An Ant Learning Algorithm for Gesture Recognition with One-Instance Training

Sichao Song, Arjun Chandra and Jim Torresen
Department of Informatics, University of Oslo, Norway
{sichaos|chandra|jimtoer}@ifi.uio.no

***Abstract*—In this paper, we introduce a novel gesture recognition algorithm named the ant learning algorithm (ALA), which aims at eliminating some of the limitations with the current leading algorithms, especially Hidden Markov Models. It requires minimal training instances and greatly reduces the computational overhead required by both training and classification. ALA takes advantage of the pheromone mechanism from ant colony optimization. It uses pheromone tables to represent gestures, which scales well with gesture complexity. Our experimental results show that ALA can achieve a high recognition accuracy of 91.3% with only one training instance, and exhibits good generalization.**

***Index Terms*—Gesture recognition, accelerometer-data classification, ant colony optimization, ant learning.**

I. Introduction

Gesture recognition techniques have extensively been discussed in recent decades, and are expected to be the next-generation human machine interaction (HMI) solutions. Currently, there are various techniques being used for recognizing gestures, such as camera-based methods and sensor-based methods. With the popularity of hand-held devices, such as iPhone and iPod Touch[1], accelerometer-based gesture recognition for facilitating such interactions is becoming ever more pervasive and promising.

While the classical leading method, Hidden Markov Model (HMM), has proved to be effective for gesture recognition, it still has several shortcomings. HMM-based methods suffer from high time complexity for training and inference [1], [2], and their performance depends highly on the number of training instances [3]. Consequently, using HMM-based gesture recognition systems can be quite difficult in several scenarios. For example, it requires quite a long time and extensive physical work to build a gesture library with the necessary amount of gestures. As a result, user fatigue can greatly reduce the quality of training gesture instances, and lengthy training can be very tedious as well. Moreover, such systems could be weak in cases like real-time gesture recognition on low-end devices.

We note that dynamic time warping (DTW) is a very effective technique for recognizing accelerometer-based gestures [2], [4] and has extensively been studied recently. It requires low computational overhead and few training samples, which can be performed on a mobile device with promising performance. However, to the best of our knowledge, both the length and number of templates increase the classification time. Compared with DTW, classification in our ant learning algorithm requires much less processing time, since it only needs to calculate the distances between pheromone tables which have fixed sizes, independent of gesture types.

For addressing these limitations, we propose an ant learning algorithm (ALA) for gesture recognition. ALA aims at reducing both the number of training instances and computational overhead requirements, while still maintaining a high recognition accuracy. Our work has proven that ALA is a very effective technique for recognizing accelerometer-based gestures.

The rest of the paper is organized as follow: In Section II, we cover some of the related work in the field, which leads to the problem description. The methodologies and approaches are described in Section III. Section IV shows the experiment results, and we conclude our work in Section V.

II. Background

There are many studies on accelerometer-based gesture recognition systems which have used HMMs in their approaches [5], [6], [7], [8]. Pylvánáinen et al. in [5] used continuous HMMs to recognize gestures. In their work, they focused on demonstrating effects of data quality on recognition performance, which included both gesture sampling rate and vector quantization. Schlómer et al. in [6] estimated the performance of both left-to-right and ergodic HMM on 5 simple gestures, given the classic recognition pipeline. A Wii-controller[2] was used as the input device. In their work, they evaluated effects of codebooks[3] with different sizes $k = 8$, 14 and 18, respectively. They showed that, using a codebook with a size $k = 14$, their algorithm performed best. Based on their research, we choose the codebook size to be 14 for quantizing acceleration vectors.

For our own gesture recognition algorithm, ALA takes advantage of the pheromone mechanism from the Ant Colony Optimization (ACO) algorithm [9], [10]. ACO takes inspiration from the foraging behavior of ants in nature. Basically, ants wander randomly for finding food, while at the same time laying down pheromone trails on their respective paths, and these pheromone trails evaporate gradually. A path is more

[1]Apple Inc's products, see http://www.apple.com/.

[2]The handheld controller for Nintendo's Wii console, see http://us.wii.com/iwata_asks/wii_remote/.

[3]A codebook is a set of prototype vectors which are used for quantizing raw acceleration vectors.

attractive to ants if it has larger amounts of pheromone, the shortest (or optimal) path thus becoming the most attractive.

In ACO, the artificial ants deposit pheromone on their paths while moving, and a probabilistic model is used to decide future movements of these ants. Pheromone deposits on the paths happen according to:

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \forall i, j \in N, \quad (1)$$

and evaporates as:

$$\tau_{ij} = (1-\rho)\tau_{ij}, \forall i, j \in N, \quad (2)$$

where τ_{ij} is the amount of pheromone already present on the path linking states i and j, $\Delta\tau_{ij}^{k}$ is the amount of pheromone laid by ant k on the path connecting i and j, m is the population size of the colony, N is the set of states, and ρ is the evaporation rate.

