

Multiscale Detection of Gesture Patterns in Continuous Motion Trajectories

Radu-Daniel Vatavu¹, Laurent Grisoni², and Stefan-Gheorghe Pentiu¹

¹ University Stefan cel Mare of Suceava, Romania

² Laboratoire d'Informatique Fondamentale de Lille, France
vatavu@eed.usv.ro, laurent.grisoni@lifl.fr, pentiu@eed.usv.ro
<http://www.eed.usv.ro/~vatavu>

Abstract. We describe a numerical method for scale invariant detection of gesture patterns in continuous 2D motions. The algorithm is fast due to our rejection-based reasoning achieved using a new set of curvature-based functions which we call Integral Absolute Curvatures. Detection rates above 96% are reported on a large data set consisting of 72,000 samples with demonstrated low execution time. The technique can be used to automatically detect gesture patterns in unconstrained motions in order to enable click-free interactions.

Keywords: gesture recognition, pattern detection, multiscale, curvature, integral of curvature, motion trajectory.

1 Introduction

Gestures have become more and more present in today's human-computer interfaces with the recent advances in robust recognition techniques as well as in acquisition devices that became more and more available and affordable [7,9,13,16,17,20]. However, current gesture-based interfaces still isolate gestures by making use of user-driven discrete events such as mouse clicks, stylus up/down movements, pushing buttons on tracking devices or by requiring users to hold a specified hand posture in vision-based processing. Recognition techniques are further applied on such user-segmented gestures by using shape similarity methods well established in the pattern recognition community [7,9]. By adopting this self-segmentation approach, it is the users that let the system know when and where the gesture commands start and end with a direct impact on the fluidity of the interaction process. The alternative would be to automatically detect gestures in constraint-free continuous motions for which we propose a novel fast technique. The challenge is a difficult one cause of the multiscale problem: given two motion curves $G(s)$ and $F(s)$, find the occurrences of gesture $G(s)$ in the user-input trajectory $F(s)$ irregardless of scale.

2 Related Work

Gesture motions may be acquired using the mouse, stylus [7,16], WiiMote [15], specialized trackers, gloves and vision-based computing with dedicated

algorithms for detecting and tracking various regions of interest such as hands, arms or the full body. Moeslund et. al [9] and Poppe [11] provide excellent surveys on the trends in video-based human capture and visual analysis of human movement while Pavlovic et al. [10] focus on hand gestures. Also, good courses on sketching recognition are available [7]. With respect to gesture recognizers, several robust approaches have been proposed such as the Rubine’s classifier [13], the \$1 recognizer [20] or elastic matching techniques [17].

Common approaches for segmenting gestures make use of predefined hand postures that simulate click events: Vatavu et al. [17] signal the beginning and end of a gesture by pointing and retracting the index finger; Wilson [19] uses the pinch gesture in the TAFFI interface; Vatavu and Pentiu [18] combine hand open and hand close. Not only posture but location has been used as well: Cerlinca et al. [4] use predefined regions of interest around the human body in order to facilitate segmentation and recognition of free hand gestures; Marcel [8] defines a body-face space with various motion and color sensitive zones.

Several attempts have been made with regards to the automatic detection of gestures in unconstrained motion. Reng et al. [12] pose the problem of identifying primitives in body motions and consider an error measurement based on density. Dong et al. [5] propose a greedy approach for segmenting body motions from long video sequences into a predefined set of motion templates. The authors do not report execution times nor complexity orders and the technique was evaluated for off-line segmentation. Arvo and Novins introduce fluid sketching [1,2] as a technique for on-the-fly recognition and morphing of users sketches to predefined classes of simple geometric shapes such as circles, boxes, lines or Bezier shapes by least-squares approximation. The motion doodles of Thorne et al. [16] represent a technique for parsing sketches into predefined tokens using a corner detection algorithm and classifying segments into straight or curved. The approach is simple and limited to 4 orientations of the straight lines and 2 directions of the curved segments.

Non-template-based approaches that split continuous motions into meaningful gestures without previous learnt templates are worth pursuing in the context of natural gesturing. Segmentation is handled in this case using various cues such as pauses in motion, hand tension or movement effort [14].

2.1 Contribution

Detecting gesture patterns in unconstrained motions is a difficult problem and current approaches address it partially by limiting the patterns to simple shapes. The problem is hard due to the multiscale issue: a (naive) algorithm that would try to match all the possible candidates would fail within the constraints of real-time interfaces. The main contribution of this paper is represented by a fast technique that rejects the majority of weak and unfit candidates. We introduce for this purpose the notion of integral of absolute curvature by taking an approach from differential geometry. By making use of a curvature-based representation, our gesture detection is rotation invariant as well. Reported results show detection rates above 96% for a large dataset of 72,000 samples.

