

NICE project (IST-2001-35293)



Natural Interactive Communication for Edutainment

NICE Deliverable D3.4-2 Gesture Interpretation Module

25 November 2004

Authors

LIMSI-CNRS : Jean-Claude Martin, Guillaume Pitel, Stéphanie Buisine

NISLab : Niels Ole Bernsen

Project ref. no.	IST-2001-35293
Project acronym	NICE
Deliverable status	Restricted
Contractual date of delivery	1 November 2004
Actual date of delivery	Thursday, 25 November 2004
Deliverable number	D3.4-2
Deliverable title	Gesture Interpretation module.
Nature	Report
Status & version	Final
Number of pages	28
WP contributing to the deliverable	WP3
WP / Task responsible	LIMSI-CNRS
Editor	Jean-Claude Martin
Author(s)	Jean-Claude Martin, Guillaume Pitel, Stéphanie Buisine, Niels Ole Bernsen
EC Project Officer	Mats Ljungqvist
Keywords	Gesture recognition, gesture interpretation
Abstract (for dissemination)	This report, Deliverable 3.4-2 from the HLT project Natural Interactive Communication for Edutainment (NICE), describes the Gesture Recognition and Gesture Interpretation modules used for the 2nd prototype.

Table of Contents

- 1 Introduction 4**
- 2 Gesture Recognition module (GR)..... 5**
 - 2.1 Informal specifications 5
 - 2.2 Algorithm 6
 - 2.3 Format of messages produced by GR 6
- 3 Gesture Interpretation module (GI)..... 8**
 - 3.1 Referable objects 8
 - 3.2 Informal specification 8
 - 3.3 Algorithm 9
 - 3.4 Format of messages produced by GI..... 13
- 4 References 15**
- 5 Appendix 16**
 - 5.1 Appendix 1: Re-visiting PT2 requirements (D1.1-2a Part2 section 4) 16
 - 5.2 Appendix 2: Training GR with gestures logged during PT1 user tests..... 19
 - 5.3 Appendix 3: Gesture Recognition Techniques..... 25
 - 5.4 Appendix 4: GI Bounding box algorithm 27

Frequently used acronyms:

- Gesture Recognition module (GR)
- Gesture Interpretation module (GI)
- Input Fusion module (IF)
- NISLab Character Module (CM)
- Telia Dialogue Manager module (DM)
- Hans Christian Andersen (HCA)
- Fairy Tale World (FTW)

1 Introduction

This report describes the Gesture Recognition (GR) and Gesture Interpretation (GI) modules in PT2. They have been designed by: considering the PT2 requirement specifications (cf. appendix 1), designing a gestural task analysis (done in cooperation with NISLab) and considering gesture shapes logged during PT1 user tests.

A 2D gestural input has several dimensions that need to be considered by the GR / GI / IF modules: shape (e.g. pointing, circle, line) including orientation (e.g. vertical, horizontal, diagonal) ; points of interest: depend on the shape (e.g. two points for a line) ; number of strokes ; location relative to objects ; media (mouse or tactile screen) ; size (absolute size of bounding box, size of bounding box relative to objects) ; timing: timing between sequential gestures. Gesture processing of these dimensions is a multi-level process involving the GR, GI and IF modules. The GR computes a “low-level” semantics from geometrical features of gesture without considering the objects in the study. The GI computes a higher level semantics by considering the list of visible objects as sent by the object tracker from the rendering engine (thus, the possibility that several objects are selected simultaneously cannot be detected by GR and has to be detected by GI). The IF computes a final interpretation of gesture by combining it with the NLU output.

The PT2 scenarios in either HCA Study or the two scenes of the Fairy Tale World (Cloddy Hans in HCA Study and in the Fairy Tale World) involve the gestural selection of object(s) or location(s). This was the only semantics that could be associated to user’s gestures as observed in video of PT1 user tests. Other semantics such as drawing to add or refer to an object, crossing an object to remove it are not compatible with the current NICE scenarios. The communicative acts for the FTW prototype are listed in D5.2b (request, ask, tell, confirm, disconfirm, askForSuggestion, askForExplanation). The table below lists the communicative acts identified for the scenario in HCA Study which is “indicate an object to get information about it” and which are expected to lead to gestural or multimodal behaviour.

	Communicative acts
1.	Ask for task clarification
2.	Ask for initial information about the study
3.	Select one referenceable object
4.	Select one non referenceable object
5.	Select several referenceable objects
6.	Select an area
7.	Explicitly ask information about selected object
8.	Negatively select an object (e.g. “I do not want to have information on this one”)
9.	Negatively select several objects
10.	Confirm the selection
11.	Reject the selection
12.	Correct the selection
13.	Interrupt HCA
14.	Ask HCA to repeat the information on the currently selected object
15.	Ask HCA to provide more information on the currently selected object
16.	Comment on information provided by HCA
17.	Comment on another object than the one currently selected
18.	Select another object while referring to the previous one
19.	Select another object of the same type than the one currently selected
20.	Move an object (user may try to do that although not possible and not explicitly related to the task)
21.	Compare objects
22.	Thank

Some difficult cases for gesture interpretation include:

- Multiple sequential gestures on the same object (eventually several gestures to do a single circle): In PT1, some users made several sequential gestures (e.g. parts of a circle) on the same object (which might be due to the fact that the selected object was not highlighted or to the fact that their finger slipped on the tactile screen). This resulted in duplicated messages sent by the GI and thus to some repetitions by the system. In order to avoid this, we proposed to: 1) have the GI provide feedback as soon as possible by having the character gaze at the selected object, 2) if several sequential strokes on the same object do not make a multi-stroke gesture (e.g. cross), the GI groups these as a single gesture on this object.
- Some objects have overlapping bounding boxes (eventually partly hollow such as the coat-rack).
- Some objects are partly hidden by others (e.g. the chair is behind the desk in several viewpoints).

This task analysis resulted in informal specifications of GR and GI described in the following sections. Both the GR and GI modules have been designed to feature the requirements of both the HCA Study and the FTW prototypes. We explain the differences we have included regarding input and output in order to cope with the differences between the two prototypes.

2 Gesture Recognition module (GR)

2.1 Informal specifications

The gestural task analysis resulted in the following set of shapes.

GR output class	Features of input gesture (shape and size)
Pointer	Point. Very small gesture (10x10 pixels) of any shape including garbage Very small line, tick, scribble
Surrounder	The following “Surrounding” gesture shape (for single object selection) were logged during PT1 user tests and are used for training the GR: - Circle, open circle, noisy circle, vertically / horizontally elongated circle. - “alpha”, “L”, “C”, “U”-like gestures with symmetrical shapes. - Square, diamond, vertical/ horizontal rectangle.
Connect	Vertical, Horizontal, Diagonal lines. Multiple back and forth lines.
Unknown	Garbage gesture. The bounding box is not very small (otherwise recognised as a point).

GR provides the 1st best gesture shape as output; scores of other shapes are available internally.

The 2 stroke “Cross” shape is recognised when 2 crossing lines are drawn. It is recognised by the GI (instead of GR) in order to avoid cumulating the delay between the two strokes of the cross with the delay between sequential gestures.

When a gesture is detected by GR, a <startOfGesture/> message is sent by the GR to the IF before launching shape recognition in order to enable appropriate timing behaviour in the IF.

Shapes “unknown” can be sent or not to the GI: when the GR is not able to recognise the shape or when the user makes noisy gestures, the GI can try to recover (considering them as

surrounder gestures) and hopefully detect any associated gestured object. The goal here is to reduce the non detection of gestured objects. Indeed, surrounder gestures logged during PT1 were quite noisy and included contour of objects. The other possibility is to force the user to gesture properly and not forward unknown shapes to the GI (yet the message <startOfGesture> sent before shape recognition has to be cancelled).

