# FINGER — FORMALIZING INTERACTION FOR GESTURE RECOGNITION

GERHARD WEBER, PETER WETZEL

## 1. Introduction

Tactile braille displays and speech output devices help blind computer users to overcome their inability to receive feedback from a screen; nonetheless user interfaces based on graphics for output and pointing devices for input are still unusable by the blind. For example dragging an icon on the screen cannot verbally be described. A tactile representation of an icon also doesn't fit on the canvas of usual braille displays with 20 - 80 braille cells.

## 2. Pointing vs. Reading

Our attempts to overcome this inability of blind users to point on the screen is centered around our large, touch-sensitive tactile pin-matrix device (see Fig. 2). The fingers of the blind user are used for reading braille on the display and also for specifying input in the form of moving the finger on the display. Thus the finger is used for reading and as a pointing device by the user.

The pin-matrix device [Schweikhardt84] can be used to display tactile graphics in an area with more than 7000 moveable pins. Movements of the forefingers can be recognized by the user interface via a sensor system. With FINGER we allow the definition of a meaning for similar movements thereby introducing an expressiveness of movements similar to the expressiveness of gestures [Weber87]. Hence a powerful set of operations is required, so that finger movements can be uniquely interpreted by the machine. The user interacts with the computer on the basis of his movements.

As the user freely moves her fingers to form a certain shape, we call these movements gestures. The following definition for gestures is for use in human-computer interaction.

**Definition** -- A *gesture* can be defined as any bodily movement carrying a conventionalized meaning. Variations in the physical properties of a gesture can modify its meaning. We assume a certain standard of prototypical gesture-form as the carrier of the basic meaning. The degree of its variation in relation to the standard modifies the gesture's basic meaning.

## 3. Description of the language FINGER

With FINGER we want to present a language that formally describes the movements of fingers, and specifies the resultant operation, thereby formalizing the interaction. The implementation of FINGER on a SUN 3/50 consists of a compiler for the language and a recognition algorithm that acts according to the statements in a gesture "program". Shapes like triangles, circles, or crosses, have been successfully described. Gestures are related to selecting objects, specifying the movements of objects and on the deletion of objects.

Fig. 1 shows the definition of gestures in FINGER for two different kinds of angles. In Fig. 2 a sample movement is shown. Every filled circle marks a location where the finger has been. Fig. 3 describes how these points are segmented, how these segments are combined into simple gestures (basics) and how finally a command is formed. According to the definition of the command-pattern down in Fig. 1, the command that is expressed with the sample movement in Fig. 2 is "go forward by 5" with one numerical parameter.

```
initpattern:
   segment(1) S
   startsegment(sensor_up, 1)
   ((0,0) ?= A.START with 1000)
   commandpatterns: up, down
end

basic angle:
   segment(2) A
   segment(2) B
   startsegment(sensor_up, wait, 2)
end

 commandpattern up:
   basic ANG type angle
   ANG.A.START < ANG.A.STOP
   command: printf("go backward by %d",
                10*(ANG.A.STOP.Y-ANG.A.START.Y)/
                (ANG.A.STOP.X-ANG.A.START.X))
end

 commandpattern down:
   basic ANG type angle
   ANG.A.START > ANG.A.STOP
   command: printf("go forward by %d",
          10*(ANG.A.STOP.Y-ANG.A.START.Y)/
          (ANG.A.STOP.X-ANG.A.START.X))
end
```

Figure 1: Definition of a gesture in FINGER

The definition of the gestures in Fig. 1 begins with the specification of the initpattern. This is done by referring to a named segment that has been calculated by the segment recognition algorithm and with boolean expressions that must hold true, otherwise the initpattern is not recognized. Expressions can be comparative (compare within a range), arithmetic, or conditional. Moreover, assignments to variables and calls to external functions (in the programming language C) are possible. Simple variables