3 Motion Representation

For all our further discussions, we will consider a gesture motion as a point moving in time, $c(t) : [0, T] \rightarrow \mathfrak{R}^2$ in the continuous domain as well as a sequence of sampled points, $C = \{p_i = (x_i, y_i), i = \overline{0, n-1}\}$ in the discrete case. We limit our method to 2D gesture motions only. Also, we are not particularly interested in the acquisition device as long as we dispose of the discrete representation of the captured motion. For example, development of the technique was carried mostly on motions acquired using the mouse while the actual testing described in section 5 and the performance results are discussed on gestures captured using a stylus as it best matches the real-world experience (i.e. the pen).

We briefly describe below the main notions employed throughout the paper. When considering the continuous case of a motion curve, $c(t) : [0, T] \rightarrow \mathfrak{R}^2, c(t) = (x(t), y(t))$, we make use of arc-length $s(t) = \int_0^t \|c'(u)\| dt$ where $\|c'(u)\|$ represents the norm of the vector $c'(u)$, $\|c'(u)\| = \sqrt{x'^2(t) + y'^2(t)}$. The curve c may be re-parametrized by taking the arc-length s as the new parameter which leads to $c(s) : [0, L] \rightarrow \mathfrak{R}^2$ where L is the length of the curve, $L = \int_0^T \|c'(u)\| dt$. We equally employ the notion of curvature defined as the signed magnitude of the second derivative of the motion curve, $\kappa(s) = c''(s)$. Equivalently, curvature may be expressed as the rate of change of the tangential angle with respect to arc-length, $\kappa(s) = \frac{d\phi}{ds}$. All the above notions pertain to the differential geometry of curves [3].

Although the reasoning will be performed using differential geometry, the associated algorithms need to work with the discrete representation of a curve, $C = \{p_i = (x_i, y_i), i = \overline{0, n-1}\}$, for which we have the corresponding definitions of arc-length and curvature:

$$s_i = \sum_{j=0}^{i-1} \|p_j - p_{j+1}\|, \kappa_i = \frac{\langle p_{i-r}p_r, p_i p_{i+r} \rangle}{\|p_{i-r} - p_r\| + \|p_i - p_{i+r}\|} \quad (1)$$

where $\langle \cdot, \cdot \rangle$ represents the angle between two line segments and r is a fixed value representing the sliding window size for discrete curvature computation.

3.1 Acquisition and Preprocessing of Motion Trajectories

Acquired motions, irrespective of the capture device, are usually composed of points sampled at a given time interval τ depending on the working frequency of the device: $\{p_i = p(\tau \cdot i) = (x_i, y_i) \in \mathfrak{R}^2, i = \overline{0, n-1}\}$. The initial trajectory is usually raw and noisy so it needs preprocessing with a polyline reduction technique. We use the fast version of the Douglas-Peucker algorithm [6] which provides a set of significant points $\{p_{i_k}, k = \overline{0, m-1}\}$ and then re-sample each interval $[p_{i_k}, p_{i_{k+1}}]$ at equal length with a given resolution r ($r = 3$ points in our approach) in order to get a smoother version of the initial data. Figure 1 shows the result on a continuous motion trajectory including a star pattern. Preprocessing also acts as a data reduction strategy where the initial 210 points of the star motion are reduced to 52 with 18 most significant points.

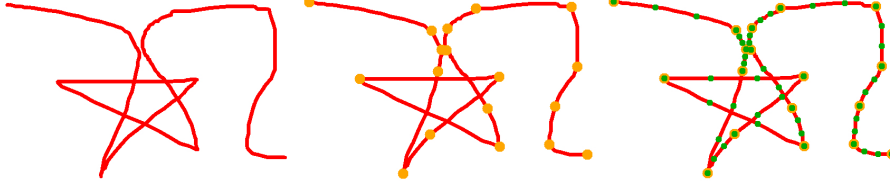


Fig. 1. Preprocessing of user-input motion: acquisition data (left); simplified motion with significant points over imposed (middle); final re-sampled motion (green points, right)

4 Multiscale Detection of Gesture Patterns

Let $G(s)$ and $\Gamma(s)$ be two curves sampled by arc-length s normalized in $[0, 1]$. Also, let $G = \{s_i/i = \overline{0}, \overline{m-1}\}$ and $\Gamma = \{s_j/j = \overline{0}, \overline{n-1}\}$ be two discrete samplings of the curves into m and n points. We pose the problem of finding the occurrences of gesture G in the longer motion Γ irregardless of scale.

