International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Greedy Algorithm for Hand Drawn Object Recognition using BPMN

Phadtare Saurabh S[1], Balwant Sonkamble[2]

[1, 2]Pune Institute of Computer Technology, Affiliated to SPPU, Pune, Maharashtra, India

Abstract: ***With the fast growing technology of touch screen gadgets, finger stroke gesture interaction has become key buzzword in interactive HCI. The finger touch based gesture recognition can be quite promising task as it varies the human hand skills and behavior. A pen or finger touch gestures are more prominent and convenient than keyboards in all means of exchanging, writing human ideas or thoughts. The naturalness and user ease of a touch-based finger gesture interface offers new opportunities for human-computer interaction. Stroke gesture interfaces are responsible for making gestures analogous to physical effects, keeping gestures simple and distinct, defining stroke gestures systematically, making them self-revealing. Developers are becoming more concerned in creation of finger touch gesture recognizer which plays vital role in ubiquitous computing. Additionally, we hope our approach and findings will provide guidelines to develop a computer-aided system for automatic translating hand-drawn objects to Business Process Management notations. In this paper, the refined greedy algorithm is applied for recognizing Business Process Management notations symbols along with to get recognition accuracy measures. The refined greedy algorithm achieves superior performance than Dynamic Time Warping, the Rubine classifier based recognition, dynamic programming, as well.***

Keywords: gesture recognition; sketch; Business Process Management (BPM); Greedy heuristics; interactive User Interfaces, Business Process Modeling Notation (BPMN).

1. Introduction

Hand-drawn business object diagrams are a natural way to share business processing information between humans and enterprises. In evolvement of computer technology, various kinds of user input have been invented such as keyboards, mice, trackballs, etc., and recently, the primary mode of interacting with a computer is the keyboard, which is quite efficient, but not really comfortable design. A pen or finger touch gestures are more convenient than a keyboard for all the means of transform, writing is probably the most precise and comfortable, as well as the most flexible. Actually, Pen computing focused on a pen or a digital stylus as the implement of user's stroke gesture articulation. At the same time, the most of the well-known hand held devices companies (i-Pad, i-Phone) are focusing on finger-based interactions. With finger-based interactions there is no device acquisition time (the time to pull out and grasp the pen) so it is more direct and more convenient to use. On the other hand, the finger touch obscures more the screen surface and is less dexterous and precise than a pen. It is possible to enable both finger and pen interactions on the same device. Currently, smart mobile applications are spreading everywhere, starting with smartphones and tablets, to smart watches, and soon be found in other wearable, as well. However, developing and deploying for each separate hand held device platform can be an exhaustive task, especially if your resources are limited.

Object sketch recognition is a classification process in which unknown input sketch is compared to set of templates in order to find the best matching class. There are mainly two classification methods widely used: structural and statistical. In existing, there are mainly three types of touch-based interaction, such as a. raw ink, that records the raw freehand sketching data, e.g. a handwriting notepad or a paint application [1]. b) Direct manipulation, which manipulates the interface when a real-time feedback is needed from the system to the user such as scaling, panning, etc. [2, 3, 5, 6]. 3) Indirect command, where an input gesture is treated as a shortcut for triggering corresponding action Computer assistance in the completion of graphical symbols can be advantageous for the users in several applications. Our system can take advantage of unique touch features of users such as finger pressure and trajectory, the speed an acceleration of movement.

While using finger based gesture recognition in a Business Process Modeling Notation (BPMN) process diagram, a user can draw a circle which would depict a Start Event Activity, similarly a drawn diamond would infer a Gateway Activity, etc. Other behaviors also can be inferred from other user touch gestures, such as a 'fast scribble back and forth' gesture could infer an area in which diagram objects should be deleted (thus mimicking use of an eraser). Compounding gestures can be used to elaborate further on existing diagram objects for example; some gesture drawn on an existing diagram object might set its sub-type etc.

