7th International conference on Intelligent Human Computer Interaction, IHCI 2015

A Framework to Plot and Recognize Hand motion Trajectories towards Development of Non-tactile Interfaces

Lalit Kanea, Pritee Khannaab*  

aPDPM IIITDM, Dumna Airport Road, Jabalpur (M.P.), 482005, INDIA

Abstract

This work demonstrates a real time framework to recognize trajectories articulated in the air using bare hand motion. A frontend is established to plot the trajectories as well as to spot the interleaved dynamic gestures. Finger detection based controls and trajectory plotting velocity help to spot the gesture boundaries. Trajectories are described through a unique Equi-Polar Signature (EPS) derived from circular grid normalization of trajectory points. EPS is invariant to translation, scale, rotation and stroke directions. k-Nearest Neighbor (KNN) classification strategy recognizes EPSs of digits 0-9 and operator symbols ‘+’, ‘-’, ‘x’, and ‘/’. Unlike previous path alignment algorithms, the proposed EPS scheme executes in linear time and fits to real-time constraints. On a customized depth video dataset of 2280 trajectories, 94.1% recognition accuracy is achieved.

© 2015 The Authors. Published by Elsevier B.V.

Keywords: dynamic gestures; trajectory recognition; finger detection; reference vector

1. Introduction

Hand gestures provide easy and natural way to communicate with humans as well as machines. Illustrative gestures are dynamic gestures articulated in hand motion trajectories when a letter, number, or shape is to be communicated instantly. A trajectory is a sequence of locations obtained by tracking a reference point on the hand during its motion. For correct interpretation of trajectories, the receiver must be in sync with the locations of the hand (space) which is tough for humans, if not impossible. A vision based computing system can accomplish the task effectively as space-time synchronization is not an issue. Applications of the presented scenario include non-tactile interfaces for interactive inquiry terminals in public transport, gesture based expression evaluators, smart TV controls, game

* Corresponding author. Tel.: +91-761-2632924; fax: +91-761-2632524.
E-mail address: pkhanna@iiitdmj.ac.in
controllers, and yet more. Realization of a Trajectory Gesture Recognizer (TGR) is challenging due to a number of reasons: (1) Gesture spotting is important because epenthetic strokes are associated as delimiters with the gesture, (2) TGR should be invariant to scale, translation, and rotation, (3) Recognition should not depend on the way strokes are made; clockwise, anticlockwise, or with differed beginning points, and (4) Correct segmentation of hand region is a challenge under cluttered background and odd illumination conditions.

This work develops a TGR based on Log-Polar Signature (EPS) derived from circular grid sampling of the trajectory points. Evaluated trajectory instances are shown in Fig. 1.

Fig. 1. Gesture trajectories recognized in the proposed system

2. Related work and contribution

The gesture frontend enables smooth acquisition of the trajectory and provides an interface to the subsequent representation. Frontend establishment is not trivial as points obtained may drift away from the actual trajectory due to imprecise depth measurements, trembling centroids due to the change of view-points and hand shapes. Gesture spotting in a continuous stream is the most important issue as epenthesis strokes always delimit the valid trajectory. Li and Zhang\(^1\) emphasized on gesture frontend by introducing fingertip detection and tracking technique using color and depth data while gesture spotting and inline recognition of trajectories were not addressed. Similarly, work by Zhang et al.\(^2\), Feng et al.\(^3\), and Lee and Lee\(^4\) also focus on gesture frontends only.

Most TGRs\(^5\)-\(^10\) process temporal sequences of trajectories. Intricacies like transformation and articulating direction invariance are hardly focused. In an HMM based strategy by Elmezain et al.\(^5\), Mean shift algorithm and Kalman filter were used to predict the hand location. A string of tangent angles defines the trajectory and gesture is spotted by negative to positive and vice-versa transitions of competitive differential observation probabilities. In other HMM implementations, Lee and Kim\(^6\) used motion chain-codes, and Min et al.\(^7\) used phase based velocity constraints analysis. In general, HMM based techniques require huge training and possibly the training of non-gestures (epenthesis strokes) too\(^5\). The Dynamic Time Warping (DTW) algorithm also processes temporal sequences by their exhaustive comparison. Doliotis et al.\(^8\) used 2D coordinates of hand locations normalized by square bounding box of constant size. DTW was further applied to match model and test coordinate strings. Sohn et al.\(^9\) acquired 3D trajectory signals from Asus XtionPRO™ depth sensor and used a combination of hand position, velocity, and acceleration features as input to DTW template matcher for comprehensive recognition performance. A variation of DTW, Dynamic Space Time Warping (DSTW), employed by Stefan et al.\(^10\) utilizes hand position and optical flow as a 4D input vector. Irrespective of its versions, DTW requires a number of model trajectories for performance enhancement and it also suffers from quadratic time complexity of template matching.

