2.g1.e1 | 1 | fist |
| 4 | 2.00 | h1.z3.g2.e2 | 0 | internal |
| 5 | 2.00 | h1.z1.g2.e2 | 0 | 1.00 time units |
| 6 | 3.00 | h1.z1.g2.e2 | 1 | flat |
| 7 | 3.00 | h1.z2.g3.e3 | 0 | internal |
| 8 | 3.00 | h1.z2.g1.e3 | 0 | 1.00 time units |
| 9 | 4.00 | h1.z2.g1.e3 | 1 | fist |
| 10 | 4.00 | h1.z3.g2.e4 | 0 | end |

Table 2
A trace to the simultaneously activated Grip and Exit operation

### *3.2.2 Overlap*

Another problem may occur when gestures have partially overlapping series of poses. In that case one gesture may get recognised "too soon", due to the fact that part of it has been recognised during the time that another gesture was active. This problem is a bit harder to formalise. At least we would like that when one gesture is finishing or active all the others are at their initial location.

This needs to be checked for each gesture separately. For example, it can be verified whether whenever *Zoom* is in location `z3` the other processes, grip and exit, can be in a location different from their initial ones `g1` and `e0`. The HyTech reachability analysis expression below checks whether zooming can be activated while both the Grip and the Exit gestures are on their way of being recognised at the same time, i.e. *Zoom* is in `z3` and *Grip* or *Exit* are not in their initial locations.

```
Analysis
var init_reg, fin_reg, reached : region;
init_reg := loc[Gesture]=h1 &
```

```
            loc[Zoom]=z1 &
            loc[Grip] = g1 &
            loc[Exit]= e0 &
            t=0 ;
fin_reg := loc[Zoom]=z3 & (~loc[Grip]=g1 |
           ~loc[Exit]=e0);
reached := reach forward from init_reg endreach;
print trace to fin_reg using reached;
```

The short trace in Table 3 shows that the above problem can easily occur. The trace shows that when *Zoom* is in location $z3$ both the Grip and Exit gesture are partially recognised. For the Grip gesture this is not a real problem because the Zoom operation ends implicitly when a flat pose is no longer recognised. However, for the Exit gesture this may lead to an unexpected recognition when the user thinks they are performing a series of Zoom and Grip gestures. Note that absence of ambiguity alone is not sufficient to guarantee absence of overlap and unexpected recognition of a gesture.

| Step | Time | Loc | Value t | Via |
|---|---|---|---|---|
| 0 | 0.00 | h1.z1.g1.e0 | 0 | 1.00 time units |
| 1 | 1.00 | h1.z1.g1.e0 | 1 | flat |
| 2 | 1.00 | h1.z2.g1.e1 | 0 | 1.00 time units |
| 3 | 2.00 | h1.z2.g1.e1 | 1 | fist |
| 4 | 2.00 | h1.z3.g2.e2 | 0 | end trace |

Table 3
Overlap in recognition of Zoom, Grip and Exit

## 4 Improving the language

Based on the results of the simple model described in the previous section we can get some ideas on how to improve the language. We deal with ambiguity and overlap in the following text.

### *4.1 Removing Ambiguity*

The first improvement is to remove the ambiguity between *Zoom* and *Exit*. In the simple version the system would start zooming whenever a flat pose is recognised. The original gesture recogniser is also able to discriminate the orientation of the hand-position to a certain extent. In fact, in one version of the gesture language [2] the Zoom gesture is made with the back of the hand towards the user and the Exit gesture starts with a flat pose with the back of the hand away from the user or to the right of the user.

In the following model, shown in Fig. 3, we refine the flat pose that the user can perform into a flatBT (flat pose with back towards user) and flatBR (flat pose with back to the right). To avoid pictures with too many transitions, we also separate the pose recognition model from the set of poses that would imply the restart of the pose recognition. For example for the Grip gesture this set is modeled in the small automaton *NoGrip* and synchronised with *Grip* via the action *ng* that is forced to happen immediately after (asap) an unexpected pose is encountered.

