Scale Detection for a priori Gesture Recognition

Caroline Appert[1,2]
appert@lri.fr

Olivier Bau[2,1]
bau@lri.fr

[1]LRI - Univ. Paris-Sud & CNRS
Orsay, France

[2]INRIA
Orsay, France

ABSTRACT
Gesture-based interfaces provide expert users with an efficient form of interaction but they require a learning effort for novice users. To address this problem, some on-line guiding techniques display all available gestures in response to partial input. However, partial input recognition algorithms are scale dependent while most gesture recognizers support scale independence (i.e., the same shape at different scales actually invokes the same command). We propose an algorithm for estimating the scale of any partial input in the context of a gesture recognition system and illustrate how it can be used to improve users' experience with gesture-based systems.

Author Keywords
Stroke, Recognition, Gesture, Scale

ACM Classification Keywords
H.5.2 Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Algorithms, Human Factors

INTRODUCTION
Gesture-based interfaces allow users to draw an arbitrary shape to invoke a command, providing expert users with a direct and efficient form of interaction. However, users have to learn the available gestures and their associated commands. This problem motivated research on online help systems improving users' transition from novice to expert. For example, Kurtenbach et al. [6] used crib-sheets showing the gesture set displayed in response to users' hesitation. More recently, Bau and Mackay [3] proposed OctoPocus, a dynamic on-line guide. If the user pauses during gesture input, the guide appears to show all possible gesture alternatives. As opposed to crib-sheets, that consume a large amount of screen space, the content of the guide is updated dynamically as the input grows and contains only the subset of gestures that match the partial input (Fig. 1-(a, b)).
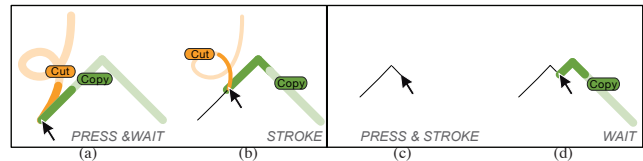
Figure 1. **(a-b) OctoPocus in novice mode (tracing *copy* causes *cut* to get thinner). (c-d) OctoPocus in intermediate mode (*cut* disappeared, and there is a scale mismatch for *copy*)**

These techniques provide effective assistance to novice users but their main limitation is their scale dependence: each template gesture has a fixed size. As illustrated with OctoPocus in Fig. 1, the template does not provide optimal guidance when the user's stroke does not match the template's scale, although the final gesture may be correctly recognized by a scale independent algorithm. While non-optimal guidance may be problematic in novice mode, such inconsistencies are even more likely to happen in intermediate mode where the user already started a gesture before invoking the online help (Fig. 1-(c, d)). In this latter case, the probability of scale mismatch between input and guiding paths is increased.

This scale independence problem motivates our main contribution: an algorithm that detects the scale of any incomplete-input relative to a gesture template. We first motivate the need for scale independence by empirical observations. We then detail the algorithm for scale recognition of incomplete input and present an evaluation of its accuracy. We finish by illustrating how our algorithm can improve users' experience with systems based on gesture recognition.