Longest Common Subsequence (LCS) algorithm is a successor to DTW, which matches common subsequences within two temporal strings. LCS is comparatively robust to noisy points as all points need not be matched and outliers can be ignored. Stern et al.\(^11\) identified the presence of unique segments in individual digit trajectories called Modified Discriminating Segments (MDS) and test streams searched for MDSs using modified LCS. Gesture spotting with an adaptive windowing mechanism is computationally complex. Frolova et al.\(^12\) introduced a probabilistic approach to track discriminative segments through LCS. However, the subgesture appearance which is a usual phenomenon in temporal sequence processing is not clearly focused. A normalized LCS (NLCS) technique introduced by Nyirarugira and Kim\(^13\) reduces the size of LCS matrix using orientation segments to make the scheme computationally efficient. Additionally, the use of rough set theory enhances subgesture reasoning. However, the 'pen down' and 'pen left' actions to spot the gestures are not exposed clearly. Finally, temporal sequence processing techniques are bound to the direction of stroke making and do not emphasize transformation invariances, especially rotation.

Proposed TGR presents a global approach which treats the trajectory as a curve rather than a temporal sequence. The curve processing mitigates rotation and scale variations through uniform reference vectors. Trajectories are described by a unique EPS feature which gives circular raster representation through precise angular and radial
Proposed TGR comprises essentially of two parts; a trajectory plotter (the frontend) and a recognizer or backend. The frontend development is detailed in Section 3 along with preliminary normalization and subsequent EPS representation. Section 4 details the dataset, classification, experimental setup and analyses results. Finally, Section 5 concludes the work with a discussion on shortcomings of the proposed TGR and further research.

3. Establishing a trajectory plotter (Frontend)

3.1. Hand segmentation and trajectory controls

Until a few years ago, vision based systems relied on 2D cameras where hand regions separated from background and face portion using skin color classifiers and various mixed color and gray intensity based schemes like calibrated thresholding\textsuperscript{14-16} and background subtraction\textsuperscript{17,18}. The deployment suffered as hand detection was imperfect due to unstable lighting conditions and false positives. Fortunately, with the availability of low cost depth sensors like Kinect\textsuperscript{TM} \textsuperscript{19} and XtionPRO\textsuperscript{TM} \textsuperscript{20}, hands can be segmented from other competing objects on the basis of depth information in the indoor environments at least. Proposed scheme solely utilizes the depth signals coming from Kinect\textsuperscript{TM} (Xbox 360) and establishes three depth bands as shown in Fig. 2(a) to facilitate trajectory drawing. Depth frame acquisition rate is set to 30 fps. Subject is allowed to articulate the trajectories in the middle (drawing) band. As long as hand segment moves in the drawing band, centroid of the hand region $O$ is plotted and stored in an array as the part of the trajectory. Assuming that hand being the closest object to the depth sensor, minimum depth $d_{\min}$ in the depth frame $D$ is determined as

$$d_{\min} = \min \{D(x, y) | 1 \leq x \leq x_{\text{max}}, 1 \leq y \leq y_{\text{max}}\}.$$  

where $x_{\text{max}} = 320$ and $y_{\text{max}} = 240$. Small noise chunks generating false $d_{\min}$ are removed by morphological open operation before probing for $d_{\min}$. A threshold $T$ on the depth is then established such that the depth segment $d_{\min}$ to $(d_{\min} + T)$ contains object of interest only. The binary region $B$ containing the object of interest is given as

$$B(x, y) = \begin{cases} 1, & \text{if } d_{\min} \leq D(x, y) \leq d_{\min} + T \\ 0, & \text{otherwise} \end{cases}$$