4.1 A Naive Search Algorithm

A naive algorithm would choose every pair $p < q$, $p, q \in [0, n-1]$ from the Γ curve and employ a gesture recognizer R in order to get the matching result between G and the extracted part of Γ , $\Gamma_{[p,q]}$. The reported start location p' and scale $(q' - p' + 1)/n$ would be those for which the recognizer outputs the minimum distance (or maximum similarity):

$$(p', q') = \min_{0 \leq p < q \leq n-1} R(G, \Gamma_{[p,q]}) \quad (2)$$

The algorithm below illustrates this idea. Although the approach taken here is brute (search for all pairs $p < q$), there is no other option when aiming at scale invariance. Similar approaches in the literature perform this kind of multi-scale searches in order to achieve the invariance goal.

Algorithm 1. Naive-Detection(G, m, Γ, n)

```

1: for  $p = 0$  to  $n - 1$  do
2:   for  $q = p + step$  to  $n - 1$  do
3:     // compute distance between gesture  $G$  and part  $[p, q]$  of  $\Gamma$ 
4:      $distance \leftarrow Recognizer(G, \Gamma_{[p,q]})$ 
5:     if  $min > distance$  then
6:        $min \leftarrow distance, p' \leftarrow p, q' \leftarrow q$ 
7:     end if
8:   end for
9: end for
10: return  $p', q'$ 

```

The naive algorithm fails in practice due to over computations which cause big response times. The discussion from the Results section 5 gives an overview of the bad performances of this algorithm which do not meet the constraints of real-time interaction. For example, searching for a pattern inside a longer motion Γ sampled into $n \approx 100$ points returned a response in ≈ 167 ms, not acceptable when searching for multiple gestures.

4.2 The Gesture Recognizer

We use the elastic matching recognizer of Vatavu et al. [17] that computes the minimum alignment cost between the curvature functions of a gesture and a given template. The curvature signature function $\kappa(s)$ of a planar curve $C(s)$ parameterized by arc-length s fully prescribes the original curve up to a rigid motion transformation [3] which makes curvature suitable for gesture recognition. Figure 2 illustrates the alignment process between the curvature functions of two gestures. The complexity of the recognizer is $O(m \times n)$ where m and n are the sampling resolutions of the curves to be matched.

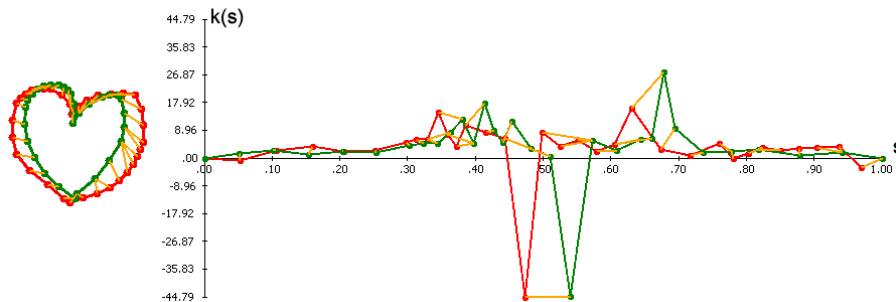


Fig. 2. Gesture recognition by measuring the cost of alignment between the signature function of a gesture (green) and that of a stored template (red)

4.3 Integral of Absolute Curvature

The Naive-Detection algorithm presented previously provides a good starting point for the scale-invariant detection problem if we set as goal filtering as many unnecessary intervals $p < q$ as possible. We build on top of the curvature representation of Vatavu et al. [17] and introduce the principle of summed area tables for the 1D case of the curvature signature function associated to a gesture motion. We thus define the *Integral Absolute Curvature*, $K(s)$, of a curve $C(s)$ parametrized by arc-length s :

$$K(s) = \int_0^s |\kappa(s)| ds \quad (3)$$

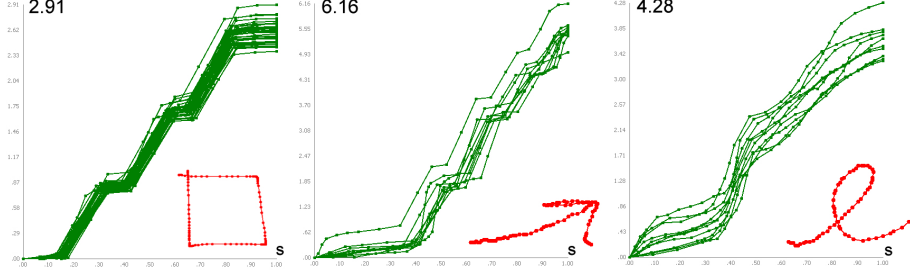


Fig. 3. Integral of absolute curvature for several gestures (multiple instances displayed overimposed): rectangle (left), arrow (middle) and pigtail (right)