While the computational model of ACO is primarily used for solving optimization problems, it can be adapted for gesture recognition as well, as will be described in Section III-C.

A. Problem Description

While there are popular methods for gesture recognition systems at present, certain limitations of these algorithms have not been satisfactorily addressed in the literature. These limitations are:

- HMM suffers from high time complexity for both training and inference, making it inefficient with large gesture libraries.
- HMM requires a sizable amount of training data. It needs quite a long time, and much physical work, if many gestures are required. The side-effect of this is that the quality of training gesture samples can be greatly reduced due to user fatigue.
- The processing time of DTW depends on both the length and number of templates.

Our algorithm aims at addressing these issues.

III. Methodologies and Approaches

An iPod Touch 4th generation is used to sense the acceleration produced by the hand motions of the user. Each acceleration vector is transformed into OSC[4] packets and sent over Wifi to a computer running the ALA. These vectors are mapped on to characteristic states which represent prototype vectors in the codebook. The characteristic states are selected as consecutive integers ranging from 1 to 14 in this paper since a codebook of size 14 is chosen. Each acceleration vector is mapped to the nearest state based on Euclidean distance, details of such quantization further described in Section III-B.

[4]Open Sound Control (OSC), see http://opensoundcontrol.org/.

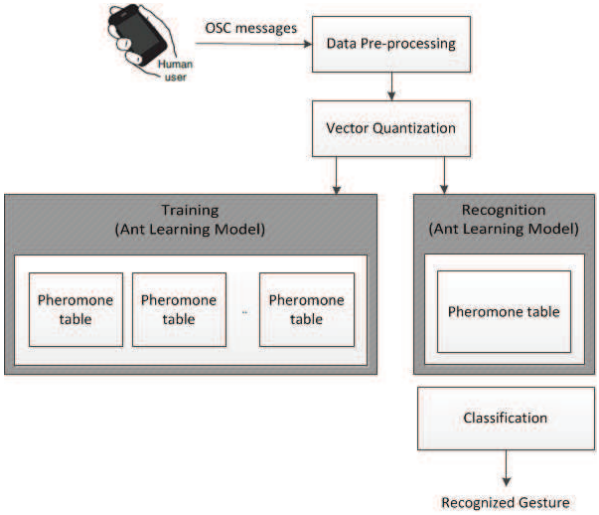

Fig. 1. System operational pipeline for the ant learning algorithm.

The magnitude of each acceleration vector i can be computed using the 3 values which represent acceleration magnitudes along the x, y and z axes respectively as:

$$acc^{i} = \sqrt{{acc_x^i}^2 + {acc_y^i}^2 + {acc_z^i}^2} \quad (3)$$

The vector i is then normalized before quantization as $acc_x^i = \frac{acc_x^i}{acc^i}$, $acc_y^i = \frac{acc_y^i}{acc^i}$, and $acc_z^i = \frac{acc_z^i}{acc^i}$.

In accelerometer-based gesture recognition, every single gesture is represented by a sequence of characteristic states. Consequently, a system operational pipeline is needed to pre-process input gesture data for both training and recognition purposes. Our system pipeline is shown in Fig. 1. It consists of 4 main components: data pre-processing, vector quantization, ant learning model, and classification.

Once a user starts to perform a gesture, the iPod Touch continuously sends OSC packets over Wifi. The packets are received in a laptop and unpacked to get the acceleration data. After pre-processing and quantization, the gesture is then represented by a sequence of characteristic states. If the gesture is performed for training, a new pheromone table (representing characteristic state transitions) labeled with the corresponding gesture type, is produced by the ant learning model and added in the gesture library for future recognition. If the gesture is performed for recognition, first a new pheromone table corresponding to the gesture is produced. This new table is used in conjunction with the tables in the gesture library by a classifier, giving a recognition result.

A. Data Pre-processing

Differing from HMM-based methods, a "directional equivalence" filter [11], [6], which eliminates all redundant vectors that are roughly equivalent to their predecessors, is not required for ALA. On the contrary, it is very important for ALA to record the number of immediate repetitions of all characteristic states in a gesture, for both training and classification.

In order to eliminate noise that may adversely effect the recognition result, an "idle" filter is used. We set $\Delta = 0.6g$ empirically, where g represents the acceleration due to gravity. It filters out all the acceleration vectors $\vec{a}$ for $|\vec{a}| < \Delta$. With the idle filter implemented, the user can make pauses during the performance of a gesture.

B. Vector Quantization

A vector quantizer maps each incoming acceleration vector onto a finite set of characteristic vectors, which is known as the codebook. In accordance with Schlömer [6], we choose a codebook with size $k = 14$, see Fig. 2.