Explicit controls can be exercised over trajectory plots by the association of hand pose information. However, determination of hand posture followed by subsequent plotting action would be a costly process considering the fact that the operation is being carried out in a single thread of execution. A finger counting module is employed at this
juncture in place of comparatively costlier explicit shape recognition. Finger count controls and associated actions are depicted in Fig 2(b). Whenever the palm takes a fist shape, centroid plotting initiates and continues with hand movement as long as fist remains intact. Before the centroid of blob \( B(x, y) \), i.e., \( O = \{ (1/n \sum x, 1/n \sum y) \mid B(x, y) \geq 0 \} \) is plotted, its presence in the drawing band must be verified with \( d_{front} \leq D(O) \leq d_{rear} \) where \( d_{front} \) and \( d_{rear} \) are maximum and minimum depths of front and rear bands, respectively. Variations in hand movement speed introduce gaps in acquired trajectory points. Adjacent points \( O \) and \( O' \) are connected by line segments to render a smooth trajectory provided the gap \( \leq 0.1 \). As soon as hand enters the front or rear band, a cursor with pertinent color is displayed at the centroid to mark the drawing band violation. To stop the centroid plotting, an open palm (five fingers exposed) is signaled. Any intermediate finger count can be employed to interrupt plotting. In eq. (2), \( T \) is set such that even with slight out-of-plane rotation, a clear object of interest can be obtained (\( T = 100 \text{ mm} \)). No forearm separation technique is required to obtain the palm portion. Fingercounts are estimated through the endpoints on skeletons obtained from binary hand region \( B(x, y) \) that appears in drawing band. Parallel thinning algorithm by Zhang and Suen\(^{21} \) is applied due to its integer arithmetic, limited passes, and spur immunity. The excessive spurs which result in erroneous endpoints are pruned by a small predetermined value. Though it appears that the number of fingers exposed would be equal to the number of endpoints detected, an additional endpoint associated with wrist portion will always be present. Thus when hand is in fist shape or when one finger is exposed, in both cases two endpoints would be detected. Centroid plotting would, however, be kept on as long as endpoints are two. Six endpoints are interpreted as an open palm and signal an erase trajectory action. Four or five endpoints would stop centroid plot and give an opportunity to modify the trajectory and to trigger the recognition action. If a maximum inscribing circle is found on the palm and the number of endpoints is detected on the basis of a radial threshold, the process of finger counting can be more accurate but would impose a high computational overhead.

A window of frame length \( N = 5 \) is established to keep track of hand velocity \( v \). If \( v < 0.5 \), hand is considered almost stationary and recognition module executes. Once a gesture is spotted, trajectory points stored in the array are handed over to recognition module. Apart from user interrupt, contents of array are also flushed once the hand gets out of the drawing band. Note that finger controls can also work without drawing band establishment.

![Fig. 3. Maximum-radius line as reference vector in sample trajectories](image)

### 3.2. EPS representation

#### 3.2.1. Establishing a reference vector

The rotation normalization is achieved by the selection of a stable reference vector/line of reference. This vector is supposed to appear identical to all instances of the same gesture trajectory. Uniform articulation styles follow a common beginning point. The vector that connects the temporal beginning point to the centroid is termed as Temporal Start (TS) reference vector. To introduce the stroke direction invariance, Maximum-Radius (MR) line that connects trajectory centroid to the farthest point as shown in Fig. 3(a-c), is employed as reference vector. MR may also be generated as one of the two different reference vectors depending upon symmetry of the shape and articulation styles. Fig. 3 (d) shows alternative MRs for trajectory samples of ‘2’, ‘3’, and ‘8’. MR for a specific trajectory class is limited mostly in two orientations and can be assigned to two different subclasses belonging to the main trajectory class. The angle \( \theta \) between reference vector and positive x-axis is used to pursue angular adjustments. Other possible candidate reference vector choices are principal axis\(^{22} \) and axis of least inertia\(^{23} \).
3.2.2. Computing Equi-Polar Signature

EPS is a unique signature for contour outlines and curves which can efficiently capture the curve (trajectory) saliencies through precise angular and radial distance sampling. Trajectory $T$ is fitted into a circular raster of $m$ angular and $n$ radial divisions as shown in Fig. 4. Each equi-polar division is section of the circular raster which has a specified angular range $\theta_{\text{min}}$ - $\theta_{\text{max}}$ and a radial range $r_{\text{min}}$ - $r_{\text{max}}$. Cells in a two dimensional matrix $M$ correspond to equi-polar divisions of the circular raster. The aim is to find a representative point in each equi-polar section and then to record its radial distance in the corresponding cell of $M$.