The essence of the entire tool should feel natural to use. For instance it should consider how users use a finger to draw; some users may draw a square in one go, some users may draw it in four individual 'strokes', some may draw it clockwise and some counter-clockwise. If the tool can recognize a shape drawn in a variety of different ways then in it means that individual users will need little or no training or instruction to use the tool and will not need to draw in specific ways to interact with the tool.

This paper is structured as follows. Section II discusses the literature survey, section III describes design methodology to recognize Hand-drawn business process diagrams, and section IV describes the experimentation details with result. The paper concludes in section V.

2. Literature Survey

Pen computing, handwriting recognition, touch Gesture, shape, and sketch identification are widely known areas of research, and various techniques have been used, such as finite state machines, Hidden Markov Models (HMMs) [5], neural networks, feature-based statistical classifiers [6], dynamic programming [7], template matching, and ad hoc heuristic recognizers [3, 6]. These approaches have been used for both on-line and off-line recognition. Many gesture recognition methods are ill-suited for use in rapid prototyping. Sophisticated pattern matching algorithms like neural networks [6] require large set of training examples and are unsuitable when the gesture set may not be defined prior to use. Some recognizers do not represent gestures as strokes, but use non temporal geometric properties of the final drawn shape. A multi stroke recognizer needs to search for the minimal matching distance between Candidate points and Template points from all the possible permutable alignments. Actually, this indicates a well-known problem in combinatorial optimization which is called the Assignment Problem. Hungarian method is quite efficient method for solving such problems. Various researchers have contributed their insights regarding handwriting recognition according their working environment and handwriting stroke corpus. The '$ Algorithms' provide an enrich set of contributions for the multi-touch gesture recognition research community.

Tahuti tool has multi-stroke sketch identification framework for class diagrams in UML where users can use touch interfaces for drawing sketches [3]. Lisa Anthony, JayeNias et al., [4] have presented a set of design recommendations for touch and gesture interaction for children that is based on findings proven to be robust over two studies with nearly 70 adults and children of various ages. They have found robust evidence that children have more trouble successfully acquiring onscreen targets and having their gestures recognized than do adults, especially the youngest age group (7-10 years old). Based on their findings, they have suggested key design and implementation challenges of supporting children's touch and gesture interactions. Michael Schmidt and Gerhard Weber [5] have proposed probabilistic classifier for multi-touch gestures specified by users. It separates input into tokens; retrieves local features and applying a new method of sensor fusion under uncertainty are adaptive to broader application ranges. While the segmentation of an input into tokens and the retrieval of local features allows for discrimination of subtle differences, but introduces the problem of pairing tokens of input and templates. Wobbrock J. O. et al., [6] have presented easy, cheap, and usable almost anywhere in about 100 lines of code. To evaluate "$1 recognizer", they have conducted an experimental study on the 16 gesture types; they used 4800 pen gestures supplied by 10 subjects on a Pocket PC. Compared to Dynamic Time Warping and the Rubine classifier on user- supplied gestures, they demonstrated that "$1 recognizer" has 97% accuracy with only one loaded template and 99% accuracy with 3+ loaded templates. Martin Field, et al., [7] have shown that it is possible to train a highly accurate recognizer using only isolated shapes as training data which is also easier to group and label isolated shapes than complicated diagrams. Mainly, they've focused on digital logic diagrams from 24 college student subjects. SeargrBelongie et al., [9] have used Euclidean, affine, and regularized thin plate splines. They have demonstrated object, prototype based recognition with using nearest neighbor technique. It was simple and easy to apply on invariant 2-D, 3-D shapes of real world objects. Edward Slender, et al., [10] have composed drawing framework around a retargetable kernel which gives a general front end to on-line recognition of any iconic representation. The kernel is expertized with UML specific improvements to segmentation and UML specific glyph recognizers. It governs simple and intuitive facilities for user correction of segmentation and recognition errors; Messy handwriting could deal with challenges for stroke segmentation.