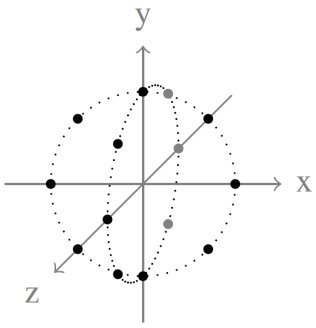

Fig. 2. Distribution of 14 characteristic vectors. Integers from 1 to 14 are assigned to each of them as per [6].

During quantization, an acceleration vector i is mapped to the closest characteristic vector with a state s^i based on Euclidean distance

$$s^i = \arg\min_n \sqrt{(acc_x^i - acc_x^n)^2 + (acc_y^i - acc_y^n)^2 + (acc_z^i - acc_z^n)^2} \tag{4}$$

where state $n \in N$, N being the set of states $\{1,2,3,..,14\}$. As a consequence, the gesture is eventually represented by a sequence of characteristic states after quantization.

C. Ant Learning Model

Inspired by [10], [12], [13], we introduced the pheromone mechanism to our learning model. As compared to both HMM-based and DTW-based methods, the ant learning model is easier to implement and deploy.

In ALA, an artificial ant moves in the state graph which is shown in Fig. 3 and deposits pheromone on edges that link these states. It moves whenever a new acceleration vector is received, and a pheromone value $\Delta\tau$ is laid on the edge linking the previous vector (state) and the current input vector (state). Since we only use one artificial ant in our algorithm, pheromone deposits on the edges happen according to:

$$\tau_{ij} = \tau_{ij} + \Delta\tau, \forall i, j \in N \tag{5}$$

This is simpler as compared to equation 1. Pheromone evaporates according to equation 2, which is the same as in ACO. $\Delta\tau$ is set to a default value based on extensive experiments (see Section IV-A). Since the ant learning model is used for both gesture training and recognition, both $\Delta\tau_{training}$ and $\Delta\tau_{recognition}$

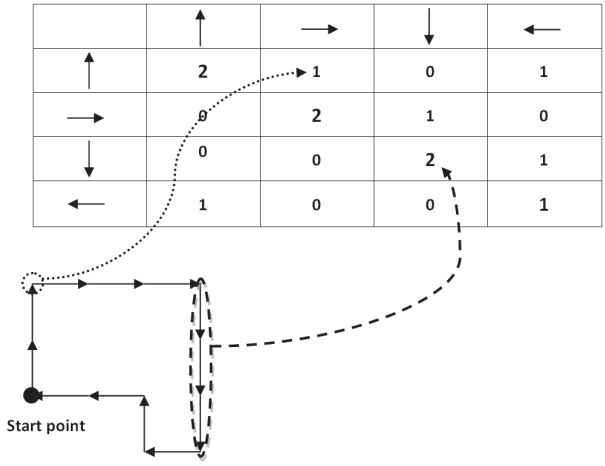

Fig. 4. The pheromone table is generated according to the incoming acceleration vector stream.

are allowed to be set separately. In this paper, we set $\Delta\tau_{training}$ equal to $\Delta\tau_{recognition}$.

The initial values in pheromone tables are set to zero. Once the algorithm starts to record a gesture, all values in the pheromone table are updated every time a new acceleration vector is received and quantized. To illustrate, Fig. 4 shows an example of a pheromone table with a codebook sized 4, which is produced corresponding to the acceleration vectors below the table. All the vectors follow a chronological order. The leading diagonal values show the number of immediate repetitions of each acceleration vector and the remaining values show the numbers of each transition point. The sequence of vectors starts from the indicated point and we assume that the initial state is $\uparrow$. A pheromone value $\Delta\tau = 1$ is added whenever the system receives a new vector. Each arrow listed in the first column represents the previously received vector and each arrow in the first row represents the currently received vector. The dotted circle shows a transition point between vector $\uparrow$ and $\rightarrow$, so it adds 1 in the corresponding cell in the table. The dashed ellipse shows a series of 3 vectors $\downarrow$. The first vector produces a transition point between $\rightarrow$ and $\downarrow$, so it does not update the pheromone value in the cell pointed to by the dashed arrow. After this, the system receives 2 more vectors $\downarrow$. As a result, a total value of 2 is added to the corresponding cell in the table.

Once the recording of the gesture is finished, both leading diagonal values and the remaining values in the table are normalized separately. We see this as a way of affecting generalization, since normalized values are transform-independent, meaning that gesture type will be represented by corresponding pheromone table, independent of its size and performing speed[5].

Each newly trained table is stored in the gesture library and available in the operational pipeline for recognition purpose. In this paper, values in a trained table will not be further modified

[5]The speed should be greater than a certain value, otherwise the input acceleration vectors will be filtered out by the "idle" filter.