2.2 Algorithm

The GR uses a Neural Network trained with gestural data logged from PT1. This includes several steps: manual labelling of logged shapes, training, testing and tuning (details are provided in appendix 2). The GR is compared with some other techniques for gesture recognition in appendix 3. The general algorithm of the GR is as follows:

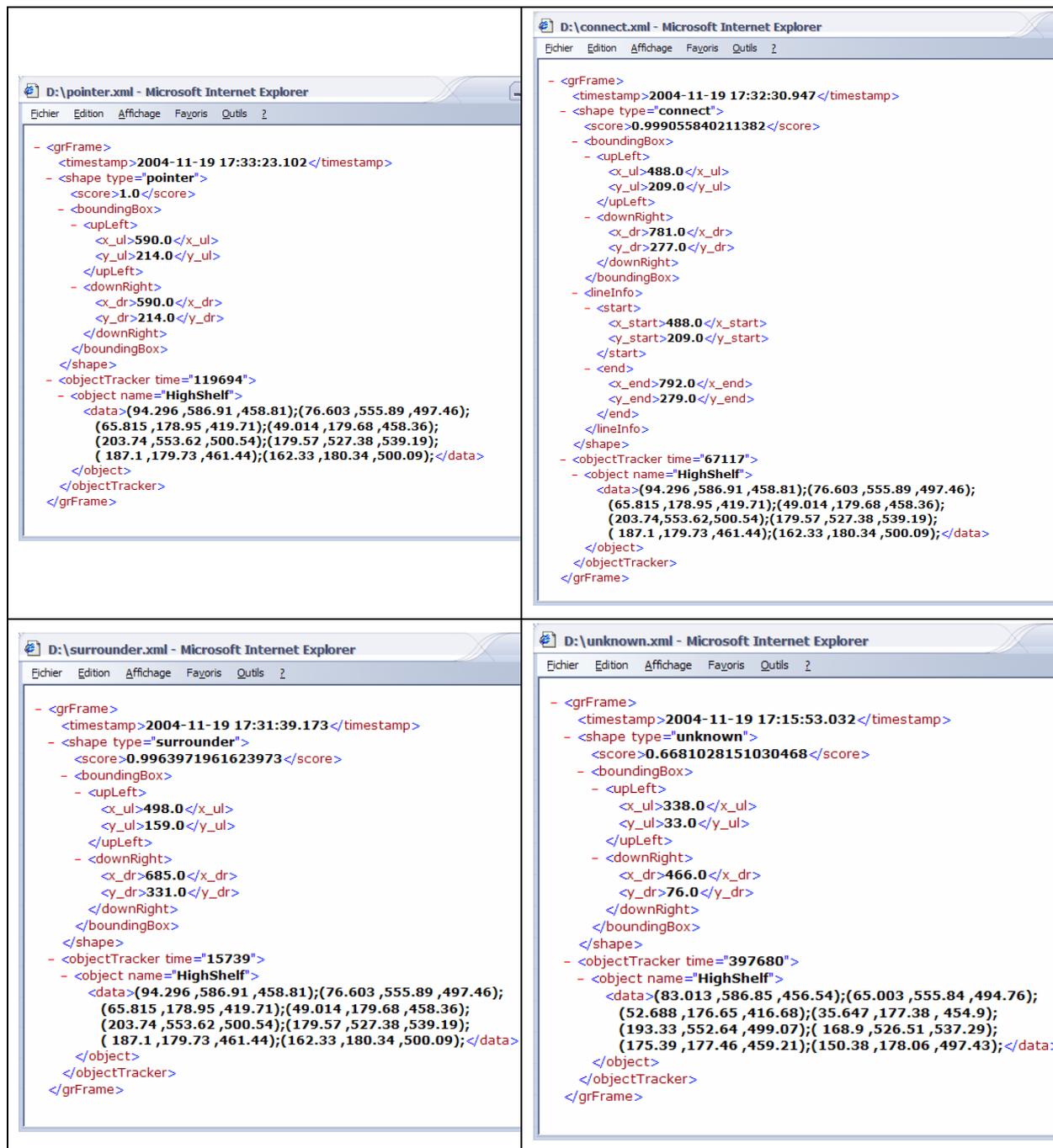
Algorithm GR

```
When a gesture is detected :  
    Send a startOfGesture message to IF  
  
    If the bounding box of the gesture is very small (10x10)  
    Then set shape = pointer  
    Else  
        Convert points to a slope features array.  
        Test the array with the neural network.  
        set shape = result from Neural Network  
            (either surrounder | connect | unknown)  
        If shape = connect  
        Then compute start and end points of the line  
  
        Build and send a grFrame for this newly detected gesture
```

End of Algorithm GR

2.3 Format of messages produced by GR

The focus point from PT1 GR messages has been removed because useless (the bounding box is used instead by GI). In case the shape is a connect, the two points of the line shape are provided in the message. We provide below examples of message for each output class (the object tracker section includes the list of visible objects).



Shapes tested during beta tests include shapes most observed in PT1 user tests (point, circle, vertical, horizontal and diagonal lines) which total 478 observed gestures (88%). Other less frequent shapes were also used for beta test (garbage, noisy circle, vertical rectangle, vertical circle).

When considering only frequent gesture shapes (point, circle, vertical, horizontal and diagonal lines), the success rate is 100%. The overall success rate in shape recognition is 96% when including all shapes including less frequent ones (eventually observed once or twice during PT1 user tests). Most recognition errors are related to “unknown” shapes which can be recognised as a surrounder.

3 Gesture Interpretation module (GI)

3.1 Referable objects

The GI has been designed by considering the properties of the graphical objects that can be displayed and that the user is able to refer to. This includes:

- spatial ambiguities due to objects that have overlapping bounding boxes, or objects that are in front or bigger objects (such as the objects on HCA desk),
- perceptual groups which might elicit a multiple selection with a single gesture, or for which a gesture on a single object might have to be interpreted as a selection of the whole group (such as the group of pictures on the wall in HCA Study, or the diamonds in Fairy Tale World).

The exact list of graphical objects displayed for each scenario is under discussion between the partners.

3.2 Informal specification

The requirements detailed in D1.1-2a have been considered during the design of PT2 GI (cf. details in appendix 1). It led us to specify the features that the GI should include in order to handle the requirements of different Characters in the two different prototypes. These features are listed in the table below and explained in the following sections.

GI Feature	Required in HCA Study	Required in FTW
Have a time out behaviour for collecting and grouping sequential gestures	X	X
Have an inhibited phase during which incoming gestures are not interpreted while the character is responding	X	
Requires objects sorted by score in case of several objects		X (for early fusion, cf. D3.6)
The GI can also provide: <ul style="list-style-type: none">• time-stamps for each object in the case of sequential selection,• score of each object,• objects' name sorted by timestamp.		

In order to group sequentially gestured objects, the GI has a relatively fast timeout. It collects what it gets before the timeout and then passes it on to the IF (and the NLU for the FTW). The message sent by the GI to the IF contains one or several objects. If several objects, this may mean either that a single gesture was done on several objects or that sequential gestures were done on different objects. An object does not appear twice in the giFrame even in the case of multiple gesture on the same object. The GI collects references to one or several objects in a given time window and passes them to the IF as a single gesture turn.

The HCA Study prototype requires that once the timeout is over, incoming gestures are ignored by the GI. The CM is expected to notify the GI with a <EndOfBehavior> message

the end of verbal and nonverbal behaviours so that the GI can start again interpreting gestures. This is not required for the FTW prototype.