Centroid $O$ and reference angle $\theta$ are computed first. Distance of the farthest point $R$ from $O$ is used for scale normalization. Algorithm begins with a point $P$ on the trajectory and keeps track of both, tangent angle $\delta$ (with respect to $\theta$) and distance $r$ from $O$ during the traversal of all $N$ points. As long as $\delta$ remains in the same angular range and $r$ remains within the same radial range, points are kept on added. As soon as angular or radial range violation is detected, added points are averaged to generate a representative point $P'$. The normalized radial distance of $P'$ is recorded in the corresponding cell $M(r_{-\text{idx}}, c_{-\text{idx}})$. Representative point is also determined when the last ($N^{th}$) trajectory point is reached or initial equi-polar section is crossed. Without checking against all angular and radial ranges, $r_{-\text{idx}}$ and $c_{-\text{idx}}$ can be determined directly using expressions in eq. 3.

\[
\begin{align*}
   r_{-\text{idx}} &= \left\lceil \frac{rn}{360} \right\rceil, \\
   c_{-\text{idx}} &= \left\lceil rn \right\rceil
\end{align*}
\]

where $r$ is min-max normalized distance. Normal radial division in circular raster does not capture information on outer circles efficiently because flat equidistant division of radius is not proportional to trajectory curve density. In this view, the number of divisions should reduce gradually with decreasing radius. The radial subdivision strategy adopted here divides $n^{th}$ radial division (from the centroid) into $n$ divisions. Value of $n$ between 3 and 5 is enough to capture the saliencies of trajectories. However after subdivision, $c_{-\text{idx}}$ would require modified formulation. Number of full divisions before $c_{-\text{idx}}$ is $n_c = \left\lceil \frac{r}{n} \right\rceil$ which contains $n_c = \frac{n_c(n_c + 1)}{2}$ subdivisions. With $k = (rn - n_c)(n_c + 1)$ denoting subdivisions up to $c_{-\text{idx}}$ after $n_c$, radial subdivision index is defined as $c_{-\text{idx}} = n_c + k$.

Fig. 4 shows EPS for trajectory of digit ‘5’ along with the matrix generated during traversal of the trajectory. Its one-dimensional representation in row major order generates EPS which is shown at extreme right. EPS can be formally defined as $\text{EPS} = \|I_{=1}^{m} M(r_{-\text{idx}}, c_{-\text{idx}})\|_{1}$ where $\|$ is concatenation operation.

Emergence of an alternate MR reference vector might affect the recognition of some trajectories. For instance, EPS for trajectories of digit ‘1’ would not vary if MR orientations are different. The trajectories of digit ‘2’ might have different MR orientations due to size and style of component semicircle and associated line’s stroke as shown in Fig 5. For symmetric trajectories like those of ‘3’ and ‘8’ alternate orientations are usually phase reversals. EPSs of such trajectories may vary depending on the extent of violation of symmetry. Fig. 5 illustrates that the EPSs for instances of ‘2’ quite differ. As trajectory of ‘8’ is symmetric in both axes, EPSs do not differ much. For trajectories like ‘2’ and ‘4’, alternate MRs are generated essentially due to writing styles, not because of symmetry imbalance. For sequential processing techniques, this is no more an issue but for non-temporal processing schemes like EPS, trajectory instances with different MRs for the same class must be handled carefully. Here, two MR instances of the same digit or symbol are considered as belonging to the two different classes.
4. Experimental setup and results

Eight signers were requested to freely articulate trajectories for digits 0 to 9 and signs ‘+’, ‘−’, ‘×’, and ‘÷’. The motive was to include various styles for each digit or symbol. Each symbol articulated 20 times giving a total of 160 symbols for each trajectory type generates 2240 trajectories. As EPSs are fixed length sequences, Euclidean distance based KNN classification fits for the recognition. The KNN classifier determines the symbol type by majority vote of \( k \) nearest EPSs. For comparison, DTW based template matching is evaluated which is a sequence of normalized 2D trajectory coordinates. DTW algorithm establishes several warping paths between a model and query trajectory sequence subject to boundary, temporal continuity, and temporal monotonicity constraints. The time complexity of exhaustive DTW search is reduced by dynamic programming. Three experiments are conducted; subject dependent evaluation on isolated and half-split samples, subject independent evaluation (on isolated samples with leave-one-out strategy), and recognition of symbols in continuous stream. A manually precut dataset of isolated gestures is used to train the system. All primary experiments are conducted with TS reference vector. Digit ‘1’, and symbols ‘−’ and ‘÷’ share the same trajectory with the only difference being in their orientations. Similar is the case with digits ‘6’ and ‘9’. For stroke order invariance, instead of clubbing the orientation feature with EPS, a rule based orientation detection strategy is adopted which is evaluated as a successive operation to EPS + KNN evaluation. A drift of ±20° is allowed in the basic orientation. Doing so refrains from weight assignments to individual attributes of the orientation clubbed feature vector.