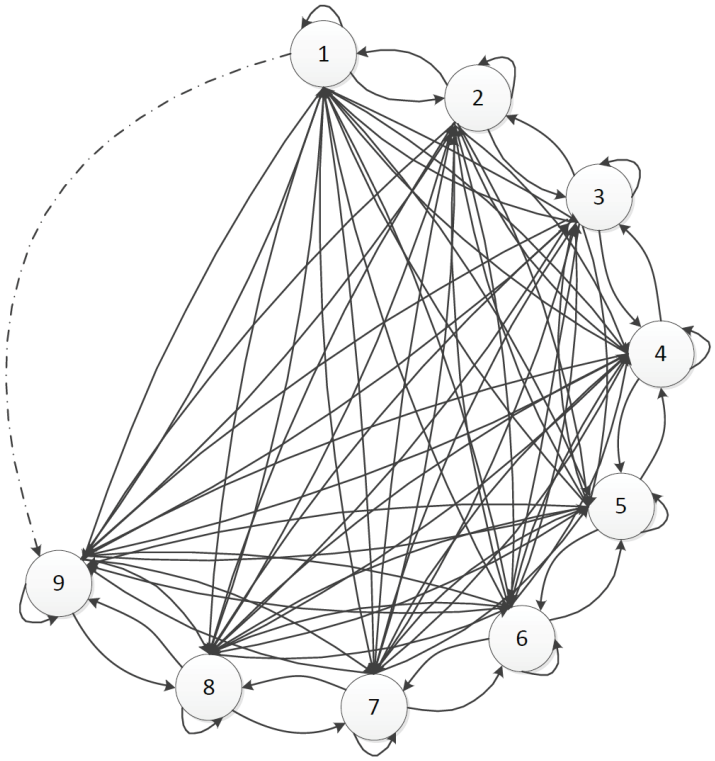

Fig. 3. Example of a graph with 14 states. States 10 – 14 are omitted for simplification purposes. The artificial ant can move from one state to another or stay in the same state.

once they are normalized. However, they can potentially be re-updated by a recognized gesture which is the same gesture type as the one that produced the pheromone table. This will make ALA an online gesture learning and recognition algorithm, and will be studied as part of our future work.

D. Classification

A new pheromone table is generated as soon as the user starts to perform a gesture. Once the gesture is done, a classifier based on Euclidean distance is used to match the new table to each of the trained gesture tables. Distance c^m is calculated as:

$$c^m = \sum_{i=1,j=1}^{k=14} (\tau^1_{i,j} - \tau^2_{i,j})^2 \tag{6}$$

where $\tau^1_{i,j}$ and $\tau^2_{i,j}$ represent the pheromone values in the corresponding two tables, the gesture that corresponds to the closest matching gesture table m, is the recognition result. In this paper, we set $m \in \{$1-Square, 2-Circle, 3-Triangle, 4-Eight, 5-Roll$\}$ according to the gestures used in our experiment.

We propose three classification methods in this paper:

- *Leading diagonal distance classifier:* It takes the leading diagonal values from the pheromone table into account. All other values are set to 0s. It examines the similarities between two gestures based on the number of immediate repetitions of all characteristic states according to equation 6 with $i = j$.
- *Rest-of-table distance classifier:* It evaluates all the values except for the leading diagonal values in the pheromone table. It evaluates the similarities with regards to the transition information in different gestures according to equation 6 with $i \neq j$.
- *Hybrid classifier:* It is a combination of the two methods whereby, both $i = j$ and $i \neq j$ are used.

The reason for considering the first two classification methods was to see whether or not gestures could be classified well based on minimal features (immediate repetitions or transition points) from within the incoming gesture data.

Each recognized gesture is mapped to a trained gesture type which has the shortest distance between them. Since we do not set any threshold for this distance, even gestures which have not been trained will still be recognized to the most similar trained gesture type. One advantage of doing so is that it provides tolerance for noise in gesture data. For example, it is not necessary to do a standard square in order to be correctly recognized, as long as the gesture has the shortest distance to the trained "Square" sample. As a result, ALA might have good performance in user-independent cases. This is out the scope of this paper but will be part of our future work.

According to the experimental results, the hybrid classifier leads to the highest average recognition rates. Thus, it is chosen as our classification method. The results can be seen in Section IV-A.

IV. Experiments and Results

In order to ensure platform independence, we implemented ALA and the algorithms (HMM and DTW) to which we compare ALA, in Java. These implementations were done within the NetBeans IDE 7.2.1 environment. A *Processing* library OSC P5[6] is used to transmit acceleration data from the iPod touch to a laptop via WiFi.

Five gestures are considered for the experiments, see Fig. 5. All samples of each gesture should be recorded or performed using the same iPod orientation.

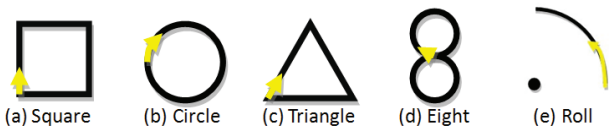

Fig. 5. Five simple gestures considered.