Gesture confidence scores are ignored in HCA Study since a fast answer from the character is preferred over an in-depth resolution of ambiguity.

The types of results provided by the GI are defined in the table below.

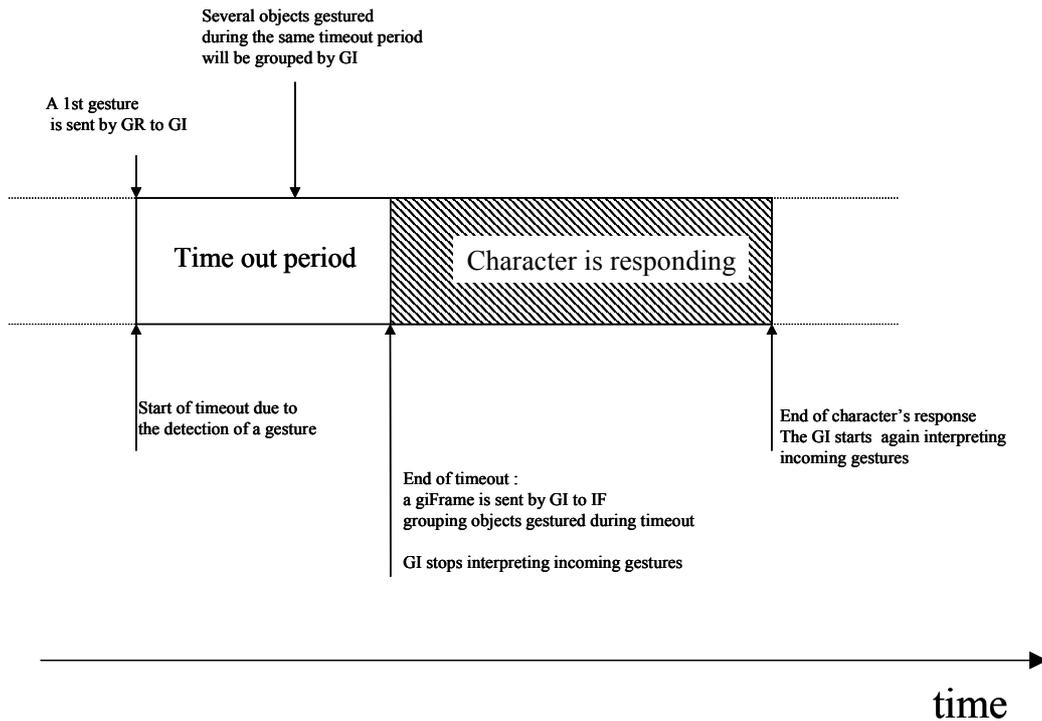
GI output semantic class	GR output class	Graphical context
select	Pointer Cross Surrounder Connect	Gesture bounding box overlaps with bounding box of only one object.
	Sequential : Pointer Cross Surrounder Connect	On the same object (close in time)
referenceAmbiguity	Surrounder Cross Connect Sequence of pointers or other shapes than unknown	Bounding box of gesture overlaps with the bounding boxes of several objects.
noObject	Any except unknown	GI failed to detect any object although a gesture was made by the user (gesture on empty space; selection of non referenceable objects).

3.3 Algorithm

3.3.1 Temporal behaviour with inhibition phase

Since the case with inhibition phase in GI is more complex, we present only this case. The algorithm without inhibition phase is simpler. The timeout period is reset each time a new gesture is recognised.

The temporal behaviour of the GI is illustrated in the following schema :



The following durations are proposed as default value for the module:

- Timeout period duration: 1.5 seconds (compatible with the literature and observations during PT1 user tests)
- Maximum duration of waiting for the character's response = 6 seconds (after this the GI starts again interpreting gestures)

3.3.2 General algorithm

Algorithm GI

Input: incoming messages from GR and CM

Output: messages sent by GI to IF

Variable: list of object(s) name gestured during timeout

// Processing of an incoming grFrame from GR

If a grFrame is received from GR

Then

If the character's response is currently pending

Then

Ignore grFrame

Else

If gesture time out period is not started

Then start gesture time out period

Call bounding box algo. to detect objects(cf. Appendix 4)

Store name of detected object(s)
in the list of gestured objects (avoid duplicates)

// Gesture time out period has finished

If end of timeout period

Then

If no object was detected during timeout

Then

 Build a "noObject" giFrame

If a single object has been detected during timeout

Then

 Build a "select" GIframe with name of this object

If several objects have been detected during timeout

Then

 Group objects names in a "referenceAmbiguity" GIframe

Send the GIframe to IF

Set characterResponsePending to true

// Character's response is finished

If message is <EndOfBehavior\> is received from the
Character/Dialog Module OR

 message <EndOfBehavior\> has been waited for too long

Then

 Set characterResponsePending to false

 Set gesture detection period not started

 Enable GI to start new timeout if a gesture is detected

End of Algorithm GI

3.3.3 Using Z dimension

In the 3D graphics, some objects hide others (for example in HCA Study a vase is hiding a table). Yet, the graphical application only delivers the coordinates of all the objects which are partly in the camera viewpoint without informing the GI if these objects are hidden or not by some other visible objects.

The objects which are hidden should not be selected by a gesture, even if this gesture is spatially relevant. In the bounding box algorithm (provided in appendix 4), we used the depth (Z dimension) of the closest side of the bounding box of objects. The salience value computed for each object is weighted by a factor of the distance, which is maximal when the front of the object is near the camera, and decreases quickly for objects which are far from the camera.

This is not a perfect solution, since an object closer on its Z dimension can actually be partially hidden by a farther one (for instance a vase on a table which hides the part of the table which is behind the vase). Thus the size of the object is also used in the algorithm. An object which fits better the size of the gesture will have more chance to be selected.

A better solution could be to allow a procedure call to a Zbuffer algorithm in the 3D engine, returning the object which is first visible at a given point of the view port.

3.3.4 Bounding boxes of objects

In some situations, the graphical application sends to the GR the bounding boxes of some objects that are indeed not visible in the current viewpoint. These bounding boxes are very large (x or y coordinates over 1000 or under -1000), and are thus always selected by the GI in case of connect gesture or pointing.

Discussions with LiquidMedia have led to the conclusion that this problem has two explanations: 1) the bounding boxes of the objects are in the frustum (cone of view) of the camera (so the objects are absolutely not visible for the user, but are considered as visible from the GI's point of view), 2) the bounding box of the objects are wrong (or too rough approximations since rectangular).

We have selected the following solution: the GI removes from the list of objects any object with x or y coordinates over 1000 or under -1000 since problematic objects are almost always with large bounding boxes while normal objects were within these bounds. Objects such as the window which are leading to such problems due to their approximate bounding box are ignored by the GI.