SCALE INDEPENDENCE
Long et al. have [7] investigated users' perception of gesture similarity. They presented sets of gestures to participants and asked them to select the one that they perceived as the most different. They tested whether scale was a discriminating feature for users by asking participants about gesture differences within triads of spiral gestures displayed at different scales. Results suggest that the gesture area was not significantly contributing to similarity judgment. This preliminary study, while very interesting, has some limitations. On one hand, scale has been investigated as one feature among many others, on a rather limited set of gestures. On the other hand, we are interested in the differences in scale of gestures drawn from memory, which is different than the visual perception of scale differences among three displayed shapes.
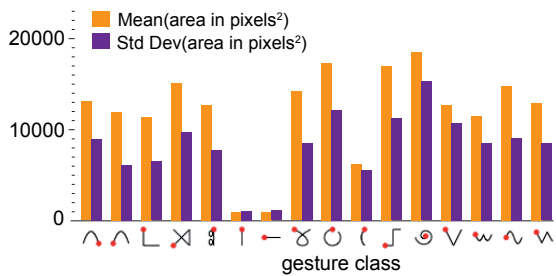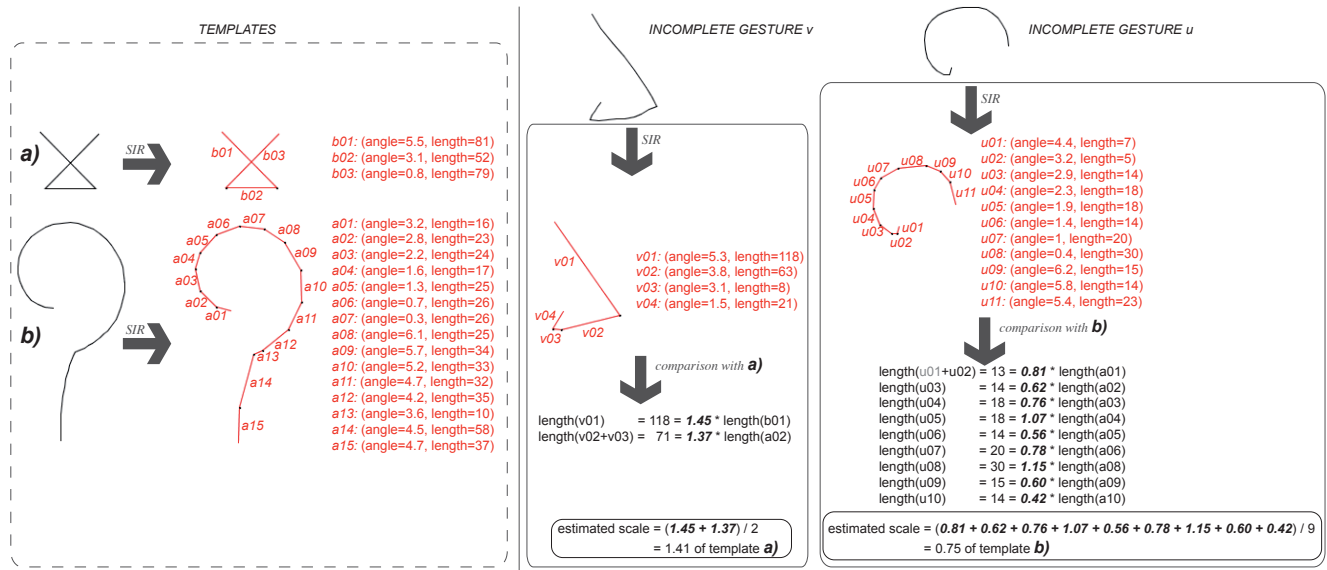
Figure 2. Scale estimation of two incomplete gestures based on our algorithm

Figure 3. Mean and standard deviation for 6 participants on 16 strokes.

As a further investigation of gesture scale variations for command strokes, we took a closer look at the stroke area on the data from an experiment conducted in [1]. In this experiment, the shape stimulus is presented in the center of the screen, always at the same scale, and disappears as soon as the participant starts drawing. Each of the 6 participants saw 11 blocks of 16 different stroke stimuli (see x-axis on Fig. 3). Fig. 3 reports the mean area and the standard deviation for each stroke stimulus over all collected answers to that stimulus. We observe a substantial standard deviation for most of the stimuli, showing that recognition mechanisms for command strokes should be robust to scale variations.

Many gesture recognition algorithms allow the recognition of gestures independently from their scale, such as the $1 recognizer [11]. Rubine's recognizer [10] can also be made scale independent if the training examples inhibit the proper features. Similarly, we are interested in adding control over scale independence to *incomplete*-input recognition.

INCOMPLETE INPUT SCALE ESTIMATION
We now present the algorithm that estimates the scale of an incomplete gesture. We were inspired by the "turning angles representation" algorithm used in image analysis [8]. It represents a shape as a vector of *turning angles*. The shape is sampled at a given number of equally spaced points to obtain a series of subsegments. Each subsegment can then be represented by the *turning angle* it forms with a reference axis, e.g., the x-axis. The distance between two shapes is simply computed as the distance between the two vectors of turning angles.