Similarly, we can define the integral absolute curvature for a portion of a curve between arc-lengths s_1 and s_2 as follows:

$$K(s_1, s_2) = \int_{s_1}^{s_2} |\kappa(s)| ds = K(s_2) - K(s_1) \quad (4)$$

Figure 3 illustrates the integral curvature functions for a few gesture types.

The integral absolute curvature presents a few interesting properties as given by the following theorems:

Theorem 1. $K(s)$ is positively increasing.

Proof. Demonstration follows easily from the definition: $K(s+h) = \int_0^{s+h} |\kappa(s)| ds = \int_0^s |\kappa(s)| ds + \int_s^{s+h} |\kappa(s)| ds \geq K(s)$ for $\forall s \in [0, L]$ and $h \geq 0$. This means we can uniquely associate a given value $K(s)$ to an interval of arc-length Δs .

Theorem 2. $K(s)$ is scale invariant (Figure 4 illustrates the concept).

Proof. Let $c(t) : [0, T] \rightarrow \mathbb{R}^2, c(t) = (x(t), y(t))$ be a curve defined over time and let $\gamma(t) : [0, T] \rightarrow \mathbb{R}^2, \gamma(t) = \lambda \cdot c(t)$ be the scaled version of $c(t)$ with a given scale λ . If we re-parametrize each curve by its arc-length we obtain $c(s_c) : [0, L_c] \rightarrow \mathbb{R}^2$ with $s_c(t) = \int_0^t \|c'(u)\| du$ and $\gamma(s_\gamma) : [0, L_\gamma] \rightarrow \mathbb{R}^2$ with $s_\gamma(t) = \int_0^t \|\gamma'(u)\| du$. It follows immediately that $s_\gamma(t) = \lambda \cdot s_c(t)$ and $L_\gamma = \lambda \cdot L_c$ and in consequence we have $\gamma(s_\gamma) = \lambda \cdot c(s_c)$ where $s_\gamma \in [0, L_\gamma]$ and $s_c \in [0, L_c]$.

If we compute the curvature of $\gamma(s_\gamma)$ we get consecutively:

$$\begin{aligned} \gamma'(s_\gamma) &= \frac{d\gamma(s_\gamma)}{ds_\gamma} = \frac{d\gamma(s_\gamma)}{ds_c} \cdot \frac{ds_c}{ds_\gamma} = \frac{d(\lambda \cdot c(s_c))}{ds_c} \cdot \frac{1}{\lambda} = \frac{dc(s_c)}{ds_c} = c'(s_c) \\ \kappa_\gamma(s_\gamma) &= \gamma''(s_\gamma) = \frac{d\gamma'(s_\gamma)}{ds_\gamma} = \frac{d\gamma'(s_\gamma)}{ds_c} \cdot \frac{ds_c}{ds_\gamma} = \frac{dc'(s_c)}{ds_c} \cdot \frac{1}{\lambda} = \frac{1}{\lambda} \cdot c''(s_c) \text{ hence} \\ \kappa_\gamma(s_\gamma) &= \frac{1}{\lambda} \cdot \kappa_c(s_c) \text{ where } s_\gamma \in [0, L_\gamma] \text{ and } s_c \in [0, L_c]. \end{aligned}$$

