Modified Levenshtein Distance for Real-time Gesture Recognition

Clementine Nyirarugira, TaeYong Kim, Monson Hayes, Hyo-Rim Choi, and JunYoung Kim
Graduate School of Advanced Imaging Sciences
Chung-Ang University
Seoul, South Korea

***Abstract*—In this paper, a real time dynamic gesture recognition method based on a modified Levenshtein distance is proposed. The method addresses the issues faced in dynamic gesture recognition methods that are due to gesture variability within the class due to differences in the gesture speed. Since the gesture speed and duration will vary from one gesture to another, or from one person to the next, without taking this variability into account there will be an increase in the number of recognition errors. Here, we use a modified Levenshtein distance to represent each gesture class by one gesture exemplar and no restrictions on the gesture speed is necessary. Experiments demonstrate real-time gesture recognition rates of 93.9% for gestures of variable speed.**

I. Introduction

Gestures are a common part of normal human communication, and the development of many human-machine interfaces involve gestures. For example, recent consumer devices such as *Smart TV*s provide the user with an intuitive interface to control the TV through gestures. Hand gesture recognition today is an active area of research, primarily due to applications involving human machine interactions. Although a considerable amount of research has been done, a number of difficult challenges remain. One of these is the variability of the gestures within a given class. This variability may due to a number of different factors, such as gesturing speed, gesture size, and variations in the way a particular person forms a gesture.

This paper presents a gesture recognition method that is robust to gesture variability within the class. Specifically, it removes any constraint on the gesture speed, thereby making the interface between the person and the machine more relaxed and robust to variations in the speed at which the gesture is made. To achieve this, a modification to the Levenshtein distance measure is used to compare the codeword of a gesture that is to be recognized to those that are in a dictionary of gestures. The recognition system that is presented is able to efficiently recognize gestures of variable speed from a dictionary of fourteen gestures. Each gesture in our gesture vocabulary is represented by one exemplar that consists of a sequence of orientations codewords. To track the gesturing object, which is a hand in this paper, an adaptive evolution strategy of particle filtering is used, which is able to track the hand efficiently even when moving at variable speeds [1]. The experiment results show that the modified Levenshtein distance has a good recognition rate and can run in real-time at over 28 fps.

The remainder of this paper is organized as follows. Section II gives an overview of some relevant research in the field of dynamic gesture recognition. Section III presents an approach for variable speed gesture recognition that is based on a modification of the Levenshtein distance metric. Experimental results are then presented in section IV, and the conclusions and some directions for future work is given Section V.

II. Related Work

A widely used approach in gesture recognition systems is a Hidden Markov Model. HMM based gesture recognition methods represent each gesture by a set of states associated with probabilities (initial, transition, and observation) learned from the training examples [2]. Based on generalization of variability and the size of the training pool, these methods produce good recognition rate. HMM recognizers choose a model with the best likelihood and classify a given gesture as the corresponding reference gesture. A hidden Markov model-based continuous gesture recognition system for hand motion trajectory was proposed [3] to recognize Arabic numbers (0-9) in real time. The proposed HMM uses a left-right banded (LRB) topology with different state numbers ranging from 3 to 10, to handle isolated gestures. The left right banded model produced good recognition rate compared to other topologies. The end of a gesture is detected by detecting zero codeword in a continuous video stream. An HMM based threshold model approach for gesture recognition has been proposed [4]. The threshold model is trained on gesture and non gesture patterns and an adaptive threshold for selecting proper gesture model is derived. This adaptive threshold model increases reliability of gesture recognition rate. A gesture spotting network is used to find the start and the end of gestures embedded in the input stream. A sign language spotting with a threshold model based on conditional random fields (CRF) has been proposed [5]. Conditional random fields offer several advantages over hidden Markov models, including ability to relax strong independence assumption made in those models. A threshold model with CRF (T-CRF) is constructed to classify non sign pattern (which includes signs out of the vocabulary) contrary to other recognition methods that use a fixed threshold. The T-CRF model is trained on gesture vocabulary and non gesture vocabulary. Another approach to gesture recognition is the

analysis of similarity measures between gestures represented by time series. One of the possible measures of similarity between two strings is the analysis of their common substring. A technique for trajectory classification termed as most probable longest common subsequence for recognition of gesture character input have been proposed in [6]. A learning processing stage is performed to create a probabilistic 2-D template from equidistant points on gesture trajectory using Gaussian mixture model and associated probability in feature space, for each gesture, which allows taking into account different trajectory distortions with different probabilities. The MPLCS is designed to measure the similarities between the probabilistic templates and the hand gesture sample. The final decision is made based on the length and the probability of the extracted subsequence.

