Gesture Interaction: $1/$N Recognizer

Objectives

The objective of this tutorial is to (re)code and extend a gesture recognition engine.

We will first explain the "One Dollar Recognizer" algorithm (see below for the link).

The objective is then to code an extension of the $1 Recognizer ($N Multistroke Recognizer) and to connect it on the ivy bus that we can then reuse in our multimodal application.

Reference documents

- http://depts.washington.edu/aimgroup/proj/dollar/
  Explanation of the project, example of "One Dollar Recognizer" to test, code samples

- http://faculty.washington.edu/wobbrock/pubs/uist-07.01.pdf
  Publication: Wobbrock, J.O., Wilson, A.D. and Li, Y. (2007). Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '07). Newport, Rhode Island (October 7-10, 2007). New York: ACM Press, pp. 159-168.

- http://faculty.washington.edu/wobbrock/pubs/gi-10.02.pdf
  Publication: Anthony L., Wobbrock J.O (2010). A Lightweight Multistroke Recognizer for User Interface Prototypes. Proceedings of Graphics Interface 2010. Ottawa, Ontario (May 31-June 2, 2010), pp. 245-252

One Dollar Recognizer

he implementation of a gesture recognition engine normally requires advanced knowledge of learning. For an HMI designer, it is therefore difficult to implement the recognition of specific gestures.

The objective of the "One Dollar Recognizer" algorithm is to enable the coding of a gesture recognition engine in different languages, without the need for the developer to have (quite) any knowledge. The algorithm has been shown to detect gestures at a rate close to that of more complex algorithms. Figure 1 shows the (unambiguous) gestures recognised by the algorithm. You can test the recognition engine on the project's website (see above).

The article describes the principles of the algorithm.

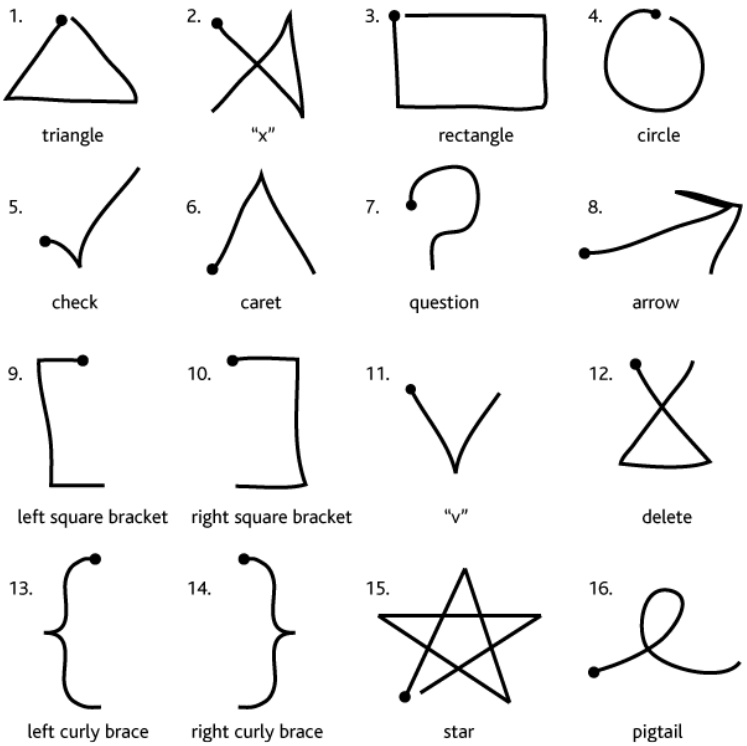

Figure 1: some gestures implemented in One Dollar Recognizer

In summary, the user's gesture is a trace of "**Candidate Points**" for which we must find the "**Template**" that most closely resembles this frame. In the rest of this document, we will speak about template and candidate.

The algorithm is independent of the way the gesture is made (with the mouse, on a touch surface, ...) but the hardware and the user's specificities determine the resolution of the points, etc. Therefore, these differences must be compensated for. The algorithm proceeds in several steps (described on pages 3 to 5 of the article). Steps 1 to 3 apply to the models as well as the traces.

1. **resample** the trace: the number of points per trace depends on the hardware, software and the speed of the user at the time of input. In a first step a resampling is done so that the points are equidistant on the trace and correspond to the number of points in the model. A number of points N between 32 and 256 would have given good results.

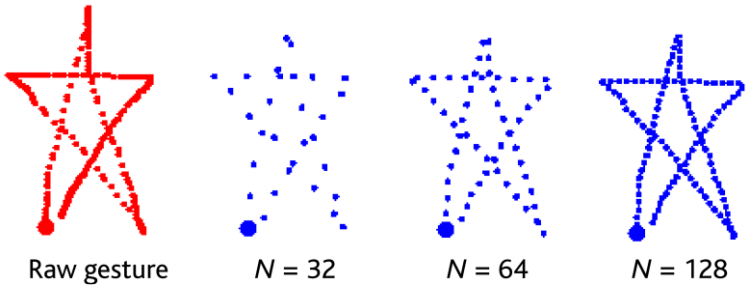

Figure 2: resample exemples

2. **Rotation**: turn the trace so that the angle between the centre and the first point is 0° (right, see Figure 3).

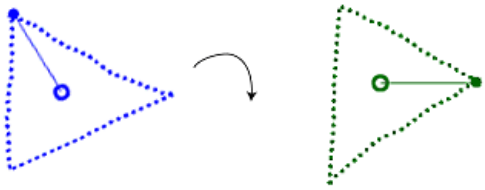

Figure 3: Rotation of the trace

3. **Resize and shift:** the trace is then resized to the size of a reference square (the proportions of the original trace are not respected). Then the centre of the trace is placed at the coordinate (0,0).

4. **Find** the best angle to obtain the best score: the average distance between the trace and the different models is calculated with the following formula:

$$d_i = \frac{\sum_{k=1}^{N} \sqrt{(C[k]_x - T_i[k]_x)^2 + (C[k]_y - T_i[k]_y)^2}}{N}$$

The model with the smallest distance is the selected model. It is also possible to convert this distance into a recognition score (see article page 4) and make a more precise rotation (see page 5 of the article).

Finally, the article provides in appendix (page 10) the pseudo-code of the algorithm.

The main steps to follow

1. Understand the algorithm described in this document and on pages 3 to 5 of the article

2. Download OneDollarIvy code here: https://github.com/truillet/OneDollarIvy, test it.

3. From the example provided, program the algorithm **$N Multistroke Recognizer** (pseudo-code here: http://depts.washington.edu/acelab/proj/dollar/ndollar.pdf) in any language you want.

4. **Program a module to allow new gestures to be added to the dictionary of recognised gestures.**

5. Integrate these modifications into an application that connects to the ivy bus. Determine the messages to be sent on the bus.

6. (**optional**) Integrate your gesture recognition with **Leap Motion** (https://www.leapmotion.com) for instance (**SDK** : https://developer.leapmotion.com).

7. (**optional**) Integrate your work in a way that allows you to apply this gesture recognition directly to a visualization element (widget)

There is no expected achievement, but if the application works and is integrated into the multimodal project, it will be credited as a bonus for the upcoming project.