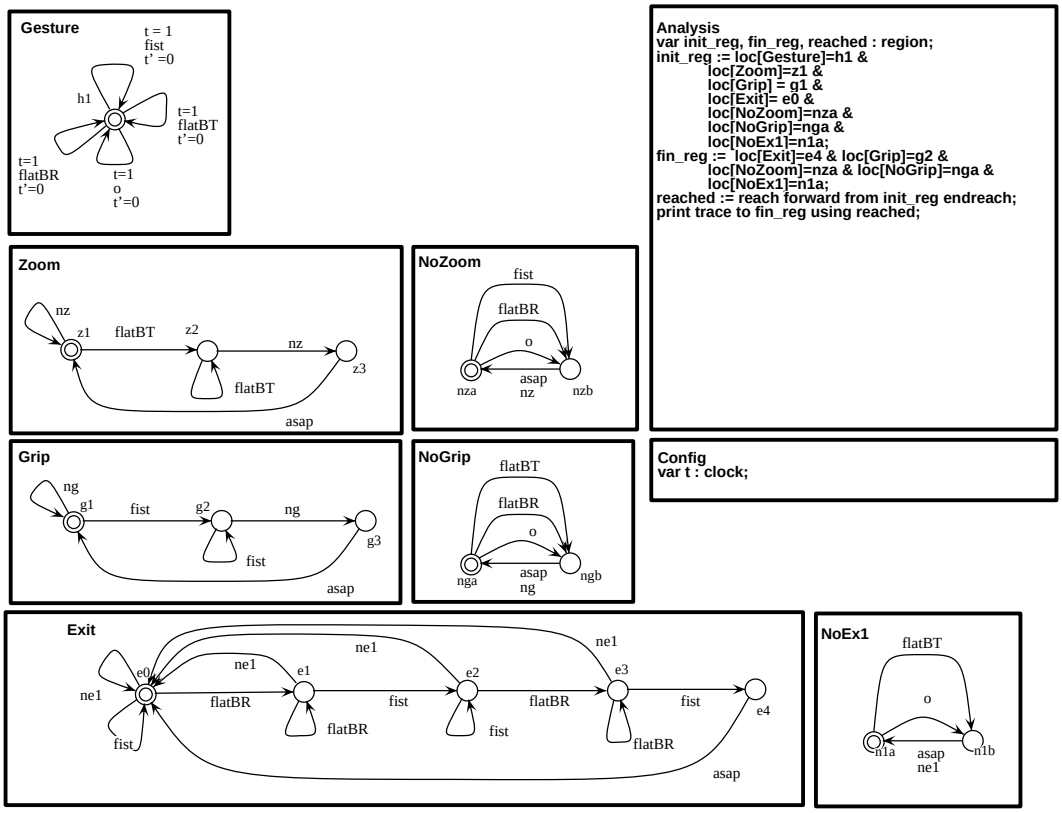


Fig. 3. Improved Gesture language: introduction of flatBT and flatBR

A similar analysis as explained in the previous section may be performed to check whether it is possible that two gestures are recognised at the same time in this new model. This is, unfortunately, again the case. A possible trace that illustrates that the locations $g2$ of *Grip* and $e4$ of *Exit* can be reached at the same time is flatBR, fist, flatBR, fist. See Table 4.1 for the details. When the last fist pose occurs, both Exit and Grip react to this pose, reaching `e4` and `g2` respectively resulting in both gestures being recognised.

One way to solve this problem is by shortening the Exit gesture and letting it be composed of a flatBR, a fist and a final flatBR. Model checking confirms that indeed `g2` and `e3` cannot be reached simultaneously any longer.