3. Multi-Stroke Sketch Recognizer System

Developer should consider several measures while designing stroke gesture recognizer viz. articulation time, indicative angle difference, axial symmetry and proportional shape distance, etc. Using shape recognition algorithms that track a user's gestures on some device (such as a tablet computer) it should be possible to detect when a particular shape is drawn and then store this into some known model for storage and later editing (for instance the standard XPDL process definition format). This would mean that the user need only use their finger (for instance, or a pen-device or mouse) in the same way as they would a pen when drawing on a white board. It also means that the diagram can be stored in an editable form that is a fully models the semantics of the diagram notation (rather than just storing 'picture of vector graphics' or 'bitmap').

Our sketch recognizer system characterizes the actual flow of multi-layer functionalities incorporated with it. Initially, touch based inputs can be captured through touch interfaces. User can draw any (vague) object diagrams on touch interfaces. Then, user have to apply certain parametric preprocessing techniques such as calculating mean of initial candidate points in order to locate centroids of drawn geometric shape; calculating angle of trajectories for further operations. User can also compute the convex hull of the set of input points thus collected, using Graham's algorithm. In general user can process the strokes once.

Multi-stroke sketch recognizer system is basically required for recognizing user touch gestures to operate command in BPM.

Major Functions of System:-

3.1Finger touch gesture i.e. input acquisition and Template preparation;
3.2Stoke processing;
3.3Geometric Feature selection using statistical structure algorithm;
3.4Classification of key-point features using Greedy search method;
3.5 Shape context matching & recognizer testing;
3.6 Shape transformation;
3.7 BPMN objects identification and conversion into XPDL process.

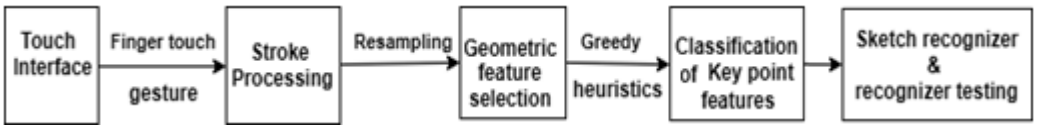

Figure 1: Sketch recognizer system

3.1 Input acquisition and Template preparation

Identify the touch input and categorize it into specific types; such as: touch gestures, Virtual typing and touch based drawing, etc. It's basically depends on touch controller i.e. sensing hardware and input software. It will sense finger gesture on touch device, interpret it; report the relative X coordinate and Y coordinate position of the finger stroke, and touch gesture direction.

3.2 Feature selection from shape based input (Stroke Processing)

After getting initial points termed as Candidate points (C) of drawn glyph, system can get raw gestures sampled at specific rate. Thus, writers' movement speed will have clear effect on the number of input points in an input gesture. In order to make gesture paths directly comparable even at different hand movement speeds, system first resample gestures such that the path defined by their original X points is defined by Y equidistantly spaced points. To resample, system first calculate the total path length of the X point set. Dividing this length by (Y–1) gives the length of each increment value, I, between Y new points. At the final step, the candidate gesture and any loaded Templates T_i will all have exactly Y points. This will allow system to measure the distance from C[k] to T_i[k] for k=1 to Y [7].

3.3 Geometric feature extraction and Training Data set

Existing"$1 recognizer" algorithm [11] searches over the space of possible angles for the best alignment between two point-paths. This system can find a gesture's indicative angle that is the angle formed between the centroid of the gesture$(\bar{x},\bar{y})$ and the gesture's first point. Then system can rotate the gesture so that this angle is at 0°. After that system can also scale the set of input points to the reference bounding box non-uniformly such as, transform the coordinates of all four corners:

$$x_1 = x_0+(x-x_0)*cos(\theta) + (y-y_0)*sin(\theta) \quad (1)$$
$$y_1 = y_0-(x-x_0)* sin(\theta) + (y-y_0)*cos(\theta) \quad (2)$$

Find the smallest of all four x's as x_min; find the largest of all four x's and call it x_max; similarly estimate it with the y's. Finally, bounding box is (x_min, y_min), (x_min, y_max), (x_max, y_ max), (x_ max, y_ min)

After scaling, the gesture is translated to a reference point. For simplicity, we choose to translate the gesture so that its centroid$(\bar{x},\bar{y})$ is at (0, 0).