The starting point is flexible for all gestures except (e) only if an evaporation rate $\rho = 1$ is used. The effect of a $\rho < 1$ is that the acceleration vectors constituting the ending part of a gesture contribute more to the pheromone accumulated in the edges, as compared to those constituting the starting part. As a result, the values in the pheromone table for the same gesture type can vary due to different starting points.

Differing from HMM-based methods, only one training instance is required by ALA for each gesture type. Instead of generating several instances, one can immediately repeat the same gesture many times during one training. It makes the training less tedious and more efficient. We use a number of training instance t to represent the number of immediate repetitions of each gesture. Since we are interested in observing ALA's performance on training gestures with $t = 1$, we mainly focused on one-instance training and recognition in our experiments.

A. Parameter Tuning

As in ACO, different values of parameters have impacts on its performance, so it is important to find satisfying parameters. Similarly, ALA parameters need to be tuned first in order to guarantee a good overall performance. Main parameters of ALA are the following:

- Codebook size $k = 14$ (fixed in our case)
- Pheromone value $\Delta\tau$
- Evaporation rate ρ
- Classifier

The strategy for finding a satisfying combination of parameters is: Decide reasonable ranges and values within the ranges for both $\Delta\tau$ and ρ first, then test each pair of values for $\Delta\tau$ and ρ with different classification methods. Since the recognition results are more affected by the quality of gesture data rather than parameter settings, it is not necessary to test ALA on all possible combinations of $\Delta\tau$ and ρ. Thus, we only picked parameter sets with limited values distributed between certain ranges for both of them empirically.

The parameter tuning experiment was divided into 3 parts based on the different classification methods: leading diagonal distance classifier, rest-of-table distance classifier, and hybrid classifier. In each part of the test, the average recognition rates across pheromone value $\Delta\tau \in \{0.01, 0.05, 0.1, 0.5, 1, 2, 5, 10\}$ and evaporation rate $\rho \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ were examined. In order to keep the gesture samples consistent for all the tests, we first recorded the raw acceleration vectors during performing the gestures, and loaded them into the system operational pipeline afterwards. For the training dataset, each gesture was performed with different number of training instances $t \in \{1, 2, 5, 10, 15\}$; For the recognition dataset, each gesture was performed with $t = 1$. In the experiment, each gesture type was tested using 20 separately recorded gesture instances and a recognition rate was averaged over these. Both training and recognition trails were performed by the same person.

Table 1 shows results for ALA parameter tuning. Here we can see that ALA can achieve much higher average recognition rates with hybrid classifier rather than the other two classifiers. This is expected since the hybrid classifier uses both information of repetitions of each character states (leading diagonal values) and numbers of transition points (remaining values), which makes it generalise to different gesture types. As a result, we selected the hybrid classifier for ALA.

It is also noticed from the table that ALA achieved almost the same recognition results independent of both pheromone value $\Delta\tau$ and evaporation rate ρ. However, ALA tends to achieve higher recognition rates with lager $\Delta\tau$ and in-between values of evaporation rates. We assume that lower evaporation rates allow ALA to "forget" previously incoming acceleration vectors of a gesture, so they contribute less to the pheromone table as compared to the acceleration vectors which are received later. The side-effect of evaporation is that it makes the recognizer rely more on the latter part of gestures. This might strengthen the online learning ability of ALA since the newly received acceleration vectors could refine the pheromone table accordingly. We will evaluate the online learning ability of ALA in our future work.

According to the results, the final parameters of ALA are set as following:

- Codebook size $k = 14$ (it is fixed in our project)
- Pheromone value $\Delta\tau = 10$
- Evaporation rate $\rho = 0.5$
- Classifier=Hybrid classifier

Although the parameters are decided based on extensive tests, they are not optimal, since we did not exhaustively enumerate all the combination of $\Delta\tau$ and ρ in the experiment. Nevertheless, it should be feasible to adjust both $\Delta\tau$ and ρ according to the practical need.

[6]oscP5 is an OSC implementation for the programming environment *Processing*. It can be used for Java based projects.

TABLE I

RESULTS FOR PARAMETER TUNING: PHEROMONE VALUE $\Delta\tau$, EVAPORATION RATE ρ AND CLASSIFICATION METHODS. THE RESULTS ARE AVERAGED ACROSS 5 GESTURES: SQUARE, CIRCLE, TRIANGLE, EIGHT AND ROLL.