| Step | Time | Loc | Value t | Via |
|---|---|---|---|---|
| 0 | 0.00 | h1.z1.g1.e0.nza.nga.n1a | 0 | 1.00 time units |
| 1 | 1.00 | h1.z1.g1.e0.nza.nga.n1a | 1 | flatBR |
| 2 | 1.00 | h1.z1.g1.e1.nzb.ngb.n1a | 0 | nz |
| 3 | 1.00 | h1.z1.g1.e1.nza.ngb.n1a | 0 | ng |
| 4 | 1.00 | h1.z1.g1.e1.nza.nga.n1a | 0 | 1.00 time units |
| 5 | 2.00 | h1.z1.g1.e1.nza.nga.n1a | 1 | fist |
| 6 | 2.00 | h1.z1.g2.e2.nzb.nga.n1a | 0 | nz |
| 7 | 2.00 | h1.z1.g2.e2.nza.nga.n1a | 0 | 1.00 time units |
| 8 | 3.00 | h1.z1.g2.e2.nza.nga.n1a | 1 | flatBR |
| 9 | 3.00 | h1.z1.g2.e3.nzb.ngb.n1a | 0 | ng |
| 10 | 3.00 | h1.z1.g3.e3.nzb.nga.n1a | 0 | |
| 11 | 3.00 | h1.z1.g1.e3.nzb.nga.n1a | 0 | nz |
| 12 | 3.00 | h1.z1.g1.e3.nza.nga.n1a | 0 | 1.00 time units |
| 13 | 4.00 | h1.z1.g1.e3.nza.nga.n1a | 1 | fist |
| 14 | 4.00 | h1.z1.g2.e4.nzb.nga.n1a | 0 | nz |
| 15 | 4.00 | h1.z1.g2.e4.nza.nga.n1a | 0 | end of trace |

Table 4
Trace to final region

### *4.2 Removing overlap and operation surprises*

There remains the problem of overlap. Every time a user would like to make the Exit gesture, the Grip gesture also gets recognised and before the Exit gesture can be finished, they find themselves passing through gripping mode. This problem could be solved by requiring that, as for selection and querying, grasping objects can only be performed *during* navigation. This is shown in Fig. 4 in the model for *Grip*. Verification by model checking shows that the specification is now free from ambiguity and overlap.

Obviously, we should also check that after removing ambiguity and overlap all gestures can still be recognised. This is verified by a simple forward reachability search showing that for each gesture the location representing activation of the gesture can be reached.