When considering partial input recognition, users provide only a prefix of the final gesture, making the sampling in equally spaced points irrelevant for comparing an input to a template. Thus, we use a modified version of this algorithm in **Step 1** of our own algorithm as described below. Given an input stroke v and a set of templates $\{\ a, b, c, ...\ \}$, the three steps of our algorithms illustrated on Fig. 2 are:

Step 1: Compute a *scale independent representation* (SIR) for input v and all templates a, b, c, ... A SIR can be seen as a coarse turning angle representation: it aggregates the subsegments that do not significantly vary in the angle they form with the x-axis (i.e., the difference is less than $\frac{\pi}{8}$). The SIR is thus a series of segments defined by their length (i.e., the sum of aggregated lengths) and their angle (i.e., the mean of aggregated angles).

Step 2: SIRs are convenient to look for the prefix that matches v in each template by simply comparing the successive segments. Two segments are similar if their difference in terms of angle is lower than `TOLERANCE`$=\frac{\pi}{4}$. If a segment v_i from v is not similar to the corresponding segment in a template, v_i is recorded as non matching. When the non matching length of consecutive input segments exceeds 10% of the length of v, the compared template is discarded. However a non matching part smaller than that threshold is considered as a noisy part of the last input segment (e.g., *v03* on Fig. 2) or the first input segment when this portion is the beginning of the gesture (e.g., *u01* on Fig. 2). Note that this tolerance to noise is applied only on the input side and not on the template side to avoid ignoring angle changes on small portions that are explicitly part of a template (e.g., a
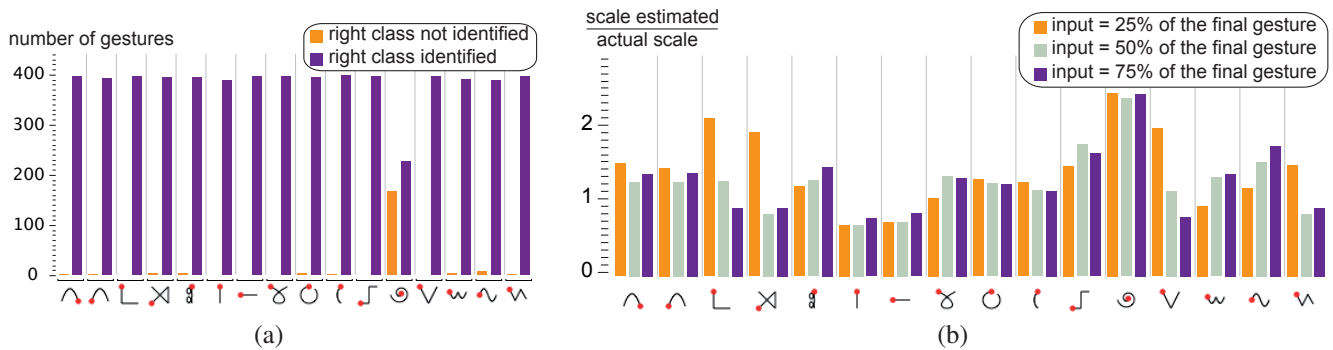
Figure 4. (a) Number of incomplete gestures with correct or incorrect class identification. (b) Ratio between estimated and actual scales by gesture class × input length.

curly brace stroke). Our algorithm thus considers templates as *perfect strokes*[1].

Step 3: Once the matching prefix (if any) is identified for each template, the scale ratio between the input and a matching prefix is computed as the mean of the ratios between lengths of matching pairs of segments.

EVALUATION

To assess the accuracy of our algorithm, we tested it on a set of incomplete gestures we generated from the data we already used for Fig. 3. We used each gesture at three different stages of incompleteness (25%, 50% and 75% of the total gesture length) and recorded the set of candidate classes our algorithm output to check if this set contained the right gesture class. This was not the case for only 187 incomplete gestures among 6318, which corresponds to a low recognition error rate of ∼ 3%. Fig. 4-a shows that recognition fails mostly for the spiral gesture which is very difficult to draw and thus exhibits too high a variability.