Dynamic time warping (DTW) is another approach often used for dynamic gesture recognition task [7]. The DTW based recognition methods attempts to line up a given sequence to gestures templates. It is an exemplar based matching procedure. Different distance metrics have been derived to produce better results. In 3D gesture recognition from one example [8], a statistical measure for calculating the distance between two aligned sequences has been proposed. Probability-based dynamic time warping and bag-of-visual-and-depth-words for human gesture recognition in RGB-D [9] used a soft distance based on the probabilistic similarity measure. The aforementioned derived distance measures improve the recognition rate of classical DTW that uses Euclidian distance as a cost function. To facilitate the incorporation of gestures into user interface, a gesture without libraries, tool-kits or training is the $1 recognizer for user interface prototypes [10]. The $1 recognizer is very simple, involves only the basic geometry and trigonometry, and uses only one representative example for each gesture class.

The Levenshtein distance is used for gesture classification in [7] where users are required to perform gestures at high speed. This produces gestures that have approximately the same length, which facilitate the classification process. In case there is no restriction made on gesturing speed, the classical Levenshtein distance would have poor performance. Therefore, we present a modification to the Levenshtein distance that can classify a gesture regardless of its size. This modification is shown to be equivalent to pruning of the codewords prior to recognition.

III. Gesture Recognition with Modified Levenshtein Distance

The general structure of the gesture recognition system presented in this paper is given in Figure 1. The key elements are the extraction of the position of the hand as a gesture is being made, the generation of a codeword that represents the complete gesture, a comparison of the gesture codeword to those that are stored in a gesture dictionary, and finding the one that closest to the given gesture to make a decision on the recognized gesture. These elements are described in further detail in the following sections.

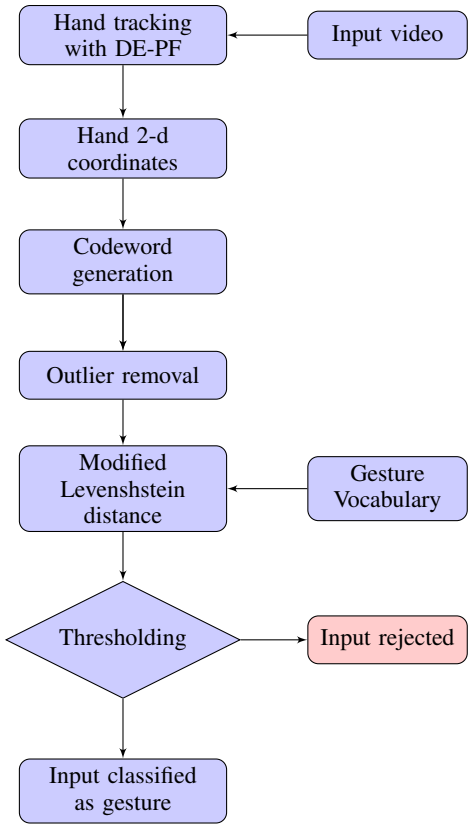


Fig. 1. Proposed Gesture Recognition Method Overview

A. Preprocessing and Gesture Representation

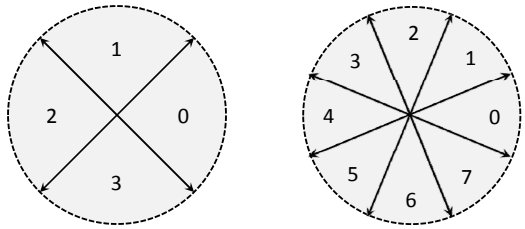
In order to build a gesture recognition system, it is necessary to have an input device that captures the hand movements corresponding to the gesture that is being made. Here, we use a USB 2.0 PC color video camera that captures 640×480 images. In order to recognize a gesture, it is necessary to detect the hand that is making the gesture and then track it throughout the entire gesture. There are many ways that have been proposed to track hands from color video (see, for example, [11], [12]). In this paper we use an adaptive evolutional strategy for particle filtering that will be referred to as DE-PF [1]. The output of the tracker is a set of 2-D hand coordinates $[x(t_n), y(t_n)]$ as a function of a discrete time index t_n. From these hand positions, the direction of hand movement is given by angle of motion as follows:

$$\theta(t_n) = \arctan\left(\frac{y(t_n) - y(t_{n-1})}{x(t_n) - x(t_{n-1})}\right) \quad (1)$$