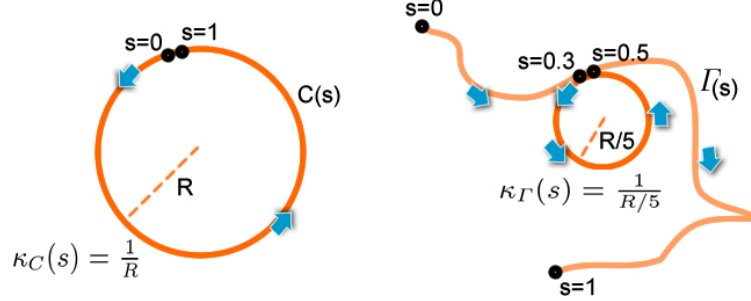


Fig. 4. The left circle $C(s)$ has been scaled down with a factor of 1:5 and inserted in the longer motion $\Gamma(s)$ from the right at location $s = 0.3$. The arrows show the direction of motion. Both curves are normalized with respect to arc-length hence the length of the downsized circle becomes 0.2. While curvature is not scale invariant: $\kappa_C(s) = \frac{1}{R}$ and $\kappa_\Gamma(s) = \frac{1}{R/5}$, the integral absolute curvature is: $\int_0^1 \frac{1}{R} \cdot ds = \int_{0.3}^{0.5} \frac{1}{R/5} \cdot ds = \frac{1}{R}$.

As for the integral of absolute curvature for $\gamma(s_\gamma)$: $K_\gamma(L_\gamma) = \int_0^{L_\gamma} |\kappa_\gamma(s_\gamma)| ds_\gamma$ and substituting $s_\gamma = \lambda \cdot s_c$, $K_\gamma(L_\gamma) = \int_0^{L_c} \left| \frac{1}{\lambda} \cdot \kappa_c(s_c) \right| \cdot \lambda \cdot ds_c = \int_0^{L_c} |\kappa_c(s_c)| ds_c = K_c(L_c)$ hence the integral absolute curvature is invariant to scale changes.

The trapezoidal rule gives the integral absolute curvature in the discrete case:

$$K_q = \sum_{i=1}^q \frac{|\kappa_i| + |\kappa_{i-1}|}{2} \cdot (s_i - s_{i-1}) \quad (5)$$

$$K_{p,q} = \sum_{i=p+1}^q \frac{|\kappa_i| + |\kappa_{i-1}|}{2} \cdot (s_i - s_{i-1}) = K_q - K_p \quad (6)$$

The last equation shows that, having computed K_p for a given curve, the computation of $K_{p,q}$ of a candidate can be achieved with $O(1)$ complexity.

4.4 Scale-Invariant Gesture Detection Algorithm

The integral absolute curvature may be used in order to design rejection rules for a given candidate marked by indexes $p < q$ on the continuous motion. Given a set of training samples for a gesture pattern, we can compute the integral absolute curvatures and store the interval $[K_{min}, K_{max}]$ for that gesture type. This will lead to a very simple yet efficient rejection rule for the candidate p, q on the Γ curve, in accordance with Theorem 2 above:

Rule #1. Reject candidate (p, q) if $K_{p,q} \notin [K_{min}, K_{max}]$

The second more powerful observation relates to Theorem 1: any integral value K is associated to a unique arc-length interval Δs due to the monotonic ascending

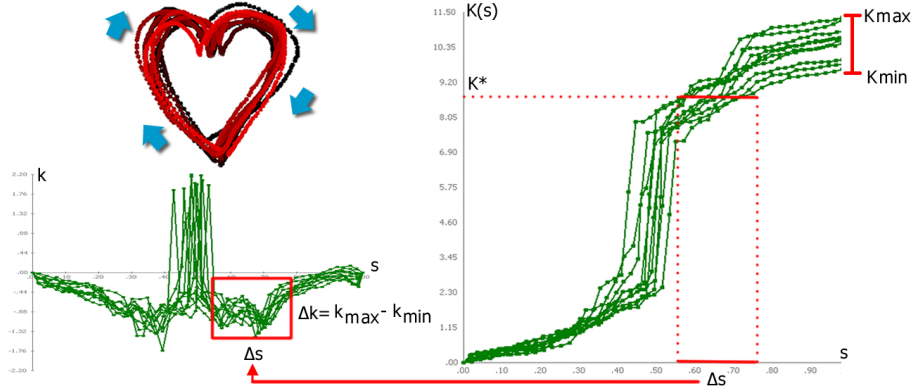


Fig. 5. Left: training set of heart-like gestures and their associated curvature functions. Right: integral absolute curvatures computed for the heart gestures. K_{min} and K_{max} as well as associations between a given K^* value and what the curvature at that s should be in terms of $[k_{min}, k_{max}]$.

property of $K(s)$. By correlating Δs with the curvature function $\kappa(s)$, lower and upper margins for curvature at K are obtained $\Delta k = \kappa_{max} - \kappa_{min}$ from the samples in the training set.

Rule #2. Reject candidate (p, q) if $\kappa_r \notin [k_{min}, k_{max}]$ for $\forall r \in [p, q]$ where k_{min} and k_{max} correspond to the value of $K_{p,r} = K_r - K_p$.