3.4 Shape classification using greedy algorithm

To operate on normalized data sets, system can implement series of greedy heuristics for matching input sample point sets.

Initially, the Euclidean distances between all the points (let n) of input training set T_r and the n points of template T_t are computed and sorted in ascending order (using quick sort). The complexity of sorting feature list is $O(n^2 * \log(n))$. Now, for each point in the first candidate set (C_i), find the closest point from the second candidate that hasn't been matched yet. Once point C_i is matched, continue with C_{i+1} until all points from C are matched (i = 1..n−1). The complexity becomes $O(n^2)$ as for every point from C a linear search needs to be performed in T [11]. As result varies with the order of selected points C_i, this algorithm can be run multiple times with distinct starting points, finally return the minimal matching of all runs. In such case, if C_k is the initial point then:

$$\sum_i ||C_i - T_j|| = \sum_{i=k}^{n} ||C_i - T_j|| + \sum_{i=1}^{k-1} ||C_i - T_j|| \quad (3)$$

Where i goes circularly through all points in C. Inspired by Radu-Daniel Vatavu et al.,[11] this system will stick with the weighted sum of Euclidean distances, as follows:

$$\sum_i w_i \, . ||C_i - T_j|| \quad (4)$$

Here, weights w_i encode the confidence in each pair (Ci, Tj) computed during the greedy run. The first match is weighted with w1 = 1.0 (meaning high confidence) because generally the first point has all the data in order to make a decision for its closest match. As the algorithm proceeds, few options remain for the rest of the points from the first cloud when searching their closest pair into the second. System can be adopted a linear weighting scheme [11] in which:

$$w_i = 1 - \frac{i-1}{n} \quad (5)$$

Where i = (1...n) encodes the current step of the algorithm. This refined greedy heuristic algorithm runs in $O(n^{2+\varepsilon})$ time.

4. Experimental Setup and Results

This finger touch gesture recognizer system is designed using web technologies such as HTML5, JavaScript, CSS modules etc. We have implemented this Multi-stroke recognizer as Hybrid app, that can be deployed on cross platforms (i.e. Android, iOS, Windows, etc.) using Apache Cordova (PhoneGap), Ionic framework, etc. Hand drawn raw touch gestures are taken as Input to the system which comprises few English alphabets (Text identification), basic geometric shapes (sketches) as xml format. Initially, user drawn test gesture is matched with predefined geometric shape from Template, and then further it can be matched with trained standard BPMN shape repository samples. The overall performance of recognizer depends on the number of training samples (user-dependent testing scenario), the number of training participants (user-independent testing case) [14] and specific sampling rate, etc. For this experimentation, execution time is measured with sampling rate of n=96. Experiment is carried out on Intel ® Core™ i5 2.27 GHz laptop and multi-touch recognizer application (.apk file) is deployed on android based smartphone. Finally,

system's execution time is ten times quicker than Hungarian method based recognizer but bit less or similar efficient to "$N recognizer" system[16] owing to distinct sampling rates. The following results are obtained from implementation carried out as proof of concept. In this experimentation, we have taken 10 samples of 4 distinct personnel at different time. We repeated this process 5 times. So, recognition accuracy is governed by averaging the result values of 40 corpuses in user-dependent testing using series of greedy heuristics i.e. X. The outcome of these greed heuristics(X) depends on the direction of stroke matching (i.e. whether Candidate(C) point set is matched to Template(T) point set or vice versa), so finally their implementations must govern:

$$Min\ (Greedy\text{-}X\ (C, T), Greedy\text{-}X\ (T,C)) \quad (6)$$

Table 1: Comparison of recognition rate (%) between Greedy heuristics (X)

