2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Ubiquitous Robotics in Physical Human Action Recognition: A Comparison Between Dynamic ANNs and GP

Theodoros Theodoridis, Alexandros Agapitos, Huosheng Hu, and Simon M. Lucas
Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, U.K.
{*ttheod, aagapi, hhu, sml*}*@essex.ac.uk*

***Abstract*— Two different classifier representations based on dynamic Artificial Neural Networks (ANNs) and Genetic Programming (GP) are being compared on a human action recognition task by an ubiquitous mobile robot. The classification methodologies used, process time series generated by an indoor ubiquitous 3D tracker which generates spatial points based on 23 reflectable markers attached on a human body. This investigation focuses mainly on class discrimination of normal and aggressive action recognition performed by an architecture which implements an interconnection between an ubiquitous 3D sensory tracker system and a mobile robot to perceive, process, and classify physical human actions. The 3D tracker and the robot are used as a perception-to-action architecture to process physical activities generated by human subjects. Both classifiers process the activity time series to eventually generate surveillance assessment reports by generating evaluation statistics indicating the classification accuracy of the actions recognized.**

I. INTRODUCTION

The investigation of human action recognition has been addressed by a number of researchers introducing various approaches to classify different physical activity patterns. The extremely flexible and expressive nature of programming languages to represent solutions to problems offers GP [1] the capacity to represent classification problems with means unavailable to other techniques such as decision trees, statistical classifiers, and ANNs [2].

In [3], a 3D gesture recognition scheme has been used to analyse the dynamics of a hand motion so that to classify manipulative and controlling gestures through an object-centred approach which computes 3D appearances using a region-based coarse stereo matching algorithm. The perceived gestures are modeled by a forward HMM and a neural network. A gesture modelling algorithm used by [4], implements an event-driven HMM which exploits sequences of events that take place within the body segments and joints to represent gestures. The advantage of this event-driven HMM is that it is independed from sequences of poses thus more complex gestures can be recognized. [5] presents an efficient Fourier transformation performed on the vertical axis of a cylindrical coordinate system used to robustly extract visual motion descriptors which are classified by distance-based methods.

Emphasis has been given to the issue of representation leading to the evolution of **(a)** decision tree classifiers [1], **(b)** classification rule-sets [1], and **(c)** numeric expression classifiers [2][6]. Similar to our work, [7] has used a dynamic programming algorithms to process 3D joint features so that to segment and recognize actions by improving the overall accuracy with a Multi-Class AdaBoost algorithm.

In our work, a dynamic ANN from the Matlab's NN toolbox [8] and a GP system have been used to compare the performance of these methods regarding the classification accuracy as well as the discrimination capability to distinguish between normal and aggressive actions through an off line processing. A robot has been used to act in the same environment as the subject-actor performer, so that to process, classify, and evaluate the physical actions perceived. The robot has a passive role by just processing global activity information and not taking any action as it is going to happen in future projects. Fig. 1 illustrates the hardware configuration setting of the system architecture used. More analytically, a person is shown to act in a 3D environment performing some physical activity. At the same time, two external devices, the 3D tracker (Vicon system) and a mobile robot (SCITOS G5), cooperate as a perception to action unit to produce surveillance assessment reports indicating analytical action classification.

The rest of the paper is organized as follows: In section II, the classification methodologies introduce the architecture used by a number of modules. Section III presents the experimental work showing the classification performances in terms of accuracies achieved. Finally, section IV points out some conclusions and future work derived from the comparison of the methodologies been analyzed.

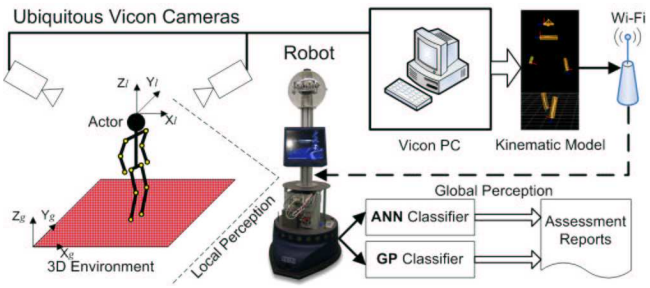


Fig. 1. Configuration setting showing an actor's action performance in a 3D environment, captured by VICON and processed by a mobile to robot to generate assessment reports.