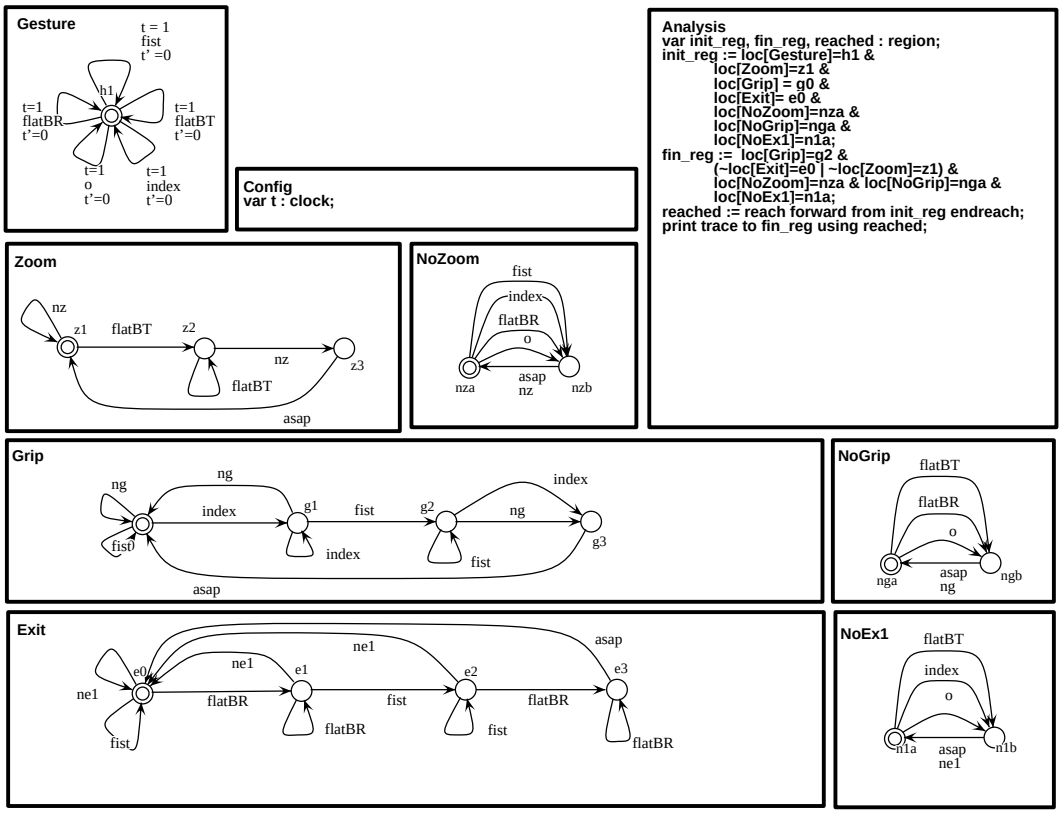


Fig. 4. Improved Gesture language: no overlap and ambiguity

### *4.3 Completing the model*

The original gesture language also has gestures for navigation, selection and querying. We can add their gesture models to the specification. Since both querying and selection are required to happen only during navigation, their models can be combined into one. Two new poses have to be added to the set of poses that a user can perform, and the other models have to be adjusted so that they deal properly with those new poses. The complete model is shown in Fig. 5.

The same analysis for detecting ambiguity and overlapping gestures can be performed as in the previous sections. Results in Table 5 show that there are no ambiguities left in the specification shown in Fig. 5. The numbers in brackets indicate the time it took HyTech to produce the results. The time is reported in seconds. Table 6 shows that there is an overlap possible concerning zooming and navigation. In the first situation (a) *Grip* can be in location g3 when *Zoom* is activated, but this means that only *Zoom* is active and *Grip* has finished and is going to its initial location without delay. A trace illustrating this is shown in Table 7.

In the second situation (b), whenever an index pose is recognised, navigation starts, but it forms also the first pose of the Grip gesture. In this case there is no risk for unexpected behaviour, because Grip, like querying and