X	*Start Event*	*Connection*	*N (char)*	*Task Event*	*Timer Event*	*Message Event*
Greedy1	90	90.1	90.7	90.12	90	90.35
Greedy2	92.27	92.5	91	91.9	91	91.8
Average of Greedyalgo.s	94.6	93.4	94.2	94.6	93.2	93
Modified Greedy	98.4	97.65	97	97.9	95.4	95

Ultimately, this recognizer achieves comparatively high accuracy with optimized complexity for simple geometric shapes over tedious or curvilinear objects or shapes inside shape with respect to both user-dependent and independent testing.

5. Conclusion

The continual evolution of pen, wand, touch, tabletop, and surface computing techniques raises the necessity for user friendly creation of stroke based gesture recognition to facilitate rapid prototyping. Hence, this multi-stroke recognizer system is capable of producing a good recognition rates (around 98%) on noncomplex, minimum stroke articulated BPMN objects from BPM prototype. This is indicating that the system should be effective for smaller user interface symbol-sets. Implemented multi stroke recognizer system achieves a well balance between gesture recognition time and recognition accuracy during user dependent testing. Multi-stroke recognizer system generalizes from single multi-stroke templates to all possible stroke orders and directions, enabling each type of multi-stroke to be defined once. The overall design of system is lightweight and easy to understand for better usability. This system is capable to recognize rotation invariant single as well as multi stroke BPMN object models with optimized speed along with efficient accuracy measures. Effective use of this recognizer system cultivates naturalness and comfort in hand held device operations.

References

[1] Luis A. Leiva, Daniel Martín-Albo, Radu-Daniel Vatavu., Synthesizing Stroke Gestures Across User Populations: A Case for Users with Visual Impairments, In Proceedings of CHI '17, the 35th ACM Conference on Human Factors in Computing Systems, New York: ACM Press, 4182-4193, May 2017.

[2] Radu-Daniel Vatavu. et. al., Improving Gesture Recognition Accuracy on Touch Screens for Users with Low Vision, In Proceedings of CHI '17, the 35th ACM Conference on Human Factors in Computing Systems, New York: ACM Press, 4182-4193, May 2017.

[3] Martez E. Mott, Radu-Daniel Vatavu, Shaun K. Kane, and Jacob O. Wobbrock., Smart Touch: Improving Touch Accuracy for People with Motor Impairments with Template Matching, In Proceedings of CHI '16, the 34th ACM Conference on Human Factors in Computing Systems, New York: ACM Press, 1934-1946, May 2016.

[4] Anthony, L., Brown, Q., Nias, J. and Tate, B. 2015. Children (and Adults) Benefit From Visual Feedback during Gesture Interaction on Mobile Touchscreen Devices, International Journal of Child-Computer Interaction, Volume 6, p.17-27, December 2015.

[5] Anderson D., Bailey C. and Skubic M., Hidden Markov Model symbol recognition for sketch-based interfaces, In Proceedings of AAAI Fall Symposium, Menlo Park, CA: AAAI Press, pp. 15-21, 2004.

[6] ZhaoxinChen,EricAnquetil, Harold Mouchre and Christian Viard- Gaudin, A graph modeling strategy for multi-touch gesture recognition, In Proceedings of 14th International Conference on Frontiers in Handwriting Recognition (ICFHR-2014), Sep 2014.

[7] SriganeshMadhvanath, Dinesh Mandalapu, TarunMadan, NazninRao, and Ramesh Kozhissery, GeCCo: Finger Gesture-based Command and Control for Touch Interfaces, In Proceedings of IHCI. 2012, pp. 16, 2012.

[8] Tracy Hammond and Randall Davis, Tahuti: A geometrical sketch recognition system for UML class diagrams, In Proceedings of AAAI Technical report SS-02-08, pp. 59-66, 2002.