II. METHODOLOGIES

The perception-to-action architecture used (see Fig. 2), consists of a number of modules [9] which carry out the action recognition task, from the data acquisition to the evaluation assessment reports. The first two main modules of the architecture are the Vicon System and a Mobile Robot which belong to the hardware part of the architecture, whereas sub modules implement the software part.

1) The Vicon System: The Vicon system encloses three modules. The Image Acquisition module which is a low level unit used for capturing and fusing data, the Kinematic Model Extraction module which is a commercial software where 3D models are designed according to the alignment of the markers on an object/body, and the Data Sampling module which configures the sampling frequency of the image capturing per second to finally generate time series.

2) The Mobile Robot: The second main module, consists of three grouped modules. The first group deals with data management including the Data Filtering module which has an embedded frequency adjuster used to resample the input time series, a zero-crossing filter for data ambiguities, and a normalizer which rescales the magnitude of the data. The second module, G/L Transformation, is a global-to-local transformation method used to provide independency of body sizes and gender by isolating ambiguous variations of actions performed by human subjects acting in different locations and under different orientations. For the G/L transformation, a cylindrical coordinate system has been used as in [5] where a rotation matrix isolates the orientation whereas Euclidean unit vectors isolate the translation variations. The third module, Fold Decomposition, divides all the time series data into five bit-folds to be used for cross validation. Each fold represents a 20% of a time series. Through this module, cross validation is performed by taking four of the folds for training (= 80%) and one fold for testing. Five experiments are taken in a single run. In the first experiment the testing fold takes the first bit of 20% and the remaining folds are kept for training. In the second experiment the testing fold takes the second bit and so on. The second group consists of two modules denoting the classifiers used (a dynamic ANN and a GP) whereas the third group produces statistical evaluation performances.

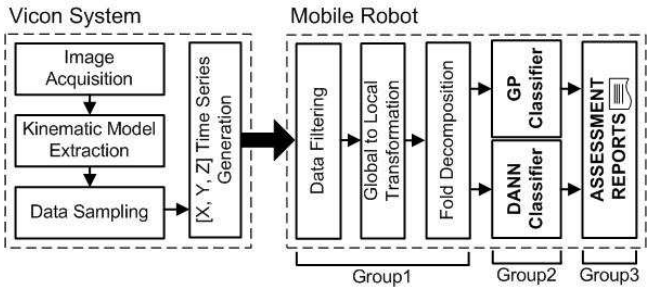


Fig. 2. Module-based architecture showing the collaboration of Vicon system and a mobile robot to produce evaluation statistics by both classifiers.

3) 3D Kinematic Models: The design of the kinematic models is essential for the data acquisition. 23 markers have been used to create six kinematic models, 2×lower legs, 2×lower arms, 1×head, and 1×shoulders. From these models only four of the end effector markers of the limbs (wrists and ankles) have been used to provide activity data while from the shoulder model the central marker was used as a local point of reference. Eventually, the classifiers' input is denoted by a 13^{th}-dimensional feature action vector: $\vec{\mathrm{p}}(t) = [\mathrm{rwrs}^T_{xyz}\ \mathrm{lwrs}^T_{xyz}\ \mathrm{rank}^T_{xyz}\ \mathrm{lank}^T_{xyz}\ \Theta^T_l]^T$, where the ankle and the wrist vectors (ank, wrs) have three dimensions plus the body's orientation θ vector. Similar to [7], each feature action vector generates a combination of motions related to multiple concatenated 3D points or trajectories. The dynamics of the overall feature action vector performance is learnt by both the dynamic ANN and the GP classifier to produce matching class equivalences.

4) The Comparison Analysis: Both classifiers have been compared and tested using similar experimental processes, since the representation domain differs from classifier to classifier, whereas the analysis of the classification accuracy remained the same for both classifiers. Hence, the method to calculate the classification accuracy (CA) as well as to compare and discuss the derived results from the experimental procedure is given by equation 1. Similar equation has been used by [10] to show the percentage of the classes recognized under certain classification methodologies.

$$\mathrm{CA} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{class_i} \times 100\% \tag{1}$$

$$class_i = \begin{cases} 1 & \text{if class } i \text{ has been recognized} \\ 0 & \text{otherwise} \end{cases}$$

where N denotes the overall number of classes, and *class* the recognized class.