The two rules run in $O(1)$ and $O(q - p)$ time. Figure 5 gives a visual illustration.

5 Results and Discussion

In order to test the performance of our gesture detector we used the dataset of Wobbrock et al. [20] composed of 1,600 already segmented gesture samples = 16 types x 10 subjects x 10 executions for each gesture type at normal speed.

The 100 samples available for each gesture type were divided into training and testing giving sets of size $p \cdot 100$ for training and $(1 - p) \cdot 100$ for testing, where $p \in (0..1)$ was the training percentage. The training set was used to generate the rejection rules parameters. Each sample from the testing set was

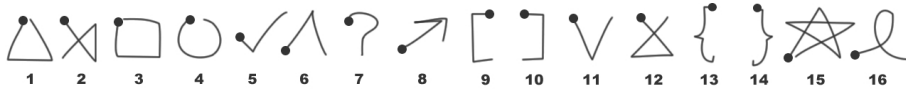


Fig. 6. The set of 16 gesture types of Wobbrock et al. [20]: triangle, x, rectangle, circle, check, caret, question-mark, arrow, left square bracket, right square bracket, v, delete, left curly brace, right curly brace, star, pigtail

Algorithm 2. SpeedUp-Detection(G, m, Γ, n)**Require:** $K[p]$ is the integral absolute curvature at index $0 \leq p < n$ **Require:** $\kappa_{min}[K]$ and $\kappa_{max}[K]$ are the lower/upper curvatures at integral value K

```

1: for  $p = 0$  to  $n - 1$  do
2:   for  $q = p + step$  to  $n - 1$  do
3:      $K_{p,q} \leftarrow K[q] - K[p]$ 
4:     if  $K_{p,q} < K_{min}$  then
5:       continue with next  $q$ , go to 2
6:     end if
7:     if  $K_{p,q} > K_{max}$  then
8:       continue with next  $p$ , go to 1
9:     end if
10:     $k[p..q] \leftarrow$  compute the scaled curvature of  $\Gamma$  for the  $q - p + 1$  scale
11:    for  $r = p + 1$  to  $q - 1$  do
12:       $K_{p,r} \leftarrow K[r] - K[p]$ 
13:      if  $\kappa[r] < \kappa_{min}[K_{p,r}]$  or  $\kappa[r] > \kappa_{max}[K_{p,r}]$  then
14:        continue with next  $q$ , go to 2
15:      end if
16:    end for
17:     $distance \leftarrow Recognizer(G, \Gamma_{[p,q]})$ 
18:    if  $min > distance$  then
19:       $min \leftarrow distance, p' \leftarrow p, q' \leftarrow q$ 
20:    end if
21:  end for
22: end for
23: return  $p', q'$ 

```

inserted at a random scale $\in [0.1 - 0.5]$ and at a random location in a randomly generated motion with the scale and location stored as ground truth. When generating random trajectories there is the danger that simple gestures such as *v* or *check* from Figure 6 are generated by chance at a different location than that of the inserted pattern which would affect the detection rate. To avoid this we proof checked each generated motion by running the naive algorithm in order to test if the pattern can be detected correctly. We varied the training percentage p from 10% to 90% of the available samples with increments of 10% (the smallest training set had 10 samples or 1 sample from each participant). For each testing set we computed the detection rate, start error, scale error, and execution time. The *start error* (e_{start}) represents the difference between the detected start position of a gesture compared with the ground truth (the exact position where the gesture was inserted) expressed as percentage of the motion length. The *scale error* (e_{scale}) is defined similarly. The detection rate equals the percentage of patterns successfully detected (for which $e_{start} < 0.1$ and $e_{scale} < 0.1$ but the average errors were less than 0.04 as we report below). In order to avoid biased results due to random sampling, we repeated 10 times each splitting procedure for a given p and averaged the results. We thus report results obtained from:

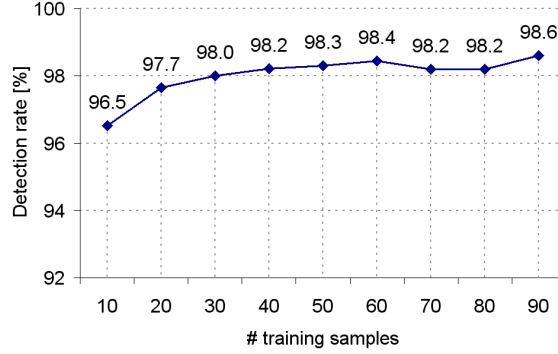


Fig. 7. Detection rate (%) vs. the number of samples in the training set ($p \cdot 100$)