These angles are quantized into one of N different angles,

$$g(t_n) = Q(\theta(t_n)) \quad (2)$$

where N is selected based on the complexity of the gestures that are to be recognized. Typically, it is expected that N would be an integer between four and sixteen. The case in which $N = 4$ and $N = 8$ is shown in Fig. 2. The sequence of quantized angles, $g(t_n)$, which we will refer to as codewords, forms the gesture *signature* that is to be recognized.



(a) (b)

Fig. 2. Codeword generation using (a) a four level quantizer for coarse gestures and (b) an eight level quantizer for finer gestures.

Fig. 3. Outlier correction.

Fig. 4. Digit gesture vocabulary.

Gesture	Codewords	
0	0 3 2 1 0	3 2 1 0 3
1	1 3	
2	0 3 2 0	0 3 2 3 0
3	0 3 2 0 3 2	0 3 2 1 0 3 2 1
4	3 0 1 3	
5	2 3 0 3 2	
6	2 3 0 1 2	
7	0 3	
8	0 3 0 1	0 3 2 3 0 1
9	2 3 0 1 3	2 3 0 1 2 3

TABLE I
CODEWORDS FOR THE GESTURES IN THE GESTURE DICTIONARY.

Due to errors in the location of the hand that is produced by the hand tracker, and due to jitter that might occur when the motion of the hand is along a direction that is close to the quantizer boundary, it is necessary to preprocess the sequence of codewords to remove these outliers and jitter. There are many approaches that may be used to deal with these errors, such as linear smoothing, median filtering, or an ad hoc approach that takes into account the type of outliers and noise that is expected. Here, the sequence of quantized angles is first processed to detect outliers that are to be removed. Given a sequence of codewords, a codeword $g(t_i)$ is identified as an outlier if the following two conditions are met,

$$|\left((g(t_n) - g(t_{n-1}) + 1)\right)_N - 1| > \alpha$$
$$|\left((g(t_n) - g(t_{n+1}) + 1)\right)_N - 1| > \alpha$$

where α is a threshold and $((a))_N$ is a modulo N. The process is repeated until all outliers are removed from the codeword sequence. Note that normal changes in orientation due to a gesture structure will not be detected as outliers. Once all of the outliers have been detected, they are then replaced with the closest *inlier*. An example is given in Fig. 3 that shows the outliers that were detected (those points that are circled at the top of the "3").

B. The Gesture Dictionary

Once a sequence of codewords for a given gesture that is to be recognized has been processed to remove outliers, it is compared to the codewords in a *gesture dictionary* and the one that is the *best match* is the recognizer output. In this paper, the gesture dictionary consists of ten digits, zero through nine as shown in Fig. 4, along with the four directional gestures, up, down, left, and right. We have performed experiments using $N = 4, 8$, and 16 quantization levels, and since there are only fourteen different gestures to be recognized, $N = 4$ worked the best and, as discussed later, results in faster processing of the gesture codewords. For the ten digits with $N = 4$, the codewords in the dictionary (the templates) are given in Table I. These codewords are defined in terms of single strokes in a given direction (possibly of variable length) that are used to form the given gesture. For example, the number "1" is formed by two orientations: one that progresses in a direction of approximately $60°$ (codeword of 1) followed by one that is downward at an angle of $-90°$ (codeword of 3). Note that in a few cases, two codewords are assigned to a single gesture to account for the possible variation in the way that the gesture may be made. For example, to allow for slightly different angles when beginning to form a gesture for the number "0" there are two codewords in the gesture dictionary for this entry. As another example, for the number "3" an extra codeword of 2 is inserted in the middle to allow for a *loop* in the middle of the gesture. Clearly, additional codewords may be added to account for additional variability in the gestures.