A. Dynamic ANN Classifier

As mentioned in [11], Time Delay networks (TDNN) have great performance advantages regarding the time series handling to pattern recognition. Based on the TDNN architecture, the Distributed TDNN networks used in this investigation, have the ability to relate and compare current input to the past history of events, hence, when the network learns its internal representation it performs recognition by passing the input time series over the delay memory vector which has the ability to encode temporal relationships. The delay memory, also called tapped delay line (TDL), is a delayed buffer which discretely shifts and accumulates the input data as time passes [12]; a fact which is the most noteworthy

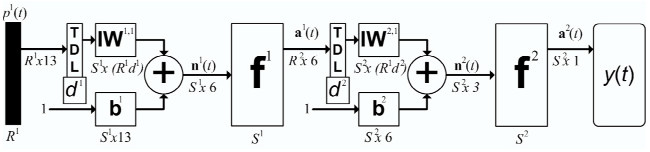


Fig. 3. DTDNN network architecture indicating topological details used for the action recognition task.

characteristic of this dynamic network. The number of taped delay lines allocated at the inputs of every layer, constitutes the network's distribution of delays denoting the network's dynamics to handle time continuities.

The distributed input delays give to the network speeding ability since it does not have to perform dynamic backpropagation for the gradient computation. Actually, the network has two back-propagation passings to compute the gradient descent of the mean squared error (MSE). At the forward pass an input pattern is applied and the network configures the randomly initialized weights to some value, from the first to the last layer, and the error is finally estimated. At the backward pass the derivative of this error is propagated back to adjust all the weights so that to decrease the error achieving thus a desired classification behaviour [11]. Similar to [13], the response of a DTDNN in time t is based on the inputs in times $(t-1)$, $(t-2)$, ..., $(t-n)$. A mapping performed by the DTDNN produces a $y(k)$ output at time k as: $y(k) = f[p(k), p(k-1), ..., p(k-D)]$ where $p(k)$ is the input at time k, and D is the maximum adopted time-delay which is allocated in every layer. This network is also very fast as the standard TDNN and it is recommended by [8] for pattern recognition and classification applications.

For the action classification task a special neuron/layer configuration topology of a DTDNN network has been selected which has shown to perform very efficiently. The selected topology consists of two layers where the first layer $\mathbf{n}^1(t)$ includes six neuron and the second layer $\mathbf{n}^2(t)$ has three as depicted by Fig. 3. The network's weight space is multidimensional consisting of 234 weights **IW** (from the first to the second layer) in overall, which are needed to be adjusted to achieve pattern recognition. This is calculated by multiplying the number of inputs p with the number of delay lines d plus the undelayed lines R. The activation functions selected were the hyperbolic tangent sigmoid for the hidden layers and the linear for the output layer whereas the training method used was the Levenberg-Marquardt (LM) algorithm. Lastly, we have adopted a single output by transforming the derived action time series from continuous form to discrete symbol representations such as class numbers as in [4].

B. GP Classifier

1) Probabilistic Model of Program Output: A novel approach for translating the numerical output of the GP classifier into a class label was introduced in [14]. They used the Gaussian distribution to model the behaviour of each program based on the training examples for each class. This methodology assumes that because normal distributions are possibly the most common distributions found in natural data, it seems reasonable that many clusters of program output could fit well. Based on this assumption, a program output distribution can be modeled as a mixture of normal distributions, with one per class in the classification problem. Clearly, a good program will produce distant output distributions for examples of different classes. A model of each program output distribution for a particular class can be acquired by evaluating the program on the example training set by taking the mean and the standard deviation (SDV) of the program outputs for those training examples [14].

2) Fitness Function Using Gaussian Models: Assuming a binary problem case, the following equation is used to determine the *distribution distance* between classes i and j, as in [14].

$$\mathrm{d} = 2 \times |\mu_\mathrm{i} - \mu_\mathrm{j}| / (\sigma_\mathrm{i} + \sigma_\mathrm{j}) \tag{2}$$

where μ_i, σ_i and μ_j, σ_j are the mean and standard deviation of the program outputs for classes i and j in the training set respectively. Under this measure, for programs that distinguish between two classes well, the distance d will be large, whereas the worst case is 0 where μ_i and μ_j are the same.