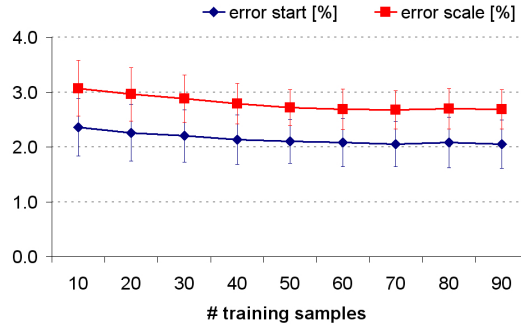


Fig. 8. Error rates (%) vs. the number of samples in the training set ($p \cdot 100$)

16 gesture types \times
 10 repetitions \times
 9 different testing sets ($p = 0.1$ to 0.9 , increment of 0.1) \times
 $(1 - p) \cdot 100$ generated motions per set =

$$= 16 \cdot 10 \cdot \sum_{p=0.1}^{p=0.9} (1 - p) \cdot 100 = 72,000 \text{ continuous motion trajectories.}$$

Figure 7 plots the detection rate vs. the size of the training set $p \cdot 100$. Even with a small training set consisting in only 10 samples (or 1 sample per participant) the detection rate is above 96% and raises up to 98% with a minimum of 3 samples per subject. The starting point error is approximately constant at 2% while the scale error is below 3% of the length of the motion trajectory as Figure 8 illustrates. Figure 9 plots the individual detection rates for each gesture type. Rates were averaged for all the trials $p \in [0.1, 0.9]$ and standard deviation are also presented. Except for the *left curly brace* gesture that averages a 94.1% detection rate, all the other patterns are detected with rates higher than 96.8%.

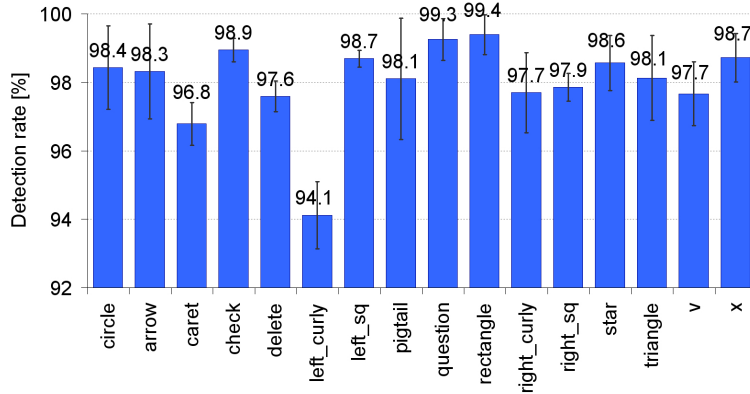


Fig. 9. Detection rate (%) vs. gesture type

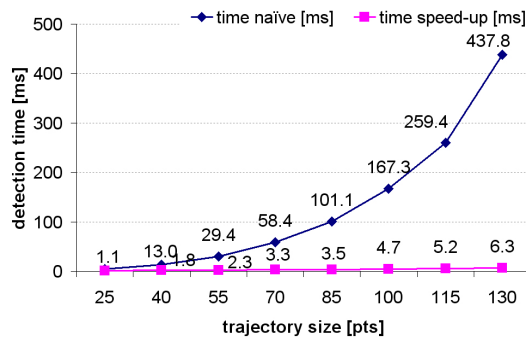


Fig. 10. Detection time (ms) vs. the size of the continuous motion (number of points)

The lower rate of *left curly brace* can be explained by its greater variation in execution: as Wobbrock et al. [20] mention, the participants disliked the curly braces as they felt "clumsy" drawing them. The standard deviation across all p has an average of 0.84% for all gestures with a maximum of 1.77% for *pigtail*.

The great advantage of the technique is presented in Figure 10 which illustrates execution time in ms versus the length of the continuous motion in points. In order to detect the full set of 16 gestures, the naive algorithm becomes impractical at a trajectory size of 70 points with 58.4 ms required per gesture while the speed-up algorithm executes in 3.3 ms. The discrepancy becomes even greater for longer trajectories: 437.8 ms for the naive versus only 6.3 ms for the speed-up at trajectories lengths of 130 sampled points. The measurements were performed on a 2.66 GHz P4 desktop computer.