	ρ \ $\Delta\tau$	**0.01**	**0.05**	**0.1**	**0.5**	**1**	**2**	**5**	**10**
Leading diagonal Distance Classifier	**0.1**	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%
	0.3	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%
	0.5	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%
	0.7	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%
	0.9	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%	82.o%
Rest-of-table Distance Classifier	**0.1**	86.8%	91.4%	91.4%	91.6%	91.6%	91.6%	91.6%	91.6%
	0.3	86.8%	91.4%	91.4%	91.6%	91.6%	91.6%	91.6%	91.6%
	0.5	91.6%	91.6%	91.6%	91.6%	91.6%	91.6%	91.6%	91.6%
	0.7	91.6%	91.6%	91.6%	91.6%	91.6%	91.6%	91.6%	91.6%
	0.9	91.4%	91.4%	91.4%	91.4%	91.4%	91.4%	91.6%	91.6%
Hybrid Classifier	**0.1**	93.6%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%
	0.3	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%
	0.5	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%
	0.7	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%
	0.9	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%

B. Evaluation

The evaluation was conducted in three parts. In the first part, we used cross validation to observe the generalization performance of ALA, HMM and DTW, in the case of one-instance training. A comparison is given in table II. In the second part, we measured the execution time of the three algorithms with different number of training instances $t \in \{1,2,3,...,15\}$, averaged across all the five gesture types. Finally, the third part comprised of six tests performed at different time periods. The reason for doing this is that a person could be in different states during the day, or across days, both physically and mentally, which could affect the quality of gesture data.

Since it is impossible for a person to perform exactly the same two gesture samples, noise is inevitably included in the datasets. Thus, we first used cross validation to evaluate the generalization of our ant learning model with the chosen parameters, and compared ALA's performance with both HMM and DTW. HMM was implemented based on the Wiigee[7] library, and DTW was implemented based on [14].

For each gesture, we recorded 20 gesture instances. Of the 20 instances, a single recorded instance was picked as training data and the remaining 19 instances were used as the test data. This was then repeated 20 times, with each instance used exactly once as the training data. An average recognition rate (across the 19 test instances) was computed after each test run, and the final result was averaged over the 20 runs. The same recorded gesture samples were used for all the three algorithms.

Results in table II indicate that, among the three algorithms, DTW achieved the best recognition performance with all five gestures always being correctly recognized. While ALA achieved a lower recognition accuracy compared with DTW, it still showed a recognition rate of 98.58% averaged across the five test gesture types. Both ALA and DTW performed much better than HMM. AS a result, we show that both ALA and DTW have very good generalization performance with only one training instance, thus able to produce precise predictions of new gesture samples.

TABLE II

RESULTS FOR CROSS VALIDATION ON FIVE GESTURES: SQUARE, CIRCLE, TRIANGLE, EIGHT AND ROLL.

	ALA	**HMM**	**DTW**
Square	97.11%±10.73%	30.79%±26.14%	100.0%±0.0%
Circle	98.95%±3.66%	92.89%±9.7%	100.0%±0.0%
Triangle	96.84%±6.48%	81.32%±26.09%	100.0%±0.0%
Eight	100.0%±0.0%	68.95%±27.69%	100.0%±0.0%
Roll	100.0%±0.0%	60.0%±32.6%	100.0%±0.0%

From the cross validation results of ALA, we note that the quality of training samples might affect the recognition results significantly. We find that in some specific tests, the recognition rates are much below the average. For example, from the cross validation result of the gesture type "Square", we observed that for one of the training instances, ALA achieved an average recognition rate of 52.63%, which is much lower than the average 97.11%. This indicates that this particular instance could be very different from other instances. In fact, this training sample contained more noise than the average, thus was very inconsistent with the others. Using such instances as training data could lead to very high misclassification rates. Thus, gestures need to have some degree of consistency and such inconsistent gesture samples should not be used as training instances.

The execution time of the three algorithms was measured on

[7]A Java-based gesture recognition library for the Wii remote, see http://www.wiigee.org/. Based on this, we implemented a HMM-based gesture recognition system using the iPod Touch as input device.