In multiclass pattern classification the fitness function is determined by considering the distribution distance between every two classes. For N-class problem there are $C_N^2 = N!/2!(N-2)!$ class combinations and the fitness function takes the following form:

$$\mathrm{fitness} = \left(\frac{1}{\mathrm{T}} \sum_{\mathrm{i}=1}^{\mathrm{T}} \sum_{\mathrm{j}=1}^{\mathrm{C_N^2}} \frac{1}{1+\mathrm{d_j}} \right) - \mathrm{w} \sum_{\mathrm{i}=1}^{\mathrm{S}} \mathrm{Series(i)} \tag{3}$$

$$Series(x) = \begin{cases} 1 & \text{if time series } x \text{ is used as parameter} \\ 0 & \text{otherwise} \end{cases}$$

where T is the number of training examples, N is the number of classes, and d_j is the distribution distance for the class combination j.

While a fitness function based on the average distribution distance between C_N^2 class combinations may suffice, it has not proven to be robust for expression trees that are presented with a large number of parameter values (here, 13 time series) in the authors' experience. Without some form of pressure towards utilising all the inputs provided during fitness evaluation the evolutionary process entails the risk of stagnating in local optima. Intuitively, this is a serious concern in the domain of action recognition where we wish to use all the parameters extracted from the kinematic models in order to enhance the discrimination capacity of the classifier. We add a form of selection pressure towards individuals that utilise as many input parameters as possible by rewarding such programs. The second term of the fitness function subtracts a weighted sum (w is the weight and S is the number of time series representing the parameters of the evolved program) from the average standardised distribution distance which is normalised within the interval $[0,\ 1]$ (0 best, 1 worst). $Series(x)$ is a function that returns 1 if the expression tree structure contains parameter x as a leaf node and 0 otherwise. The weight w is set to 0.001.

3) Probabilistic Pattern Classification: To measure which class belongs to a given pattern, we used *multiple best programs* similarly to [14]. M best programs in the population have used the probability $Prob_c$ of a given pattern being of class c is calculated by:

$$\mathrm{Prob_c} = \prod_{\mathrm{i}=1}^{\mathrm{M}} \mathrm{P}(\mu_{\mathrm{i,c}}, \sigma_{\mathrm{i,c}}, \mathrm{o_i}) \tag{4}$$

TABLE I

PRIMITIVE ELEMENTS FOR EVOLVING CLASSIFIER PROGRAMS

Method	Argument(s) type	Return type
$+, -, *, /$	double, double	double
mean	List	double
std.dev	List	double
skewness	List	double
kurtosis	List	double
$>, \geq, =, <, \leq$	double, double	boolean
and, or	boolean, boolean	boolean
not	boolean	boolean
Conditional		
IF-Then-Else	boolean, double, double	double
Terminal	**Value**	**Type**
Constant	π, $-\pi$, $\pi/2$, $-\pi/2$, $\pi/6$, $-\pi/6$, $\pi/12$, $-\pi/12$, -1.0, 0.0, 1.0	double
Parameter	time series	List

where P is the normal probability density function [14], o_i is the output of program i with the pattern to be classified, $\mu_{i,c}$ and $\sigma_{i,c}$ are the mean and standard deviation of the outputs of program i for class c. The class with the highest probability is designated as the class of the pattern.

$$P(\mu,\sigma,o) = \frac{\exp\left(\frac{-(o-\mu)^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}} \quad (5)$$