6 Conclusions

We presented a new technique for the multiscale detection of gesture patterns in 2D continuous motions. The technique is fast due to several rejection rules that filter out most of the weak candidates. Detection rates above 96% were obtained for a large data set of 72,000 samples. We introduced the integral absolute curvature and showed its application in the discrete case. Our technique makes possible automatic segmentation of continuous motions without constraining users to segment their own gestures. As future work, it would be interesting extending the technique to 3D for which torsion next to curvature could be employed.

References

1. Arvo, J., Novins, K.: Fluid Sketches: Continuous Recognition and Morphing of Simple Hand-Drawn Shapes. In: ACM UIST 2000, pp. 73–80 (2000)
2. Arvo, J., Novins, K.: Fluid Sketching of Directed Graphs. In: 7th Australasian User Interface Conference, pp. 81–86. Australian Computer Society (2006)
3. Do Carmo, M.: Differential Geometry of Curves and Surfaces. Prentice-Hall, Englewood Cliffs (1976)
4. Cerlinca, T.I., Pentiu, S.G., Vatavu, R.D., Cerlinca, M.C.: Hand posture recognition for human-robot interaction. In: WMISI at ICMI 2007, pp. 47–50 (2007)
5. Dong, Q., Wu, Y., Hu, Z.: Gesture Segmentation from a Video Sequence Using Greedy Similarity Measure. In: ICPR 2006, pp. 331–334 (2006)
6. Hershberger, J., Snoeyink, J.: Speeding Up the Douglas-Peucker Line-Simplification Algorithm. In: Proc. of 5th Symposium on Data Handling, pp. 134–143 (1992)
7. LaViola, J.J.: Sketching and gestures 101. In: ACM SIGGRAPH 2007 Courses, p. 2. ACM, New York (2007)
8. Marcel, S.: Hand Posture Recognition in a Body-Face centered space. In: ACM CHI 1999 Extended Abstracts, pp. 302–303. ACM Press, New York (1999)
9. Moeslund, T.B., Hilton, A., Krüger, V.: A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding* 104(2), 90–126 (2006)
10. Pavlovic, V.I., Sharma, R., Huang, T.S.: Visual interpretation of hand gestures for human–computer interaction: A review. *IEEE TPAMI* 19(7), 677–695 (1997)
11. Poppe, R.: Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding* 108(1-2), 4–18 (2007)
12. Reng, L., Moeslund, T.B., Granum, E.: Finding Motion Primitives in Human Body Gestures. In: Gibet, S., Courty, N., Kamp, J.-F. (eds.) *GW 2005. LNCS (LNAI)*, vol. 3881, pp. 133–144. Springer, Heidelberg (2006)
13. Rubine, D.: Specifying gestures by example. In: Proc. of SIGGRAPH 1991, pp. 329–337. ACM Press, New York (1991)
14. Sowa, T.: The Recognition and Comprehension of Hand Gestures - A Review and Research Agenda. In: Wachsmuth, I., Knoblich, G. (eds.) *ZiF Research Group International Workshop. LNCS (LNAI)*, vol. 4930, pp. 38–56. Springer, Heidelberg (2008)
15. Schlomer, T., Poppinga, B., Henze, N., Boll, S.: Gesture Recognition with a Wii Controller. In: *TEI 2008, Bonn, Germany*, pp. 11–14 (2008)

16. Thorne, M., Burke, D., van de Panne, M.: Motion doodles: an interface for sketching character motion. In: ACM SIGGRAPH 2004, pp. 424–431 (2004)
17. Vatavu, R.D., Grisoni, L., Pentiu, S.G.: Gesture Recognition based on Elastic Deformation Energies. In: Sales Dias, M., Gibet, S., Wanderley, M.M., Bastos, R. (eds.) GW 2007. LNCS (LNAI), vol. 5085, pp. 1–12. Springer, Heidelberg (2009)
18. Vatavu, R.D., Pentiu, S.G.: Interactive Coffee Tables: Interfacing TV within an Intuitive, Fun and Shared Experience. In: Tscheligi, M., Obrist, M., Lugmayr, A. (eds.) EuroITV 2008. LNCS, vol. 5066, pp. 183–187. Springer, Heidelberg (2008)
19. Wilson, A.D.: Robust computer vision-based detection of pinching for one and two-handed gesture input. In: ACM UIST 2006, pp. 255–258 (2006)
20. Wobbrock, J.O., Wilson, A.D., Li, Y.: Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes. In: UIST 2007, pp. 159–168 (2007)