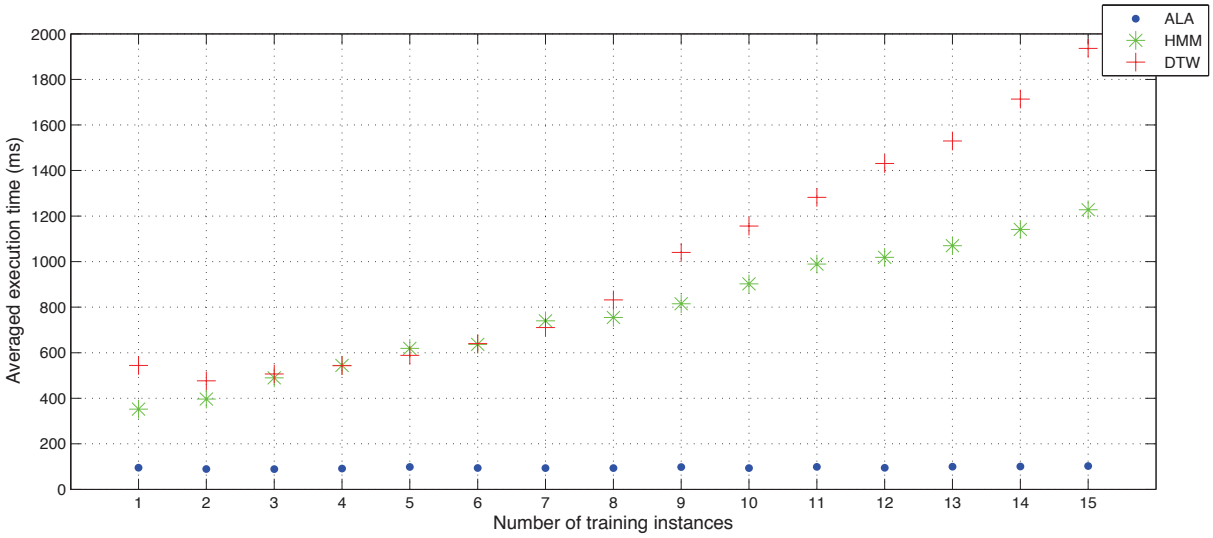

Fig. 6. Averaged execution time vs. Number of training instances, measured for ALA, HMM and DTW, respectively.

a Thinkpad E420 114139c laptop with an Intel(R) Core(TM) i5-2410M Dual-core processor (2.30GHz, 2.30GHz) with 4GB RAM. The test was performed on Windows 7 (64-bits). The execution time we measured consists of both training and recognition.

Each algorithm was tested on gestures with different number of training instances $t \in \{1,2,3,...,15\}$. For each number of training instance t, the execution time was averaged across all the five gestures.

Fig. 6 shows that, as compared to both HMM and DTW, ALA's averaged execution time was much shorter, and remained more or less the same, independent of the number of training instances. While DTW had an execution time similar to HMM when the number of training instances was below 7, it performed much slower than HMM when more than 8 training instances were used.

Thus, we can see that ALA requires extremely low computational overhead, and outperforms both HMM and DTW with regards to execution time.

Finally, in order to evaluate the performance of our system in practice, we conducted 6 separate tests. All the tests were carried out by the same person in different periods during 2 days: 3 tests were performed separately in the morning, afternoon and evening in the first day, and the other 3 tests were performed in the same periods during the second day, respectively. In each test, all the gesture types were re-trained before recognition.

The same 5 gestures as shown in Fig. 5 were used for the evaluation. 100 gesture trails were performed in each test with 20 trails for each gesture. As a result, in total 600 gesture samples were recorded in the experiment. An average recognition rate of 91.3% across the five gestures was observed. The average recognition rate for each gesture is shown in Fig. 7 and the average recognition rate for each test is shown in Fig. 8.

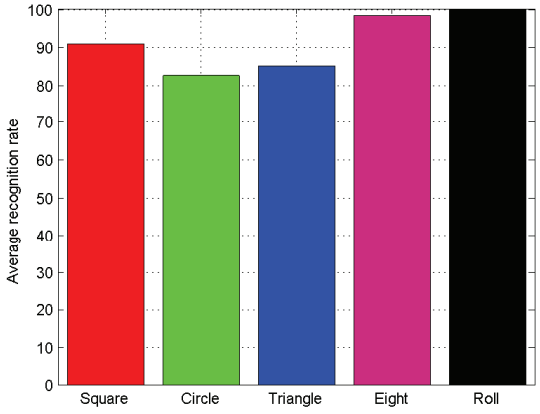

Fig. 7. Average recognition rates of the five gestures. The results were *Square* = 90.8%, *Circle* = 82.5%, *Triangle* = 85.0%, *Eight* = 98.3% *and Roll* = 100.0%, respectively.

Results in Fig. 8 show that the recognition rates can vary corresponding to different training samples, even though they are trained by the same person. Hence, the quality of training data can affect the recognition accuracy.

C. Discussion

The experiment results show that ALA can achieve an average 91.3% recognition rate with one-instance training. It indicates that ALA can recognize gestures with high accuracy while using minimal training data compared to HMM-based method. Thus, a user can easily build a large gesture library with many types of gestures and re-train gestures with much less effort. It should suits well to applications such as motion based music performance and video game playing.

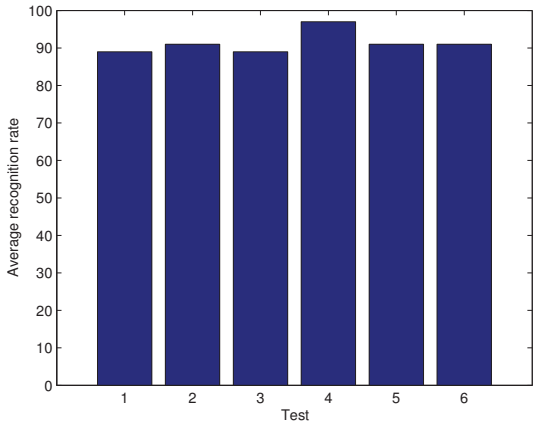

Fig. 8. Average recognition rates of the 6 tests. The results were 89.0%, 91.0%, 89.0%, 97.0%, 91.0% *and* 91.0%.