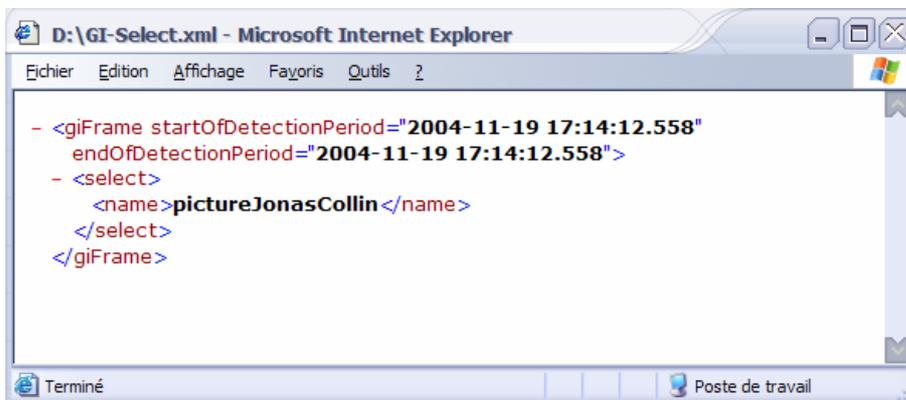
3.4 Format of messages produced by GI

3.4.1 noObject



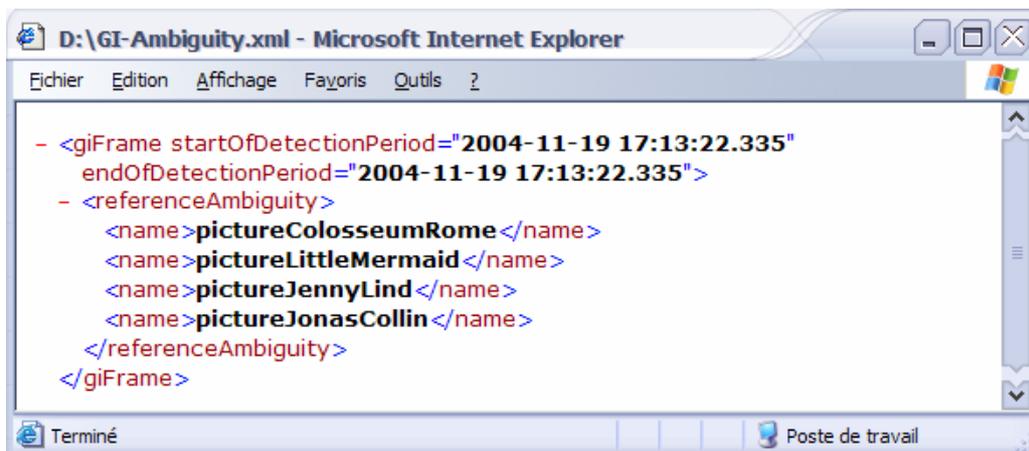
```
- <giFrame startOfDetectionPeriod="2004-11-19 17:15:53.032"
  endOfDetectionPeriod="2004-11-19 17:15:53.032">
  <noObject />
</giFrame>
```

3.4.2 select



```
- <giFrame startOfDetectionPeriod="2004-11-19 17:14:12.558"
  endOfDetectionPeriod="2004-11-19 17:14:12.558">
- <select>
  <name>pictureJonasCollin</name>
</select>
</giFrame>
```

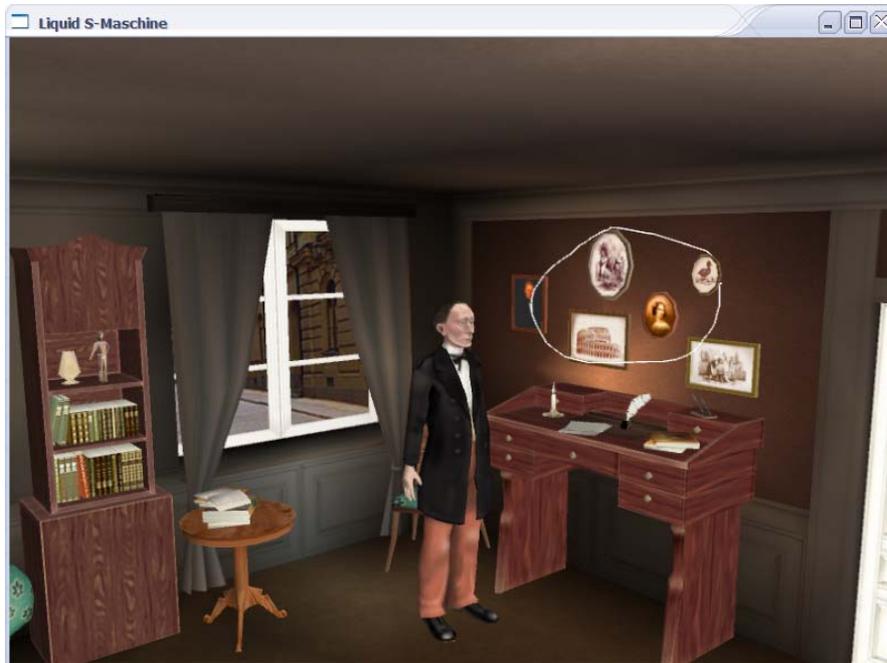
3.4.3 referenceAmbiguity



```
- <giFrame startOfDetectionPeriod="2004-11-19 17:13:22.335"
  endOfDetectionPeriod="2004-11-19 17:13:22.335">
- <referenceAmbiguity>
  <name>pictureColosseumRome</name>
  <name>pictureLittleMermaid</name>
  <name>pictureJennyLind</name>
  <name>pictureJonasCollin</name>
</referenceAmbiguity>
</giFrame>
```

4 Screen dump example

A gesture is done encircling several pictures on the wall in HCA Study:



Once the shape is completed by the user, the GR sends to the GI a frame containing the recognized shape (“surrounder” in this example) and the data about objects on the screen:

```
C:\WINDOWS\System32\cmd.exe
a.awt.Point[x=644,y=234], java.awt.Point[x=644,y=234], java.awt.Point[x=644,y=
t.Point[x=644,y=234]]
-----
Message for GI:
<grFrame>
  <timestamp>2004-09-30 17:15:55.009</timestamp>
  <shape type="surrounder">
    <score>0.9813835318405603</score>
    <boundingBox>
      <upLeft>
        <x_ul>470.0</x_ul>
        <y_ul>181.0</y_ul>
      </upLeft>
      <downRight>
        <x_dr>643.0</x_dr>
        <y_dr>310.0</y_dr>
      </downRight>
    </boundingBox>
  </shape>
  <objectTracker time="27896">
    <object name="HighShelf">
      <data>(37.631,606.76,443.37);(18.532,575.19,480.09);(4.7367,185.17,406.4
1,488.92);(127.44,542.17,525.64);(132.77,186.06,452.02);(106.86,186.66,
</object>
    <object name="ReadingChair">
      <data>(-215.9,807.62,298.36);(-321.3,733.89,339.11);(-269.9,418.71,274.3
```

The GI sends a message containing the name of objects that have been detected in a <referenceAmbiguity> message:

```

C:\WINDOWS\System32\cmd.exe
2D Bounding Box of DeskBooks = (556.12, 370.79, 641.32, 401.91)
2D Bounding Box of candle = (475.29, 324.99, 505.78, 363.93)
2D Bounding Box of chair = (335.05, 365.41, 435.57, 518.92)
2D Bounding Box of door = (753.16, 251.43, 1419.6, 844.67)
2D Bounding Box of featherPen = (548.49, 338.61, 578.46, 374.56)
2D Bounding Box of pictureColosseumRome = (503.33, 258.6, 569.82, 318.03)
2D Bounding Box of pictureHCAMother = (609.62, 284.94, 677.66, 343.65)
2D Bounding Box of pictureJennyLind = (569.06, 241.48, 604.55, 289.98)
2D Bounding Box of pictureJonasCollin = (452.39, 225.83, 485.56, 282.32)
2D Bounding Box of pictureLittleMermaid = (518.81, 182.81, 564.17, 251.61)
2D Bounding Box of pictureUglyDuckling = (612.54, 206.55, 643.84, 246.01)
2D Bounding Box of table = (137.31, 434.68, 278.94, 603.68)
2D Bounding Box of window = (125.75, 170.72, 331.78, 402.15)
2D Bounding Box of writingDesk = (425.97, 328.35, 681.24, 617.67)
2D Bounding Box of clock = (-40821.0, 1981.33, -1098.0, 8478.5)
2D Bounding Box of HC = (326.35, 249.15, 473.32, 578.25)
Java.awt.Rectangle[x=470,y=181,width=173,height=129]
-----
Message for IF:
GI_<glFrame startOfDetectionPeriod="2004-09-30 17:15:55.009" endOfDetectionPeriod="2004-09-
<referenceAmbiguity>
  <name>pictureLittleMermaid</name>
  <name>pictureJennyLind</name>
  <name>pictureUglyDuckling</name>
  <name>pictureColosseumRome</name>
</referenceAmbiguity>
</glFrame>