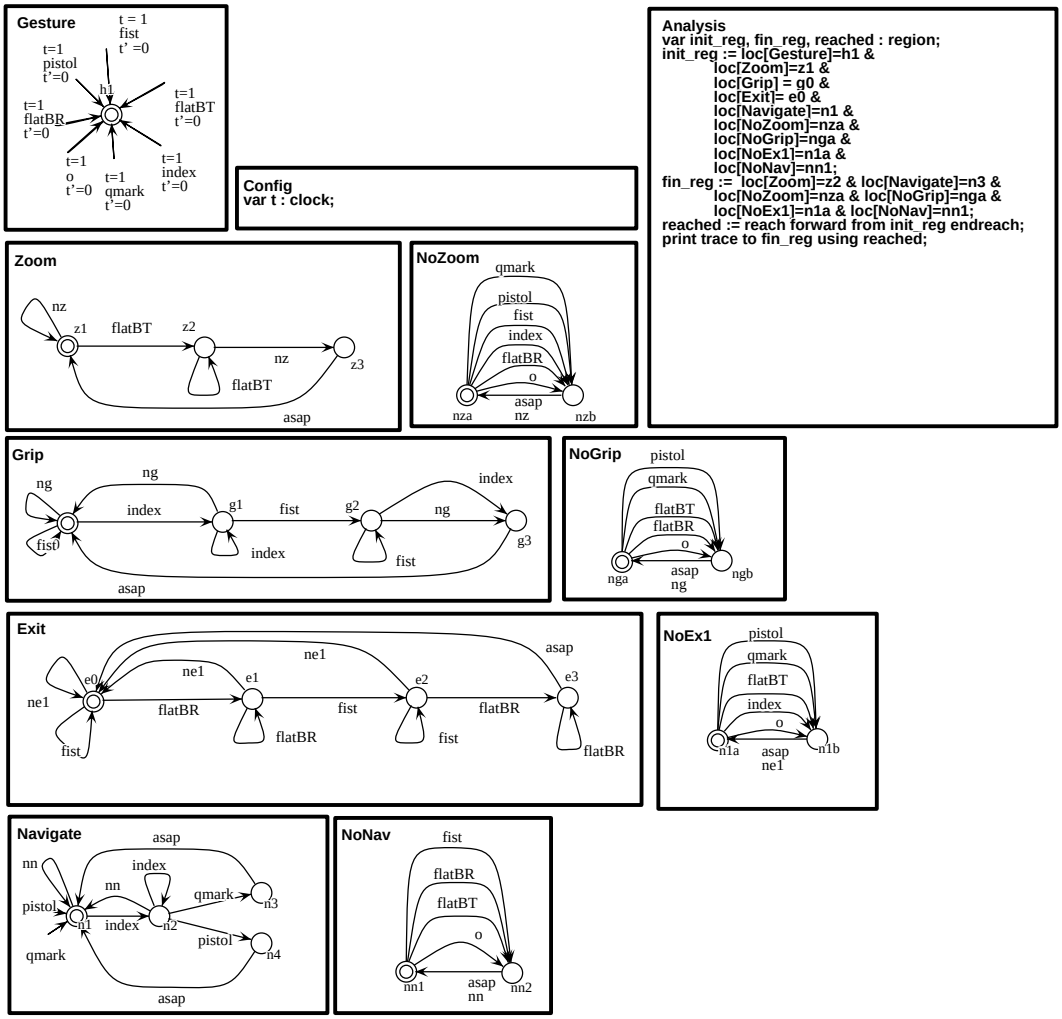


Fig. 5. Complete Gesture language

selecting, is an operation that is supposed to occur during navigation. A trace illustrating this is shown in Table 8.

| | Zoom z2 | Grip g2 | Exit e3 | Query n3 | Select n4 |
|---|---|---|---|---|---|
| Zoom z2 | – | No (4.92) | No (4.83) | No (4.97) | No (4.97) |
| Grip g2 | | – | No (4.95) | No (4.98) | No (4.97) |
| Exit e3 | | | – | No (5.01) | No (5.03) |
| Query n3 | | | | – | No (4.96) |
| Select n4 | | | | | – |

Table 5
Analysis detecting ambiguity

Also for the complete language it can be shown that each gesture may be

| | Zoom | Grip | Exit | Query | Select |
|---|---|---|---|---|---|
| Zoom z2 | – | g3(a) | e0 | n1 | n1 |
| Grip g2 | z1 | – | e0 | n1 | n1 |
| Exit e3 | z1 | g0 | – | n1 | n1 |
| Query n3 | z1 | g0 | e0 | – | – |
| Select n4 | z1 | g0 | e0 | – | – |
| Navigate n2 | z1 | g1(b) | e0 | – | – |

Table 6
Analysis detecting overlap

| Time | Loc | Value t | Via |
|---|---|---|---|
| 0.00 | h1.z1.g0.e0.nza.nga.n1a.n1.nn1 | 1 | index |
| 0.00 | h1.z1.g1.e0.nzb.nga.n1b.n2.nn1 | 0 | nz |
| 0.00 | h1.z1.g1.e0.nza.nga.n1b.n2.nn1 | 0 | ne1 |
| 0.00 | h1.z1.g1.e0.nza.nga.n1a.n2.nn1 | 0 | 1.00 time units |
| 1.00 | h1.z1.g1.e0.nza.nga.n1a.n2.nn1 | 1 | fist |
| 1.00 | h1.z1.g2.e0.nzb.nga.n1a.n2.nn2 | 0 | nz |
| 1.00 | h1.z1.g2.e0.nza.nga.n1a.n2.nn2 | 0 | nn |
| 1.00 | h1.z1.g2.e0.nza.nga.n1a.n1.nn1 | 0 | 1.00 time units |
| 2.00 | h1.z1.g2.e0.nza.nga.n1a.n1.nn1 | 1 | flatBT |
| 2.00 | h1.z2.g2.e0.nza.ngb.n1b.n1.nn2 | 0 | ne1 |
| 2.00 | h1.z2.g2.e0.nza.ngb.n1a.n1.nn2 | 0 | nn |
| 2.00 | h1.z2.g2.e0.nza.ngb.n1a.n1.nn1 | 0 | ng |
| 2.00 | h1.z2.g3.e0.nza.nga.n1a.n1.nn1 | 0 | end trace |