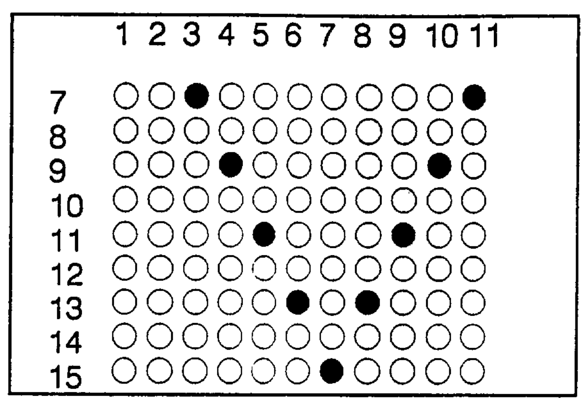


Figure 2: Part of the pin-matrix device

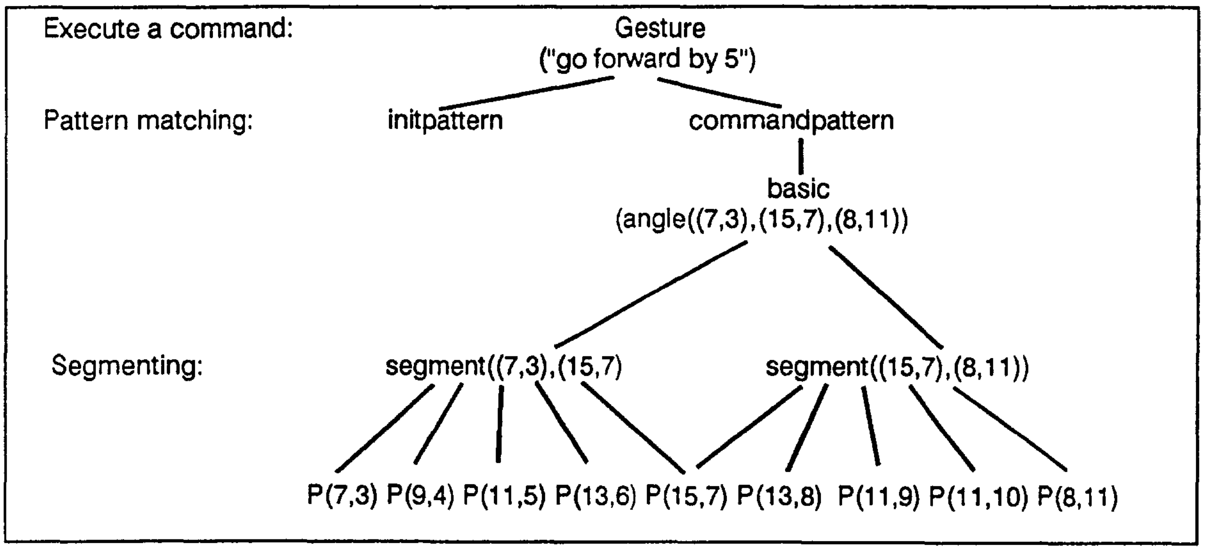


Figure 3: Recognizing gestures in FINGER

are the names of basics and segments. In general variables are notated in record-style syntax to get access to the components of basics and segments. Constants are numbers, strings and points (coordinates of a point are expressed in multiples of 100).

If the initpattern is recognized, the flow of control is split up so that different descriptions of gestures (commandpatterns) can be examined for every finger in parallel. Thereby the recognition algorithm acts like an ATN-interpreter. The commandpatterns are built up like the initpattern with expressions and named segments. Additionally basics that act like subroutines whose local variables are its parameters can be used. Every commandpattern is finished by specifying a command. The predefined external function printf is used therefore in our example, which is similar to it's meaning in C.

More refined calculations are necessary if the restrictions on a gesture are improved. Therefore constraints on segments are possible. Recursive basics are definable for movements like a circle.

These concepts of our programming language make it possible to calculate the variation of a gesture and thereby come up with parameters for the gesture. In our example the data about the starting and ending point of a segment can be used to calculate the steepness of the segment. Such a calculation is expressed in FINGER and not an implicit part of the recognition algorithm. FINGER allows to explicitly formulate the relations between the segments and to determine the properties of an movement. In the example we calculate how far the user wants to go and provide this number as an the parameter in the issued command. Compared to simple pointing only parameters provide additional expressiveness to gestures.

## 4. Conclusion

Parameterized gestures open-up an additional input channel in human-computer interaction. The programming language FINGER is a tool for the specification of gestures. An ATN-type interpreter recognizes parallel finger-movements according to the gesture definition. Gestures are a natural and intuitive form of human communication, which blind can use.

**Acknowledgements**

We wish to thank Prof. Dr. R. Gunzenhäuser and Dr. W. Schweikhardt who made this project possible.

The construction of the touch-sensitive pin-matrix device by E. Schäfer of the company METEC was the basis for our work. We wish to thank him as well the students M. Paul and G. Wahl who worked with the pin-matrix device.

**Literature**

[Schweikhardt84]

Schweikhardt, W.: Representing Videotex-pages to the Blind, in Proc. of Third Annual Workshop on Computers and the Handicapped, IEEE, 1984, pp.23-29

[Weber87]

Weber, G.: Gestures as a means for the Blind to interact with a computer, in Bullinger, H.-J.; Shackel, B. (eds.): Human-Computer Interaction INTERACT '87, North Holland, Amsterdam, 1987, pp.593-595

**About the authors:**

Dipl.-Inform. Gerhard Weber is a member of the research group "Applied computer science for the Blind" at the University of Stuttgart, F.R. Germany. He currently teaches Asian and European students in using graphics and 8-dot Braille at the Overbrook School for the Blind, Philadelphia, PA. His research interests are in alternative input and output devices that enrich the human- computer interaction and that make graphics and visually oriented user-interfaces accessible for the blind.

Dipl.-Inform. Peter Wetzel is a scientist at the Fraunhofer Institut IAO, Stuttgart, F.R. Germany. His research interests are in natural language processing, speech output devices and the design of user interfaces.