[9] Lisa Anthony, Quincy Brown, Berthel Tate, JayeNias, Robin Brewer, Germaine Irwin, Designing smarter touch-based interfaces for educational contexts, Personal and Ubiquitous Computing Journal, Springer, Volume 18, Issue 6, pp. 1471-1483, August 2014.

[10] Michael Schmidt, Gerhard Weber, Template based classification of multi-touchgestures, Elsevier Journal of Pattern Recognition, Vol no. 46, pp.24872496, 2013.

[11] Wobbrock J. O., Wilson A. D., Li Y., Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes, In UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology (New York, USA), ACM, pp. 159168, 2007.

[12] Anthony L., and Wobbrock J. O., A lightweight multi-stroke recognizer for user interface prototypes, In Proceedings of Graphics Interface '10, ACM, pp.245252, 2010.

[13] SeargrBelongie, Jitendra Malik, Jan Puzicha, et.al., Shape Matching and object recognition using shape contexts, IEEE Transactions on Pattern analysis and Machine intelligence, 2002.

[14] Edward Lank, Jeb S. Thorley and Sean Jy-Shyang Chen, An Interactive System for Recognizing Hand Drawn UML Diagrams, In CASCON '00 Proceedings

of the conference of the Center for Advanced Studies on Collaborative research, 2000.
[15] Radu-Daniel Vatavu, Lisa Anthony, Jacob O. Wobbrock, Gesture as Point Clouds: A $P Recognizer for User Interface Prototypes, In Proceedings of ICMI '12, ACM 978-1-4503-1467-1, pp. 273-280, 2012.
[16] Anthony L. and Wobbrock J. O., $N-Protractor: A fast and accurate multi-stroke recognizer, In Proceedings of Graphics Interface Conference '12, ACM, pp.117120, 2012.
[17] A. H. Toselli, A. Juan, J. González, I. Salvador, E. Vidal, F. Casacuberta, D. Keysers, and H. Ney, Integrated handwriting recognition and interpretation using finite-state models, Int. Journal of Pattern Recognition and Artificial Intelligence 18, no. 04, 519–539, 2004.
[18] S. Zhai, P. O. Kristensson, C. Appert, T. H. Andersen, and X. Cao, Foundational issues in touch-screen stroke gesture design-an integrative review, Foundations and Trends in Human-Computer Interaction 5, no. 2, 97–205, 2012.
[19] R. Plamondon and S. N. Srihari, On-line and off-line handwriting recognition: A comprehensive survey, IEEE Trans. on Pattern Analysis and Machine Intelligence 22, no. 1, 2000.
[20] Isokoski, P. Model for unistroke writing time. In Proc. CHI 2001, ACM Press, 357-364, 2001.
[21] Giovanni Dimauro, S. Impedovo and Giuseppe Pirlo, A new database for research on bank-check processing, 8th International Workshop on Frontiers in Handwriting Recognition, IEEE conference, February 2002.
[22] ShuminZhai, Per Ola Kristensson, Caroline Appert, Tue Haste Anderson, and Xiang Cao, Foundational Issues in Touch-Surface Stroke Gesture Design - An Integrative Review, Foundations and Trends® in Human–Computer Interaction: Vol. 5: No. 2, pp 97-205, Dec 2012.

Author Profile

Saurabh Phadtare is a post graduate student, pursuing Masters Degree in computer engineering fromComputer Dept. of Pune Institute of Computer Technology(PICT), affiliated to SavitribaiPhule Pune University, Pune. His research area includes Machine learning- Pattern recognition, Web Technologies, Human Computer Interaction and Open Source Technologies.

Mr. B.A. Sonkamble is a Professor in the Computer Department of Pune Institute of Computer Technology, affiliated to Savitribai Phule Pune University, Pune. He has successfully defended his Ph.D in Speech processing domain. His area of interest includesArtificial Intelligence, Machine Learning, Speech Processing, etc.