Table 7
Trace showing overlap in Zoom (g3(a) in Table 6)

activated using forward reachability analysis. The language model could be improved further by combining *Grip* with the *Navigate* automaton, so that it would be more clear that all three operations grasping, selection and querying can be performed only during navigation.

| Time | Loc | Value t | Via |
|---|---|---|---|
| 0.00 | h1.z1.g0.e0.nza.nga.n1a.n1.nn1 | 1 | index |
| 0.00 | h1.z1.g1.e0.nzb.nga.n1b.n2.nn1 | 0 | nz |
| 0.00 | h1.z1.g1.e0.nza.nga.n1b.n2.nn1 | 0 | ne1 |
| 0.00 | h1.z1.g1.e0.nza.nga.n1a.n2.nn1 | 0 | end trace |

Table 8
Trace showing overlap between Grip and Navigation (g1(b) in Table 6)

## 5 Human factors

In the previous sections we have shown how a simple automaton model can be used to detect language related properties such as ambiguity and overlap-recognition. In this section we pay more attention to the human factor aspects and their influence on the usability of the language.

### *5.1 Resilience to intermediate poses*

In general humans are unable to perform a pose more than once in exactly the same way. To deal with this phenomenon the poses are recognised in a way which allows a certain freedom to the user in forming a pose. It is clear that when this freedom is very small, relatively many poses would be interpreted as 'other', i.e. as an unrecognised pose. The consequence is that it is hard for the user to perform the series of poses required for the intended functionality. This can be illustrated in the specification by adding the model of a typical gesture made by a human, consisting of a few recognisable poses and many 'other' poses. An example of an automaton modelling a user who performs a Grip gesture is shown in Fig. 6 in the box labeled 'User'. Reachability analysis shows that a human making the modeled gesture in the original language does not obtain the intended result, even though the language related requirements are fulfilled.

One approach to deal with this problem is to make the language more resilient to intermediate poses. But in doing so, we would like to know the effect of such a change on the reliability of the language recognition. This is particularly useful to obtain a first indication about the performance of different recognition strategies.

One assumption we could make is that most unintended poses are not kept for a very long time, and therefore would almost always result in the recognition of an 'other' pose, i.e. one that does not coincide with any of the predefined intensional poses of the language. If this would be a valid assumption about the behaviour of the user and the recognition of poses, then we could relax the requirements for gesture recognition by allowing 'other' poses for a certain amount of time between every two 'real' poses. If the series

of 'other' poses lasts too long or when other, non-expected 'real' poses are recognised, the gesture recognition is interrupted as before.

In Fig. 6 an example is given of how the acceptance of a series of 'other' poses in a gesture may be modeled. The clock $d$ is used in the model of the user gesture to model the duration of intermediate 'other' poses performed by the user. The clock $w$ defines the maximum amount of time during which 'other' poses can be accepted by the gesture recognition process. If we constrain this clock to a maximum of 5 time-units the user-gesture modeled by the automaton User shown in Fig. 6 leads to the recognition of the Grip gesture. If the time constraint is reduced to one time unit, the Grip gesture is no longer recognised. This can be automatically verified by reachability analysis starting from the initial state and looking for a state in which the Grip gesture is recognised (i.e. Grip in state g2).

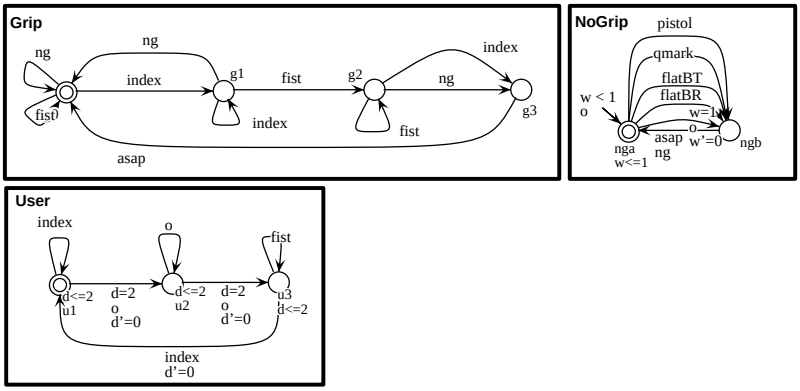


Fig. 6. Grip gesture accepting 'other' poses

Of course it should be shown that the modified language does not contain ambiguity or overlap problems. This may be verified in much the same way as we did for the version shown in Fig. 5.