```

5 References

Jiazhi Ou, Susan R. Fussell, Xilin Chen, Leslie D. Setlock, Jie Yang. Gestural communication over video stream: supporting multimodal interaction for remote collaborative physical tasks, *Proceedings of the 5th international conference on Multimodal interfaces*, November 05-07, 2003, Vancouver, British Columbia, Canada

Dean Rubine. *The Automatic Recognition of Gestures*. PhD thesis, Carnegie Mellon University, December 1991

Tracy Westeyn, Helene Brashear, Amin Atrash and Thad Starner, 2003, Georgia tech gesture toolkit: supporting experiments in gesture recognition, *Proceedings of the 5th international conference on Multimodal interfaces*.

6 Appendix

6.1 Appendix 1: Re-visiting PT2 requirements (D1.1-2a Part2 section 4)

We study for each identified requirement how it is handled in PT2 (in *italics*).

4.2 Requirements on GR

These requirements concern only the shape of the gesture. The requirements involving the detection of selected object(s) are describe in the GI section.

4.2.1 Expects as input

The GR needs to receive the sequence of 2D coordinates sent by LM rendering module (works fine in PT1).

=> *Ok.*

4.2.2 Provides as output

A n-best list of gesture shape, confidence score, timestamps of start and end of gesture, gesture bounding box and possible points of interest for each shape (e.g. starting and ending point of lines).

=> *These information are available internally and provided differently to HCA Study and FTW. Only the 1st best is considered for output in NISLab HCA Study.*

4.2.3 Recognises the following mono-stroke gesture shapes: generalised spot touching (pointing, small line, small noisy gesture), circle, long line, small garbage gesture, long fiddle gesture

=> *This list of recognised classes of shapes has been updated under agreement with NISLab after gestural task analysis. It now includes pointer, connect, surrounder, unknown.*

4.2.4 Recognises the cross multi-stroke gesture

Otherwise provides output for each touch event.

=> *Ok (recognised by GI instead of GR for avoiding delaying the processing : waiting for the second stroke of the cross is combined with waiting for sequentially gestured objects).*

4.2.5 Works with mouse and tactile screen

=> *Ok. Three gestural models were built from PT1 logged files : mouse, tactile screen, union of both. The model built from union of both the mouse and the tactile screen gave better results (since containing more data for training).*

4.2.6 Set-up should minimise gesture noise

The set-up should minimise gesture noise, for example by emulating the efforts of 3D gestures as closely as possible (e.g. vertical tactile screen).

=> *Depends on user tests set up. Under discussion with both partners for user tests. HCA Study will probably use tactile screen. FTW will probably use gyro mouse or ordinary mouse.*

4.3 Requirements on GI

4.3.1 Expects as input

The GI needs to receive:

- From GR: n-best list of recognised shape
- From rendering module: objects (and their 3D bounding box) which are (even only partly) visible in the current viewpoint

=> *Receives only 1st best recognised shape in order to decrease combinatorial complexity (the score of n-best are available internally).*

=> *Problems related to the list of visible objects sent by the rendering modules are described in this report.*

4.3.2 Provides as output

The GI has to produce:

- A n-best list of object(s) selection(s) hypotheses (single object, two objects connected, several objects simultaneously selected) with timestamps and confidence score for each object selection hypothesis.

=> *Ok. As agreed with NISLab during gestural task analysis for HCA Study, the GI does not provide the CM with scores for each object.*

4.3.3 Reliable detection of gestured referenceable objects

Requirement regarding referenceable objects are described in the section "4.5 Requirements from LIMSI to other partners". GI will use the depth Z dimension of objects' bounding box for gesture interpretation (e.g. if one object is partly hidden by another object). Objects in the front will be given higher score than the one behind (e.g. for avoiding selecting the chair if behind the desk when both are candidates for gesture selection).

=> *A few problems remain (e.g. gesture on the vase induces detection of the table behind).*

4.3.4 Detects selection of a single object with pointing, circling or small line ("pointer" category)

=> *Ok : « select » message.*

4.3.5 Interprets several sequential gestures (pointing / circling / combinations of these) on the same object (tapping, or parts of a single circle, equivalent of false start) as a single selection of an object

=> *Ok : duplicate objects are avoided.*

4.3.6 Recognises as the simultaneous selection of different objects a single circle around several different objects

Issues will be considered such as problems raised by overlapping objects (e.g. a big circle on the desk could be interpreted either as selecting the whole desk or all the objects that are on the desk).

=> *Ok : « referenceAmbiguity » message.*

4.3.7 Recognises a line between two different objects

=> *Ok* : « *connect* » *shape*.

4.3.8 Forward several sequential gestures on different objects individually to the IF

It is expected that user could sequentially gesture on 1 to 3 objects, but this number might change depending on the distance between objects (e.g. the user might use a single gesture if they are close to one another, otherwise use several gestures). The GI will forward individually each individual selection to the IF to avoid having the GI wait for the end of the gesture sequence, and enable the IF to start fusion processing. The order needs to be kept for input fusion (associate order rank as a semantic attribute to each selected object).

=> *Finally grouped in the GI in a « referenceAmbiguity » tag (agreed with partners).*

4.3.9 Provides possible semantic function for cross (negation), line (relating), garbage gesture (noise), out of domain (fiddle gesture, gesture on non response)

=> *cf. table of classes of GI output agreed with partners. The cross is finally interpreted as a selection since more probable in the agreed scenarios.*

4.3.10 Manages ambiguous gesture between several objects

The GI will compute one confidence score for each gesture interpretation by combining 1) the overlapping of the bounding boxes of each visible object and the gesture with 2) the confidence score assigned to each shape hypothesis. This score will be forwarded to the IF.

=> *Does not use score and generate a « referenceAmbiguity » tag.*

4.3.11 Recognises selection of location

The need to detect location selection (e.g. assign a location semantics to the floor, the carpet, the desk or to the selection of places where no object stands) in the selected scenarios will be discussed.

=> *Not necessary in selected scenario (in Cloddy Hans in HCA Study, the possible locations are objects : machine slots).*

4.3.12 Manages selection of the remaining non referenceable objects

Although more referenceable objects will be added in the study, some non referenceable objects will remain. The GI should in this case not recognise any referenceable object and send to the IF a frame without any object.

=> *Ok* : « *noObject* » *message*.

4.3.13 Manages different categories of referenceable objects

3 categories of conversational objects : rich, poor, no response (e.g. carpet, wall).

=> *Not necessary in selected scenario*

6.2 Appendix 2: Training GR with gestures logged during PT1 user tests

6.2.1 Introduction

We trained a back-propagation neural network (BNN) used in the Gesture Recognition Module (GR) from corpus data, while the BNN was previously trained from artificial forms.

The two main goals of this approach were:

- Increase recognition success rate
- Adapt the GR to the input media (Mouse or Tactile screen)

It included the following tasks:

- CLASS: Classification of corpus data. A tool has been developed to visualize each capture of shapes (visual form + other information) drawn by the subjects of the PT1 experiment, and to manually assign a class label to the shape.
- TRAIN: Training the network with a subpart of the corpus data + classification information. The network is trained to associate each shape to the class proposed in T-CLASS.
- TEST: The trained network is tested with the rest of the corpus data. Success rate is computed and used for tuning the network configuration.
- TUNE: The configuration of the training (BNN configuration, number of training cycles, training/testing ratio) is tuned to obtain the best performance.