We can see from the cross validation experiment that ALA has good generalization performance. It outperforms HMM in the case of one-instance training. However, a bad training sample could significantly reduce the recognition rate. Thus, we should re-train the gesture if the current training sample leads to a very low recognition accuracy.

We show that ALA requires the shortest execution time among all the three algorithms. With the number of training instances ranging from 1 to 15, ALA's execution time remains more or less the same.

V. Conclusion

Accelerometer-based gesture recognition methods are expected to be exciting techniques for developing human machine interaction applications. In our work, we propose a novel algorithm based on ant colony optimization which requires minimal training effort and very low computational overhead. This is desirable for enhancing the interactive experience and make gesture recognition technique more pervasive. Moreover, devices with low computing power can also benefit from it and achieve real-time gesture recognition. A result of an over 90 percent recognition rate can be achieved, which is promising but leaves room for further improvement.

Future work could be observing how ALA performs in online learning cases. We plan to evaluate the recognition results on-the-fly, where trained pheromone tables are continuously updated during recognition. Besides, user-independent experiments will be carried out to see how ALA performs in those cases. It will also be very interesting to explore the applicability of ALA to pattern recognition problems other than gesture recognition.

Acknowledgment

The research leading to these results was conducted in the EPiCS project (Engineering Proprioception in Computing Systems) and has received funding from the European Union Seventh Framework Programme under grant agreement n^o 257906. http://www.epics-project.eu/

References

[1] S. Rajko, G. Qian, T. Ingalls, and J. James, "Real-time gesture recognition with minimal training requirements and on-line learning," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*. IEEE, 2007, pp. 1–8.

[2] G. Niezen and G. P. Hancke, "Evaluating and optimising accelerometer-based gesture recognition techniques for mobile devices," in *Proc. AFRICON (AFRICON'09)*. IEEE, 2009, pp. 1–6.

[3] L. Yin, M. Dong, Y. Duan, W. Deng, K. Zhao, and J. Guo, "A high-performance training-free approach for hand gesture recognition with accelerometer," *Multimedia Tools and Applications*, pp. 1–22, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11042-013-1368-1

[4] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657–675, 2009.

[5] T. Pylvänäinen, "Accelerometer based gesture recognition using continuous HMMs," in *Pattern Recognition and Image Analysis*, ser. Lecture Notes in Computer Science, J. S. Marques, N. Pérez de la Blanca, and P. Pina, Eds. Springer, 2005, vol. 3522, pp. 639–646. [Online]. Available: http://dx.doi.org/10.1007/11492429_77

[6] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a Wii controller," in *Proc. 2nd International Conference on Tangible and Embedded Interaction (TEI'08)*. ACM, 2008, pp. 11–14.

[7] F. Bevilacqua, B. Zamborlin, A. Sypniewski, N. Schnell, F. Guédy, and N. Rasamimanana, "Continuous realtime gesture following and recognition," in *Gesture in Embodied Communication and Human-Computer Interaction*, ser. Lecture Notes in Computer Science, S. Kopp and I. Wachsmuth, Eds. Springer, 2010, vol. 5934, pp. 73–84. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12553-9_7

[8] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and D. Marca, "Accelerometer-based gesture control for a design environment," *Personal and Ubiquitous Computing*, vol. 10, no. 5, pp. 285–299, 2006.

[9] M. Dorigo, "Learning and natural algorithms," Doctoral dissertation, Politecnico di Milano, Italie, 1992.

[10] A. Colorni, M. Dorigo, V. Maniezzo *et al.*, "Distributed optimization by ant colonies," in *Proc. 1st European Conference on Artificial Life (ECAL'91)*, vol. 142. Paris, France: Elsevier, 1991, pp. 134–142.

[11] M. Klingmann, "Accelerometer-based Gesture Recognition with the iPhone," Master's thesis, Goldsmiths University of London, London, UK, 2009.

[12] C. Guéret, N. Monmarché, and M. Slimane, "Ants can play music," in *Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Sttzle, Eds. Springer, 2004, vol. 3172, pp. 310–317. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-28646-2_29

[13] A. Hamdi, V. Antoine, N. Monmarché, and M. Slimane, "Artificial ants: from collective intelligence to real life optimization and beyond," 2010.

[14] N. Gillian, R. B. Knapp, and S. O'Modhrain, "Recognition of multivariate temporal musical gestures using n-dimensional dynamic time warping," in *Proc. 11th International Conference on New Interfaces for Musical Expression (NIME'11)*, 2011.