For recognized gestures, we computed the ratio between the size of the bounding box of the gesture as completed by our algorithm and the size of the bounding box of the actual user's complete input, which should ideally be equal to 1 (i.e., perfect accuracy). Figure 4-b shows that this ratio was ∼ 1-1.5 in most of the cases (the mean is 1.34 with a standard deviation of 0.53). The estimation is less accurate when only 25% of the total gesture is input, especially for gestures where the first 25% is a straight segment. This is not surprising since the scale estimation provided by our algorithm ignores the last segment in the SIR. Indeed, it is impossible to know if this last segment is complete or not and taking it into account would introduce too much uncertainty in the scale estimation. Thus, until the SIR of an incomplete gesture contains only one segment, our algorithms simply outputs the template scale.

APPLICATIONS

Visual supports

The partial-input scale estimation algorithm described in the previous section can be applied to improve existing guiding

---

[1]To facilitate template input, our program can load a set of files that contain the perfect templates as SVG polylines that can easily be produced with a graphical editor such as GIMP or Adobe Illustrator.
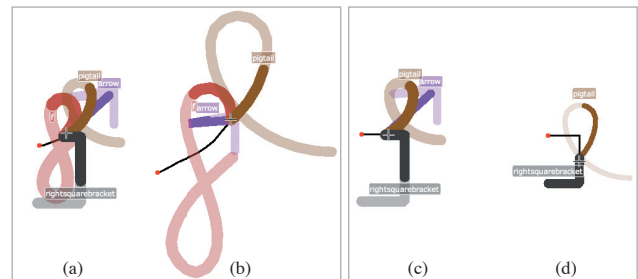

Figure 5. Scale detection for a visually coherent guide

techniques which can be invoked at any time while gesturing, such as the crib-sheets of the Tivoli system [6] or OctoPocus [3]. First, it allows to identify candidate gesture classes, resulting in less screen space consumption thanks to smaller crib-sheets. Second, the visual support they provide can be made more coherent by adapting the template scales to the scale of the current partial input. For example, Fig. 5 illustrates how OctoPocus can be improved: for each matching template, the missing part is rescaled using the proper ratio identified by our algorithm before being concatenated (Fig. 5-(a-b)).

Informal user feedback on the augmented OctoPocus prompted us to implement a scale computation policy that depends on the stroke's nature. On the one hand, dynamic scale adjustment at each input point is required for the guide to remain coherent when the user cuts corners in polyline sections as on Fig. 5-(c-d). On the other hand, computing the scale at each point for curves results in too much visual distraction. Actually a curved part such as the beginning of the question mark on Fig. 2 contains many small segments in its SIR so the scale estimation varies frequently (each time a new segment is detected) on the partial input. For an optimal user experience, we suggest to first update the guide at its invocation and then update it only when the segment lengths in the SIR exceed an acceptable length.

Motor support

Our algorithm is reliable enough to draw a set of reasonable predictions of user input. We can thus take advantage of it to add a magnetic mechanism to facilitate stroke input from a motor perspective. To that end, we "dig ditches" along the most probable template trajectories in motor space
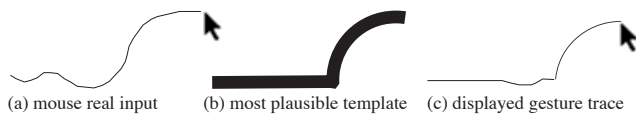
(a) mouse real input　　(b) most plausible template　　(c) displayed gesture trace

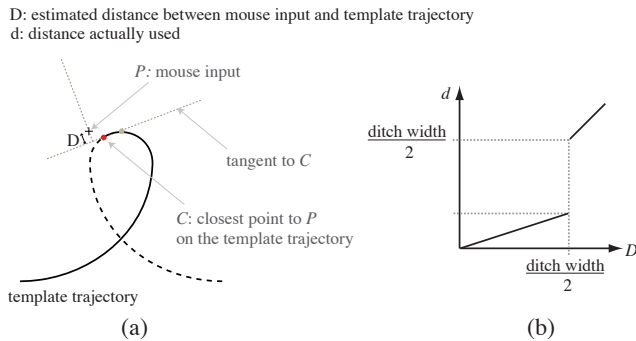**Figure 6. (a) Mouse raw input (b) Template (c) Actual gesture trace.**

Figure 7. (a) Distance estimation between mouse input and the ideal stroke trajectory - (b) Function linking mouse distance, D, to distance actually used in gesture trace, d.

in the spirit of kinematic templates [4]. The width and depth of each ditch are proportional to the template's plausibility given the current partial input: the more probable a template, the larger and the deeper its ditch. To provide a smooth and continuous behavior, ditch depth is maximal in the center and progressively decreases towards the boundaries. The users have more control than with a binary snapping mechanism, enabling them to deviate from the most probable trajectory to draw another gesture.