## 6 Further analysis

There are many other questions that could be answered following a model based design of gestural interaction. We mention some of them in the following without working out the details in this paper. They are the topic of future work.

**Consequences of slips of action.** In some cases, a number of consecutive similar but wrong poses could be accepted for a while when recognising a gesture. This could work in the case of the Exit gesture, avoiding an accidental entering of zooming mode.

In the case of Navigation and Querying this would be less appropriate because it is unlikely that the unintended Qmark pose would only last for a short period of time. On the other hand, the Query function is not leading to serious mode surprises and can be easily recovered from by the user.

into another cannot be analysed with the modelling technique used in this paper. However, if there is the suspect or experimental evidence that some poses can easily turn into another pose, the consequences of this can be investigated based on an automaton model.

**Consequences of unreliable pose recognition.** Usually the recognition of poses and gestures are based on statistic techniques with a reliability to recognise poses correctly in the range varying from 50% to 100%, see for example [4]. This means that relatively often poses may not be recognised properly. Automata models that allow the expression of stochastic time such as SPADES [5] could form an interesting approach to model these issues at a sufficiently abstract level. This would allow to predict the reliability of the whole system based on data related to individual poses. Another important advantage of stochastic modelling is that it allows the utilisation of human performance data and gives much richer performance figures as the result of analysis.

**Consequences of delay in feedback.** An important topic that we mentioned in the introduction and that has not been dealt with in this paper is that of the consequences of delay and variation in delay of feedback on gesture recognition [6].

## 7 Conclusions

In this paper we have examined a dynamic gesture language for interaction in 3D user interfaces. We developed preliminary automata models providing systematic analysis of ambiguity and overlap in the gesture language and indicated a number of usability issues for which further modelling and verification could provide useful information for the design of gesture languages. The gesture language we studied was designed for a system that allows specification and recognition of dynamic gestures defined by sequences of dynamic and static hand poses. This is a common technique that can be supported by different devices such as data glove or vision based techniques. Single poses are recognised over time and compared with predefined models of these poses and models of gestures. Each gesture has its own recognition process that tries to recognise the gesture using the incoming pose data. The recognition processes for each gesture operate in parallel.

Although it is not difficult to specify an arbitrary gesture language for the system, it is far more difficult to detect afterwards what are the causes for possible malfunctioning of the interface. Factors that may have contributed to the problems can be language related, such as ambiguity and overlap in gesture recognition, or human related, such as limits on the tolerance on variation of a pose or gesture and limits on timeliness of feedback by the system. Some factors may lead to annoying operation surprises or oscillating behaviour due to over compensation by the user as a reaction to unpredictable delays in feedback.

Other difficulties are to foresee in the design phase the consequences of adaptations of the language to deal with typical human factors such as the difficulty to repeat gestures with great accuracy or to maintain a pose for a longer time as for example is needed when navigating.

In this paper we have dealt with only a small subset of the above problems as a first step towards the more ambitious goal of informing design decisions based on theoretical models of different aspects of gestural interaction. In particular we showed how a rather simple graphical timed automaton specification can be used to analyse automatically ambiguity and overlap in a gesture language. Further a timed extension of the model was developed to analyse a time dependent technique to make the language more robust. The analyses have been performed by means of reachability analysis provided by the tool Hytech. This tool provides an exhaustive search through the state space of the language specification.

The language specification can be considered as a first prototype of the gesture language on which a number of essential properties can be verified. This is a great advantage over an approach that relies only on experimental, a posteriori, validation of the interface for various reasons. First of all, as we have remarked previously, human users have difficulty repeating gestures in exactly the same way. This makes it hard to test the recognition system in order to find flaws in the language itself. The automata model provides a simple way to check for language related problems such as ambiguities or mode-surprises. Secondly, the reachability analysis tool gives a trace as a result in case a problem has been encountered. This is extremely helpful for improving the language where necessary. Further, we have shown that an automaton model can be used to investigate the effect of the introduction of techniques to cope with human factor related problems, such as making the language resilient to intermediate poses.