6.2.2 Tools

Three programs written in Java have been designed. The first one (Corpus-Display) is used for classification of the corpus. The second one (Discretize) is used to check visually the discretization model used both for training and testing. The third one (Train-Test-Tune-Network) is used for the TRAIN, TEST and TUNE tasks.

6.2.3 CLASS Task

The corpus with classification information (classification information is initially empty) is stored in a file, loaded at start-up, and displayed by the tool.

When the person intended to classify the corpus wants to modify classification information, she can directly jump to a given shape, using a session number and a gesture number. Then she can push the button associated with the class label she wants to assign to the shape, and the next shape is displayed.

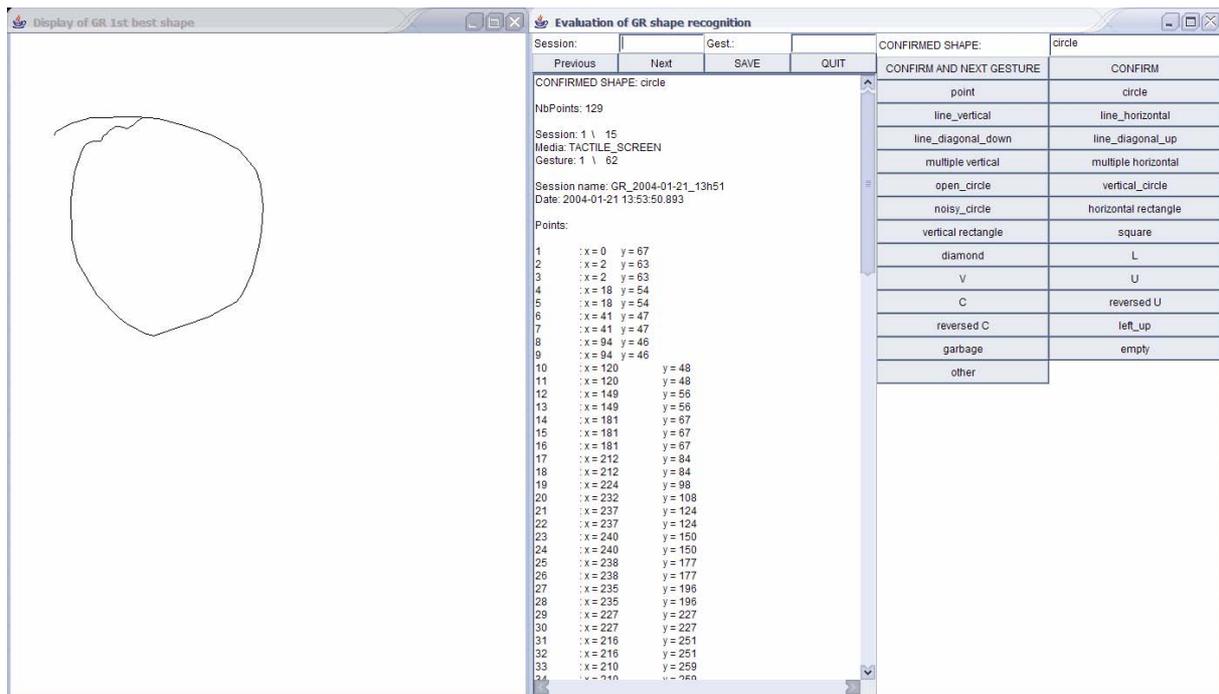


Figure 1 : Tool for CLASS Task. LEFT: visualization of the shape. MIDDLE : shape information + controls for corpus navigation. RIGHT : buttons for selecting the class label to assign to the shape.

6.2.4 TRAIN Task

The TRAIN task was designed using the Discretize and the Train-Test-Tune-Network tools.

6.2.4.1 Discretization

One of the constraints of using a neural network is that data must be described within a fixed size array of data. A first step consisted in transforming the shapes into a gray levels matrix. The smaller the matrix, the faster the training process (training speed is important in order to find a good configuration in a reasonable time). While the size of the matrix should have been a subject of the tuning, we preferred not to include it in the configuration, and adopted a (12 points)*(12 points)*(8 bits) matrix.

The second step was oriented into a vector approach. The idea was to extract statistical information about the curve, using first and second derivative. This second approach gives much better results, and has thus been finally preferred to the first one.

First Step

The discretization process is illustrated by a screenshot of the Discretize tool (Figure 2). The process of discretization is as follows:

- The gesture of the user is displayed in a small image with appropriate translation and scaling values, calculated from the original drawing bounding box.
- The destination drawing is configured to be anti-aliased, in order to decrease the step effect. This gives an approximate +5% recognition success.
- The drawing is made using a thickness which is proportional to the size of the original image. This way, the result is the same whatever the original size. Moreover, after we stated that some shapes drawn with a little number of points but with long distance between points were subject to recognition error, we found that the thickness has to be adapted for each line depending on its length, because a discretization of short lines

gave a darker result than a discretization of long lines, and thus a different behaviour for neural network learning.

- The final image is then blurred in order to capture (if learned) or be captured (if tested) more easily. The gain seems to vary in the interval of +5% and +8%. The parameters of the blur have to be selected with respect to the precision of the matrix. A too strong blur indeed induces a loss in recognition rate.

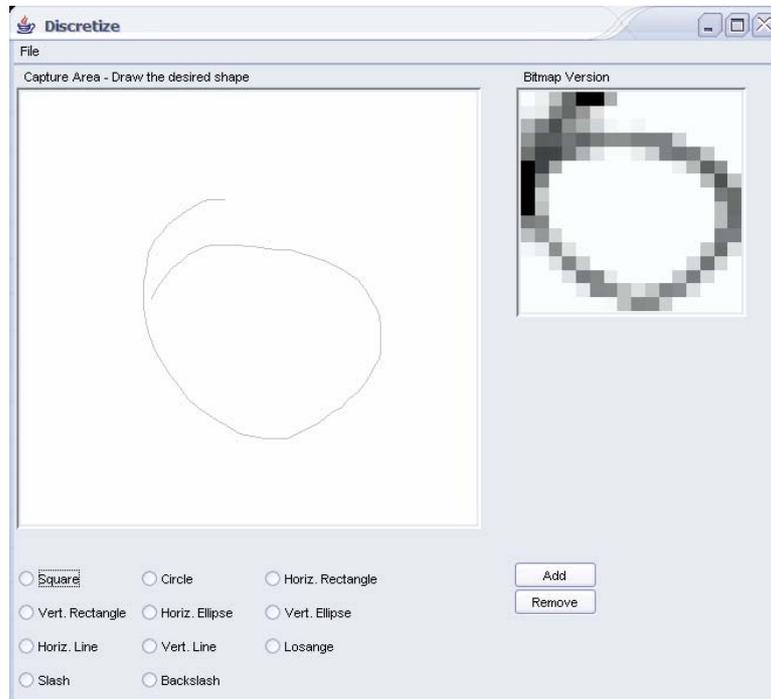


Figure 2 : Discretize tool. LEFT : drawing area. TOP-RIGHT : a 16*16 version of the discretized, scaled version of the drawing.

Second Step

In the second step, the following procedure is followed:

- The curve is refined, inserting points when distance between two points is too long, and removing points in the opposite case.
- The curve is smoothed, using a Bezier extrapolation;
- The sine and cosine of relative angles between each of two successive segments are computed, giving two arrays whose size varies between shapes.
- Mean and standard deviation of each array is computed, and used to build the representative of the curve.
- The sine and cosine arrays of the first and second derivative of the angles are then computed, and their mean and standard deviation are used to build the remaining of the representative of the curve.