C. The Levenshtein Distance

The Levenshtein Distance (LD) is a metric that has been used to measure the difference between two strings of symbols or characters, and may be used to find the closest entry in the gesture dictionary to the gesture that is to be recognized [13]. The LD between two strings is equal to the minimum number of operations necessary to transform one string into another where the operations may be symbol insertion, deletion or substitution. Given a gesture $\mathbf{g}$ from the dictionary that is represented by an m directional codeword,

$$\mathbf{g} = [g_1, g_2, \ldots, g_m]$$

and a codeword $\mathbf{c}$ with n directional codewords that is to be recognized,

$$\mathbf{c} = [c_1, c_2, \ldots, c_n]$$

the Levenshtein distance between $\mathbf{g}$ and $\mathbf{c}$ is found from an $n \times m$ matrix $\mathbf{M}$ that is constructed as follows:

1) The elements in the first row and first column of $\mathbf{M}$ are defined by

$$\{\mathbf{M}\}_{i,0} = i \text{ and } \{\mathbf{M}\}_{0,j} = j$$

2) Then moving from left to right from one row to the next, the matrix entries are computed as follows.
   (a) If $c_i = g_j$ then

$$\{\mathbf{M}\}_{i,j} = \{\mathbf{M}\}_{i-1,j-1}$$

   (b) Otherwise,

$$\{\mathbf{M}\}_{i,j} = 1 + \min[\{\mathbf{M}\}_{i-1,j}, \{\mathbf{M}\}_{i,j-1}, \{\mathbf{M}\}_{i-1,j-1}]$$

Once all of the entries in $\mathbf{M}$ have been found, the Levenshtein distance is the element in the lower right hand corner of $\mathbf{M}$,

$$d_L(\mathbf{g}, \mathbf{c}) = \{\mathbf{M}\}_{n,m}$$

and corresponds to the minimum number of operations necessary to transform $\mathbf{g}$ into $\mathbf{c}$. This distance is computed between $\mathbf{c}$ and each of the codewords in the dictionary, $\mathbf{g}(k)$, and the recognized gesture is then $\mathbf{g}(k_0)$ where

$$k_0 = \arg\min_{g \in d} \{d_L(\mathbf{c}, \mathbf{g}(k))\}$$

provided that the distance between $\mathbf{c}$ and $\mathbf{g}(k)$ is below some threshold. If the distance exceeds this threshold, then the gesture is classified as *not recognized*.

An important indicator of how robust the gesture recognizer will be is the minimum distance between any pair of codewords in the dictionary. This will indicate how robust the classifier might be in recognizing gestures, and will indicate which gestures are most likely to be confused, and would suggest for which gestures more features should be added or that the gesture should be modified in some way. Shown in Table II are the distances between each gesture $\mathbf{g}(j)$ in the dictionary. Those that are in red correspond to a distance of one, which means that they are the gestures that are most likely to be confused with each other. These include the pairs (0,2), (1,7), (5,6) and (9,6) and it should be clear that the motions used to create these digits are very similar to each other.

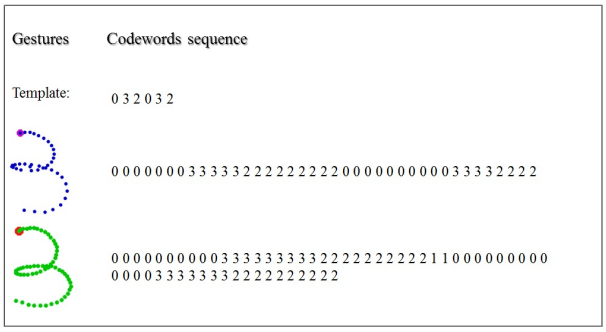
When testing the gesture classifier using the Levenshtein distance, one quickly observes that this distance will change as the speed of the gesture is varies, thereby making it more difficult to correctly recognize the gesture. For example, shown in Fig. 5 is an illustration of the effect of gesture motion on the Levenshtein distance for the gesture "three." The dictionary codeword for this gesture is

$$\mathbf{g} = [0,\ 3,\ 2,\ 0,\ 3,\ 2]$$

and the gesture sequences $\mathbf{c}_1$ and $\mathbf{c}_2$ for two different gesturing speeds are shown in the figure. Also shown are the lengths

	0	1	2	3	4	5	6	7	8	9
0	0	4	1	3	3	4	3	3	2	3
1	4	0	3	5	2	4	4	1	3	3
2	1	3	0	2	4	3	4	2	2	4
3	3	5	2	0	4	2	3	4	3	4
4	3	2	4	4	0	3	2	2	2	1
5	4	4	3	2	3	0	1	3	3	2
6	3	4	4	3	2	1	0	4	2	1
7	3	1	2	4	2	3	4	0	2	3
8	2	3	2	3	2	3	2	2	0	2
9	3	3	4	4	1	2	1	3	2	0

TABLE II
THE LEVENSHTEIN DISTANCE BETWEEN CODEWORDS IN THE DICTIONARY OF GESTURES.