In subject and stroke dependent evaluation, alternate trajectory samples are included in the training and testing sets. Experiments are conducted on varying grid sizes. Maximum recognition accuracy of 93.4\% is obtained on circular grid of size 16x6. Fig. 6(a) shows the confusion matrix for digits 0-9 and signs ‘+’ and ‘−’. Symbols ‘−’ and ‘÷’ are not included in the table as their accuracy is determined by the correct recognition of base symbols and orientation detection. Symbol ‘2’ confuses most to ‘3’ by 7.5\% while ‘2’ to ‘7’, ‘3’ to ‘2’, and ‘5’ to ‘6’ confusions are 5\%, 6.25\%, and 5\%, respectively. In confusion matrix of Fig. 6(b), accuracy of individual letters reflects the effect of variations in articulation styles. Subject independent evaluation, under a leave-one-out strategy, obtains 84\% accuracy. Impact on recognition performance with gradual increase in grid size is shown in Fig. 7.
For evaluation on a continuous stream, a gesture beginning is detected when hand extends and starts moving. Similarly when hand is pulled back, an end of the gesture is determined. However, the epenthetic boundary strokes (noise) may still be present in the resulting gesture which causes gesture misinterpretation. Recognition performance of EPS on continuous streams is shown with confusion matrix in Fig. 6(c). Symbol ‘3’ confuses most to ‘2’ by 10%. The overall accuracy achieved is 90.1%. An efficient frontend which separates noisy delimiters with the help of velocity constraints as mentioned in section 2.1 can avoid gesture misinterpretation in real time. On inclusion of trajectories for digit 9 and symbols ‘–’, ‘/’, recognition accuracies obtained for subject dependent, subject independent, and continuous stream evaluations are 94.1%, 85%, and 90.7%, respectively. The best subject dependent and stroke invariant recognition performance of 90.1% is obtained on a 24x3 grid. MATLAB R13a is used on a 32-bit Windows 7 platform for experimental evaluation and real-time testing.

Performance of DTW with normalized coordinates feature is 76.6%. A comparison on individual trajectory accuracies is shown using DTW and subject independent EPS in Fig. 7. Difference in the performances of EPS and coordinates based DTW is solely due to the effectiveness in dealing with rotated trajectory instances. However, if features sufficiently robust to rotation are employed with DTW, accuracy can be significantly improved. Due to diversity of the datasets employed in various TGRs, a direct comparison on recognition accuracies is difficult. Nevertheless, a qualitative comparison within various TGR works is given in Table 1.

Table 1. A comparison among various TGR work

<table>
<thead>
<tr>
<th>TGR Work</th>
<th>Trajectories recognized</th>
<th>Invariance to rotation</th>
<th>Invariance to stroke direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tangent angles + HMM$^\dagger$</td>
<td>Digits 0-9</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Normalized coordinates + DTW$^{8,9}$</td>
<td>Digits 0-9</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MDS-LCS$^{11}$</td>
<td>Digits 0-9</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NLCS$^{13}$</td>
<td>Digits 0-9, 22 Alphabets</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>EPS + KNN (Proposed)</td>
<td>Digits 0-9, Symbols +, −, ×, and /</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The DTW algorithm takes $O(mn)$ time to determine the best warping path for a model sequence of length $m$ against a query sequence of length $n$. EPS computation is $O(n)$ complex where $n$ is the number of trajectory points. On our dataset, the average time taken by DTW algorithm to generate a warping path is 0.52 seconds, while the EPS + KNN scheme takes only 0.023 seconds on an Intel 3GHz dual core processor with 4 GB memory.

5. Conclusions

Presented TGR framework follows match-with-the-flow strategy while avoiding complex matching mechanisms of temporal sequences. The most promising features of the proposed TGR are its acceptance to direction-free stroke
making and invariance to rotation. Due to unique calibration of trajectory points in angular and radial divisions it is possible to generate sequences of fixed length with EPS feature which enables matching of \( O(n) \). Gesture spotting is made comparatively intuitive through finger detection based controls and velocity constraints.

In this framework, finger detection based writing controls are used. Yet, a more intuitive interface with automatic finger or pen detection is needed followed by a qualitative or quantitative analysis of the same keeping in view the ease of usage\(^2\). Further trajectories of alphabets and various symbols also need to be evaluated and recognized. Recognition of trajectories containing interleaved shape information is an extension to trajectory recognizers which can be used in the sign language recognition.

References