6.2.4.2 Train set selection

In order to test the efficiency of the trained network, the data used for training must not intersect with the data used for testing. The user of the Train-Test-Tune-Network (TTTN) tool must thus select the ratio of data (control “Test/Learn ratio”) to use for training compared to the data used for testing when the training set (selected through combo box “Learning

source”) and the testing set (selected through combo box “Testing target”) come from the same source (data sources are either Mouse, Tactile Screen or both).

When the source for training data and the source for testing data are the same, a random set of shapes is selected in the training data and discarded for the testing data (testing data contains the remaining shapes).

As the training sets and testing sets are randomly built, it is necessary to run several train/test cycles and compute the average and standard deviation on the success rates returned by each cycle.

6.2.4.3 Training configuration

When run as a single configuration tester, the TTTN tool can be configured regarding the number of training iterations to perform before testing (“Learning cycles”), and the number of input and hidden neurons in the BNN.

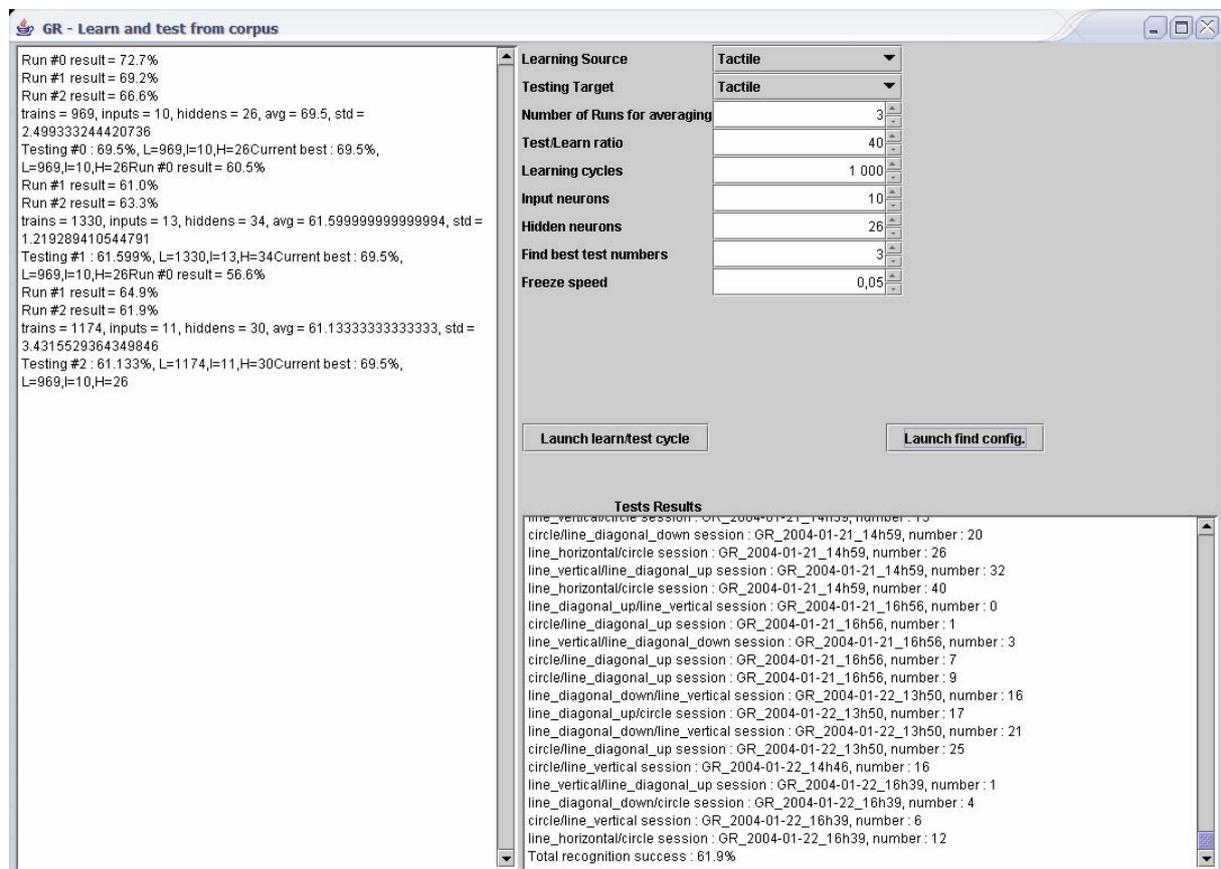


Figure 3 : Train-Test-Tune-Network tool. LEFT: results for the tune process. BOTTOM-RIGHT: results from the test process. TOP-RIGHT: configuration controls.

6.2.5 TEST Task

The test task is quite simple, since it consists in testing the BNN with the calculated testing set (which is calculated simultaneously with the training set).

Once the training is finish, the TTTN tool automatically launches the test, compares the results from the network against the target result (that is, the class label assigned to the shape during the CLASS task) and counts 1 point for a successful recognition (when the first recognized class is the target class) and 0 otherwise.

It then returns the percentage of successful recognition.

Since the training and testing sets are randomly built, it is necessary to repeat the process several times in order to obtain a significant value. This is configured through the “Learning cycles” control.

6.2.6 TUNE Task

The tune task consists in a random exploration of configuration values, with a simple simulated annealing approach. The number of test phases is controlled by the “Find best test numbers” control and the cold down speed is defined by the “Freeze speed” control.

6.2.7 Results

6.2.7.1 PT1 version

On a total of 542 shapes, 457 were correctly classified by PT1 GR. If the point is taken into account, it gives a recognition success rate of 86%. If points are moved out of the picture, the recognition success rate falls down to **75%**. This value is our comparison point for the new model.

6.2.7.2 PT2 version

All the line classes are grouped together. We indeed found that a lot of errors where confusion between different kind of lines. Moreover, it is easy to find the kind of line using the first and last point of the shape.

As the training data set and the testing data set must not overlap, the success of recognition of the previous version and of the new version cannot be directly compared (the success rate of the new version are higher if the training set and the data set are identical). As an alternative, we have to compare the success rate calculated for the new version on a basis of taking 90% of the corpus to train the network, and the remaining 10% for testing. In order to get enough precision on the results, the success rate is calculated several times with different training/testing sets, always with the same ratio.

First step

Results were as follows:

1/ concerning the data coming from the tactile screen:

90.5% success on non-point shapes (estimated overall success rate = 95%).

parameters = (trains=700, inputs=8, hidden=8) : average = **90.5%**, standard deviation = 7.5
{ 83.3%; 92.3%; 91.6%; 90.0%; 94.4%; 83.3%; 100.0%; 100.0%; 94.7%; 75.0%}

2/ concerning the data coming from the mouse :

83% success on non-point shapes (estimated overall success: 92%).

22 runs, parameters = (trains=1100, inputs=10, hidden=10) : { 86.9%; 76.4%; 88.8%; 88.8%; 77.2%; 100.0%; 93.7%; 93.7%; 82.6%; 61.1%; 88.2%; 76.9%; 78.2%; 70.5%; 80.0%; 78.9%; 76.4%; 86.9%; 72.2%; 100.0% } average = **82.87**, standard deviation = **9.72**.

3/ concerning the whole data set (with trains=1100, inputs=10, hidden=10): average = 82% success rate on non point shapes, standard deviation = 6.5. Estimated overall success: 91%.

4/ when both trained and tested with all shapes of either mouse, tactile or both modalities, the success rate is almost always **100%** (it may sporadically vary because of the random initialisation of the neural network).