The relative ease of modelling and its complementarity to a posteriori validation of gesture based interfaces makes it worth considering formal modelling whenever possible. We do not claim that formal models allow us to find *all* problems and possible flaws. Rather we hope to have shown that formal modelling can be very helpful in finding some problems in an early stage of development in an area rather different from protocol design in the context of which most formal methods have been developed.

## 8 Acknowledgements

This research was supported by the TMR TACIT project funded by the EU under the TMR Programme Contract N. ERBFMRXCT970133. The authors would like to thank Michael Wilson for his valuable comments on an earlier version of the paper.

## References

[1] Barnard, P.J. and J. May.: 1993. Cognitive Modelling for User Requirements. In: P. F. Byerley, P. J. Barnard, and J. May, editors, *Computers, Communication and Usability: Design Issues, Research and Methods for Integrated Services*, North-Holland Series in Telecommunication, Elsevier.

[2] Bordegoni, M.: 1992. "Dynamic Gesture Machine". RAL Report 92-019, February.

[3] Bordegoni, M.: 1993. "Gesture Interaction in a 3D User Interface". Arbeitspapiere der GMD, no. 733.

[4] Martin, J., D. Hall and J. L. Crowley: 1999. Statistical Gesture Recognition Through Modelling of Parameter Trajectories. In: Proceedings of Gesture Workshop, Gif-sur-Ivette, LNAI 1739, Springer-Verlag.

[5] D'Argenio, P.R.: 1999. "Algebras and Automata for Timed and Stochastic Systems". PhD Thesis, University of Twente, ISBN 90-365-1363-4.

[6] Doherty G.J., M. Massink: 2000. Continuous Interaction nad Human Control. In: Proc. of the XVIII European annual Conference on Human Decision Making and Manual Control, pp. 80-96, Group D Publications.

[7] Duke, D.J.: 1995. Reasoning About Gestural Interaction. In: F. Post and M. Göbel (guest eds.), Eurographics'95, Blackwell Publishers.

[8] Fels,S.S.: 1990. "Building Adaptive Interfaces with Neural Networks: the Glove-Talk Pilot Study". University of Toronto, Technical Report CRG-TR-90-1.

[9] Harling, P.A., D.N.Edwards (Eds): 1996. Hand Tension as a Gesture Segmentation Cue. Proceedings of Gesture Workshop, Springer-Verlag, York, U.K.

[10] Harrison, M. D. and D. Duke: 1995. 'A review of formalisms for describing interactive behaviour'. In: R. Taylor and J. Coutaz (eds.): IEEE Workshop on Software Engineering and Human Computer Interaction: Joint Research Issues. pp. 49–75.

[11] Henzinger, T. A.: 1996. The Theory of Hybrid Automata. In: Proceedings of 11th Annual IEEE Symposium on Logic in Computer Science. pp. 278–292.

[12] Henzinger, T. A., P. H. Ho, and H. Wong-Toi: 1997. HyTech: A Model checker for Hybrid Systems. In: Proceedings of the Ninth International Conference on Computer Aided verification, Vol. 1254 of Lecture Notes in Computer Science. pp. 110–122.

[13] Jensen, P., M. Sørensen, J. Gravgaard, P. Christensen: 1996. Using Autograph to Create Input for Hytech. Available from http://www.docs.uu.se/docs/rtmv/uppaal/.

[14] Murakami, K. and H. Taguchi: 1991. Gesture recognition using Recurrent Neural Networks. ACM 1991, pp. 237-242.

[15] Polhemus: 1987. "3 Space user's manual, Polhemus." A Kaiser Aurospace & Electronics Company, May 22.

[16] Rushby, J.: 1999. Using model checking to help discover mode confusions and other automation surprises. In: D. Javaux (Ed.), Proceedings of the 3rd Workshop on Human Error, Safety, and System Development (HESSD'99), University of Liege, Belgium, 1999.