Each time our algorithm receives a new mouse input P, it computes its distance D to the current template candidates to determine the template which is the most probable at P. If P is out of any ditch, it is added to the current prefix without any treatment. If P is inside the ditch, we turn P, distant from D to the template, into P', distant from $d < D$, so as to simulate the ditch resistance. Fig. 7-(a) shows the computation of D and Fig. 7-(b) the function mapping D to d.

Fig. 6 shows an interesting side effect of magnetism: the stroke gets *beautified* as it is drawn. Not only does this inform the user that he can draw more quickly since the trace looks good, but it also provides a pleasant aesthetic experience. To our knowledge real-time beautification of arbitrary strokes has never been proposed before. Diagram beautification [9] or the SST toolkit [1] provide beautification by turning the stroke into its ideal template only *after* the stroke is fully drawn. Interactive beautification [5] provides a more real-time experience: it beautifies simple segments as soon as they are entered thanks to inferred geometric constraints based on other already existing segments. It is more powerful in the sense that it supports drawings composed of several strokes, but it is not able to handle an arbitrary shape made of a single stroke. In the same spirit, the prototype presented in [2] continuously morphs gesture input into ideal shapes but the algorithmic approach for the recognition engine differs

from ours and only handles families of predefined shapes (circles, axis-aligned boxes, and line segments) and, as Arvo and Novins point out, expanding the repertoire of shapes is more complicated than with our approach because it "involves more than simply adding additional classes to the recognizer" [2].

CONCLUSION

We have presented an algorithm for detecting the scale of incomplete gesture input and showed how to apply it to improve users' experience with gesture-based interfaces by providing better visual and motor support. We now plan to extend it so it can handle non uniform scaling (i.e., provide a "x scale" and a "y scale") to implement a fully functional drawing editor able to handle a large number of shape families, e.g., from simple ellipses to different kinds of arrows.

REFERENCES

1. C. Appert and S. Zhai. Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Proc. CHI '09*, 2289–2298, New York, NY, USA, 2009. ACM.

2. J. Arvo and K. Novins. Fluid sketches: continuous recognition and morphing of simple hand-drawn shapes. In *Proc. UIST '00*, 73–80, New York, NY, USA, 2000. ACM.

3. O. Bau and W. E. Mackay. Octopocus: a dynamic guide for learning gesture-based command sets. In *Proc. UIST '08*, 37–46, New York, NY, USA, 2008. ACM.

4. R. Fung, E. Lank, M. Terry, and C. Latulipe. Kinematic templates: end-user tools for content-relative cursor manipulations. In *Proc. UIST '08*, 47–56, New York, NY, USA, 2008. ACM.

5. T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: a technique for rapid geometric design. In *Proc. UIST '97*, 105–114, New York, NY, USA, 1997. ACM.

6. G. Kurtenbach and T. Moran. Contextual animation of gestural commands. *Eurographics Computer Graphics Forum*, 13(5):305–314, 1994.

7. A. C. Long, Jr., J. A. Landay, L. A. Rowe, and J. Michiels. Visual similarity of pen gestures. In *Proc. CHI '00*, 360–367, New York, NY, USA, 2000. ACM.

8. W. Niblack and J. Yin. A pseudo-distance measure for 2d shapes based on turning angle. *Image Processing, International Conference on*, 3:3352, 1995.

9. T. Pavlidis and C. J. Van Wyk. An automatic beautifier for drawings and illustrations. *SIGGRAPH Comput. Graph.*, 19(3):225–234, 1985.

10. D. Rubine. Specifying gestures by example. *SIGGRAPH Comput. Graph.*, 25(4):329–337, 1991.

11. J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proc. UIST '07*, 159–168, New York, NY, USA, 2007. ACM.