Second step

For the second step, we only tested the training/testing with the whole corpus. The results were as follows.

Estimated overall success : **96%** (6% better than first step)

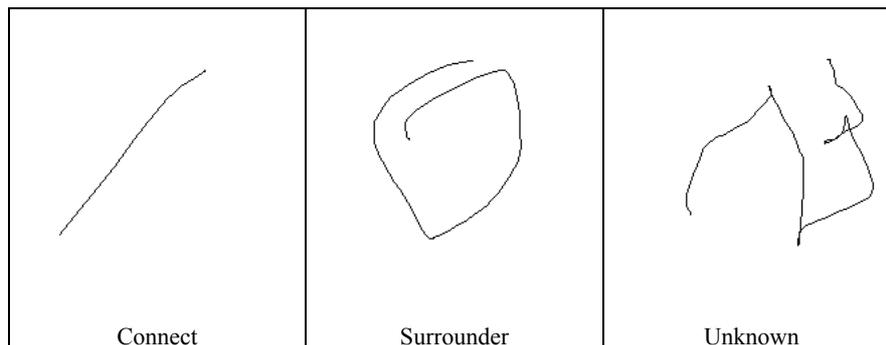
{90.47%; 98.18%; 97.14%; 92.06%; 91.93%; 96.36%; 95.91%; 89.85%; 94.82%; 92.42%; 91.38%; 96.0%; 91.07%; 87.93%; 89.28%; 92.54%; 90.74%; 93.06%; 90.48%; 91.67% }

trains=1510; Inputs=7; hidden=7; avg=**92.67**

6.3 Appendix 3: Gesture Recognition Techniques

In NICE, gesture recognition is primarily intended to be used in order to distinguish between four classes of gestures:

- Pointers,
- connectors (lines),
- surrounders (circles, half-circles, alpha-like shapes, squares, rectangles, triangles, free-forms),
- unknown.



Examples of shapes logged during PT1 user tests. Surrounder shapes feature high variation.

The important point here is that all forms are likely to be freely rotated or distorted, and more generally, that encountered shapes are not normalized forms.

6.3.1 Rubine approach

The Rubine algorithm is based on the original vector data of the shape produced by a user.

The principle of the Rubine approach to gesture recognition is to extract some features on the drawn shape, for instance distance and angle between the starting point of the gesture and the ending point, distance and angle between opposite corners of the shape's bounding box, and so on. This approach gives excellent results (97% for letters up to 100% for simple shapes) on standardized shapes, such as digits and letters, because it appears that these characteristics are relatively constant in those cases.

For our purposes, however, the possible forms are very unstable. The start and end points of the shape can vary and almost all positions can be found in a corpus of gestures.

6.3.2 Pixel-Based approach

In this approach, the shape to be recognized is not kept intact, but transformed into a pixel matrix. A classification method is then applied to match an observed shape with a predefined set of shapes.

This method is better than the Rubine approach for our purposes, since it does only depend on the final shape of the drawing, not on the way it has been produced. It however performs poorly with our particular application, especially because: 1) in order to take a reasonable time for the learning process, the matrix size has to be kept low, leading to recognition error when a surrounder is narrow (because two close lines can be confused into a single line),

2) there is an important variability of the shapes which are recognized as surrounders, and it harden the task of the learning process, because it has to automatically generates hidden classes which are not strictly distinguishable.

[Westeyn, 2003] reports a recognition success rate of 99.2%, with a set of seven distinguishable shapes.

6.3.3 Slope feature extraction

While the Rubine method is not adapted to our purposes, because of the fact that it relies on stereotypical shapes, and thus will not handle correctly the important variability of our recognition task, the “vector” approach seems more interesting than the “matrix” approach. Indeed, the “matrix” approach can only recognize non rotated, learned shapes, while the “vector” approach can learn shapes, based on the kind of movement realized by the user, and thus is rotation-insensitive, and can accept more variability of the shape.

The method of Slope feature extraction is based on computing the relative angle between each segment of the shape. Then, the resulting data array is used to learn and/or recognize a given shape. For instance, if a triangle is drawn, the slope feature array will show up two peaks, if it's a square, there will be three peaks. If the shape is a circle, there will be no peak, but a constant non-null value, and so on.

This method has been successfully used by the DOVE system [Ou et al., 2003] in order to recognize shapes from a set of twelve. The success rates go from 75% to 100%, depending on the target shape, with a overall average of 91%.

6.3.4 Gesture recognition in NICE

In the NICE project, we had to distinguish between points, connectors (lines) and surrounders (half-circles, closed shapes, circles, ellipses, alpha-like shapes, squares, rectangles, L-like shapes) and unknown shapes. We have first investigated the use of a Pixel-Based approach, which gave results of 82% for the generic case or better scores (83-90%) depending on the kind of input modality (tactile screen or mouse gestures). The main problem with this approach was that, in order to make the system usable, the pixel matrix into which the gesture was transformed was quite small (12x12 pixels) compared to the original shape (from 30 to 200 pixels). Consequently, a lot of gestures needing a better precision were misinterpreted.

In a second development step, we choose to keep the vector information, to compute first and second derivative on curve's slope, and to use this information in order to train the recognition network. This method gave us a success rate of 92.7% in the generic (modality-independent) case.

In the two experiments, we trained the networks on a part of a gestures corpus, and tested the network on the remaining, unused shapes. We removed from the corpus all shapes considered as pointing, because we have a 100% recognition success rate using a simple heuristic. Consequently, if we consider the whole corpus, including pointing gestures, recognition rates are respectively 90% and 96%.

6.4 Appendix 4: GI Bounding box algorithm

Algorithm bounding box

{ Algorithm for computing salience of an object for a gesture depending on the recognised shape (the weight used for computation were empirically selected) }

If the gesture shape is "Pointer"

Check the overlapping of a 8x8 box around the pointing spot and of the object bounding box

if there is an intersection,

then

if pointing spot in the object bounding box

then *inObject* = 1

else

{ Use a combination of

1) intersection of gesture and object,

2) size of object and

3) distance of gesture to center of object }

1) *inObject* =

(intersection (8x8 box around pointing spot , object's bounding box))
/

(surface of 8x8 box)

2) *objectSize* =

1 -

Min(1, Arctangent(1/Square Root (surface of object) / 70) / (PI/2))

3) *distanceToCenter* =

(smallest distance of the spot to the center of one of the object's axis) /
corresponding width or height

ZDistance = 1-(*objectMinZ*/1200)

{ Compute salience as the combination of these 3 values }

Salience = *inObject* *

(0.3 * *objectSize* + 0.1 * *distanceToCenter* + 0.6 * *ZDistance*)

Else *Salience* = 0

If the gesture shape is "Connect "

gestureRectangle = a rectangle so that the line of the gesture is its symmetry axis on the longest length, and the width of the rectangle is 20% of the line's length, and the height of the rectangle is the line's length.

If (intersection(gestureRectangle, objectBoundingBox) > intersection(gestureBoundingBox, objectBoundingBox))

Then

gestureSurface = gestureRectangle

Else

gestureSurface = gestureBoundingBox

overlappingSurface = intersection (gestureSurface, objectBoundingBox)

ZDistance = 1-(objectMinZ/1200)

Saliency =

0.2 * (overlappingSurface / gestureSurface) +

0.2 * ZDistance +

0.6 * overlappingSurface / union of gesture and object surfaces

If the gesture shape is "Surrounder "

overlappingSurface =

intersection of gesture bounding box and object bounding box

ZDistance = 1-(objectMinZ/1200)

Saliency =

0.35 * (overlappingSurface / gestureSurface) +

0.35 * overlappingSurface / union of gesture and object surfaces +

0.3 * ZDistance)