	Length	Matrix size	Classic LD	Modified LD
Normal	39	11x39	33	0
Slow	61	11x61	55	1

Fig. 5. An example showing the details of the classical and the modified Levenshtein distance metric for gestures made at two different speeds.

of the codewords for each gesture, the size of the matrix needed to compute the Levenshtein distance, and the distances between $\mathbf{g}$ and $\mathbf{c}_i$ for $i = 1, 2$.

D. Modifying the Levenshtein Distance

Since it is difficult to control the speed of a gesture, and since any practical gesture recognition system should allow for some variations in gesture speed throughout the gesture, a gesture recognizer should be robust to these changes in speed and duration. In other words, the recognizer should only process the information related to the *path* that the hand makes. Since the Levenshtein distance between two gesture varies as the speed of the gestures change, it is necessary to modify the way that the distance is computed so that it is independent of gesture speed. A modified Levenshtein distance may be defined by modifying step 2 by adding 2(c) as follows:

   (c) If $c_i = c_{i-1}$ then

$$\mathbf{M}_{i,j} = \mathbf{M}_{i-1,j} \tag{3}$$

With this modification, repetitive codewords do not contribute the distance measure as shown in Fig. 5.

E. Codeword Pruning

Although the modified Levenshtein distance addresses the problem of having a distance metric that changes as the speed of the gesture to be recognized changes as illustrated in Fig. 5, the size of the Levenshtein matrix $\mathbf{M}$ will depend on the gesture speed. This is because the number of rows in $\mathbf{M}$ is equal to the number of codewords in the gesture $\mathbf{c}$, whose length depends on the gesture speed. Since minimizing the size of $\mathbf{M}$ will reduce the amount of memory and increase the speed of the gesture recognizer, a preferred approach would be to process the gesture codeword $\mathbf{c}$ prior to computing the Levenshtein distance. More specifically, since the key feature in the formation of a gesture is the initial gesture direction followed by the sequence of *direction changes* that are made throughout the gesturing process, it would be more efficient to *prune* the gesture codeword by removing repeated codewords and then use the classical Levenshtein distance. Therefore, given a raw codeword sequence that is produced by the hand tracker, the sequence is first processed to remove outliers, and then all runs of two or more identical codewords are replaced by a single codeword.

IV. Experiments

To evaluate the effectiveness of the gesture recognition algorithm presented in this paper, multiple gestures were recorded and processed by the recognizer. These gestures included the ten digits from zero through nine, and the four directional gestures of *up*, *down*, *left*, and *right*. Due to the simplicity of the directional gestures along with the coarse quantization of the gesture directions (four), these gestures were recognized easily without error. For the digits, 130 gestures were recorded, with between ten and twenty gestures for each. To evaluate the overall gesture recognition method, the recognition rate is defined as follows;

$$\text{Recognition Rate} = \left(1 - \frac{n_d}{N}\right) \times 100 \qquad (4)$$

where n_d is the number of misclassified gestures and N is the total number of gestures. For the ten digits, the experimental results are summarized in Fig. 6 for each digit, and the overall recognition results are shown in Table 1. In these experiments, the number of quantization levels was equal to four, and a threshold of $\alpha = 1$ was used (the threshold for outlier detection). Note that the gesture that is the most difficult to recognize is the digit "4." Since the distances between gestures given in Table II do not indicate that this gesture would be particularly difficult to recognize, the problem is probably associated with the wide variety in the way that a person may form the gesture for "4."

A second round of experiments was performed using a continuous video stream of multiple gestures. The objective was to get a meaningful sequence of gesture from a continuous motion of the hand. The end of each gesture was indicated by *freezing* the hand. Recognition results similar to those reported for single gestures was obtained.

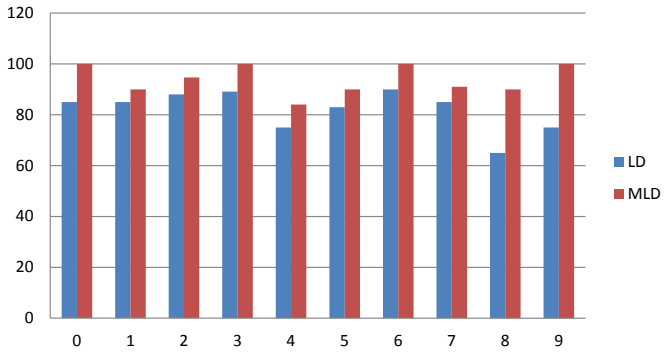


Fig. 6. Gesture recognition results.

Methods	Recognition Rate
LD	75.7%
Modified LD	93.9%

TABLE III
Gesture Recognition Rate

V. Conclusion and Future Work

This paper introduced a modified Levenshtein distance measure that is well-suited for variable-rate gesture recognition. Only one representative exemplar is used for each gesture in the gesture vocabulary. Although 2-D trackers will produce noisy data, the worst distortions are considered to be the ones that affect the produced orientation codeword provided that a gesture is represented as a sequence of orientation codewords. The proposed gesture recognition method uses outlier detection mechanism to correct them by replacing them by their closest inlier codewords. String matching algorithms are good candidates for dynamic gesture recognition (trajectory classification). However, they need to be adapted to the problem at hand. The modified Levenshtein distance is a good fit for gesture recognition systems that let users choose variable gesturing speeds, as a result of tolerating successive identical codewords produced by motion variability. The proposed method shows promising recognition rate and operates in real-time. Future work is to formulate an efficient gesture time segmentation method, gesture spotting. We have observed that missing the initial orientation codeword of a gesture is the source of many of recognition errors.

Acknowledgment

This research was supported by the Basic Science Research Program of the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (2013-009166).

References

[1] C. Nyirarugira and T. Y. Kim, "Adaptive evolutional strategy of particle filter for real time object tracking," *International Conference on Consumer Electronics (ICCE)*, pp. 37-38, January 2013.

[2] A. Jonathan, V. Athitsos, Q. Yuan and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1685-1699, September 2009.

[3] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis, "A hidden Markov model-based continuous gesture recognition system for hand motion trajectory," *International Conference on Pattern Recognition (ICPR)*, pp. 1-4, December 2008.

[4] H. K. Lee and J. H. Kim, "An HMM-based threshold model approach for gesture recognition," *IEEE Trans. Pattern Analysis and Machine Learning*, vol. 21, no. 10, pp. 961-973, October 1999.

[5] H. D. Yang, S. Sclaroff, and A. W. Lee, "Sign language spotting with a threshold model based conditional random fields," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1264-1277, July 2009.

[6] M. G Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Trans. Evolutionary Compotation*, vol. 15, no. 1, pp. 99-118, Feb. 2011

[7] S. Jeong, J. Jin, T. Song, K. Kwon, and J. W. Jeon, "A single-camera dedicated television control system using gesture drawing," *IEEE Trans. Consumer Electronics*, vol. 58, no. 4, pp. 1129-1137, November 2012.

[8] M. Sohn, S. Lee, D. Kim, B. Kim, and H. Kim, "3D hand gesture recognition from one example," *International Conference on Consumer Electronics (ICCE)*, pp. 171 - 172, January 2013.

[9] A. Hernndez-Vela et al., "Probability-based Dynamic Time Warping and Bag-of-Visual-and-Depth-Words for Human Gesture Recognition in RGB-D," *Pattern Recognition Letters*, 2013 (in press).

[10] J. O. Wobbrock, A.D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: A $1 recognizer user interface prototypes," *ACM journal of User Interface Software and Technology*, pp. 156-168, Octobr 2007.

[11] X. Zhang, W. Hu, M. S., X. Li and M. Zhu, "Sequential particle swarm optimization for visual tracking," *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-8, January 2008.

[12] Y. Wu, H. Ling, J. Yu, F. Li, X. Mei and E. Cheng, "Blurred target tracking by Blur-driven Tracker," *International Conference on Computer Vision (ICCV)*, pp. 1100-1107, November 2011.

[13] V.I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, 1966.