4) Representation Language, Evolutionary Algorithm and Run Parameters: Evolvable individuals employ an expression-tree representation. The primitive language is depicted in Table I. During fitness assignment each program is being evaluated with 13 parameters representing the time series. The evolutionary algorithm used, was a panmictic generational Genetic Algorithm (GA) combined with elitism (1%). The algorithm uses tournament selection with a tournament size of 4. The evolutionary run proceeds for 100 generations and the population size is set to 1000 individuals. Evolution halts when all of 100 generations have elapsed. Ramped-half-and-half tree creation [1] with a maximum depth of 7 is used to perform a random sampling of program space during the initial generation. During the run, expression trees are allowed to grow up to depth of 15. Our search employs two variation operators. *Subtree Macromutation* (MM – substituting a node in the tree with an entire randomly generated subtree with the same return type) and *Point Mutation* (PM – substituting a non-terminal node with another non-terminal node of the same return and parameter types, or substituting a terminal node with another terminal node of the same return type). Each operator is applied with a probability of 50%. For the case of PM, the whole expression tree is being traversed and each node is being perturbed with a probability of 15%. Neither reproduction nor crossover have been used.

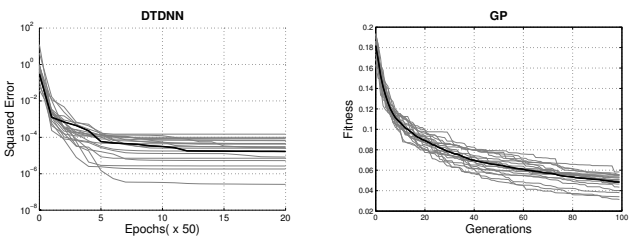
III. EXPERIMENTAL RESULTS

The experimental work has been carried out using the system's architecture Fold Decomposition module (Fig. 2), to test the classification performances using five-fold cross validation by decomposing each time series in five bits. Twenty experimental runs have been taken where each run includes five trials. Each trial tests the classifiers' performance by engaging certain training and testing percentage of data. In overall, 100 experiments have been taken to evaluate the classification accuracy.

The types of actions have been taken from the every day life; such actions are: standing, waving, punching, kicking, etc [5][7]. Fig. 4 depicts an instance of each of the nine actions used in this analysis represented by a human kinematic model which is decomposed in six segmented submodels (head, left and right limbs). To make the expression performance of the actions more realistic, a human partner has been used to help the actor's performance for the handshaking, pushing, and pulling actions, whilst for the slapping, kicking, and punching actions relevant equipment has been used such as punching pads and standing bags.

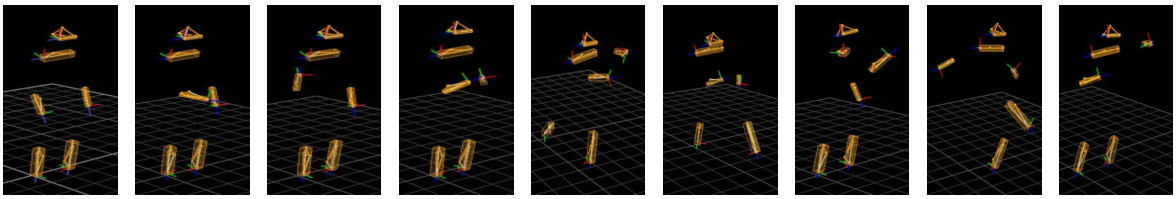
A. DTDNN Analysis

The DTDNN network has been trained for 1000 epochs with a given neuron/layer topology as described in Fig. 3. As it is being observed by Fig. 5(a), sufficient training shown by the average MSE error which converges after ≈400 epochs. The simulation performance of the network (see Fig. 6) shows an expected behaviour where the network's responce in 20% of newly presented testing data was quite stable for normal actions, whilst for aggressive action the responce was unstable. To see this clearly, two evaluation techniques have been tried to illustrate the action performances. Fig. 6(a) depicts a linear regression output performance (LROP) in a correlation evaluation between the input targets with the output responces over 20 experiments. When perfect fit between the input target and the derived output data occurs then the linear regression methodology outputs 1; for $1 \succ \text{LROP} \succ 1$ the output data show an unstable target matching. The behaviour of the normal actions (1, 2, 3, 4) shown in Fig. 6(a), is observed very stable whereas the aggressive actions (5, 6, 7, 8, 9) seem to have unstable behaviour. Similar results have been taken by the second evaluation methodology using the standard deviations (SDV) of the simulated classes. Fig. 6(b) presents the SDV of the classes over 20 experiments. Here, the rational result is to achieve SDVs close to 0, denoting again perfect match.



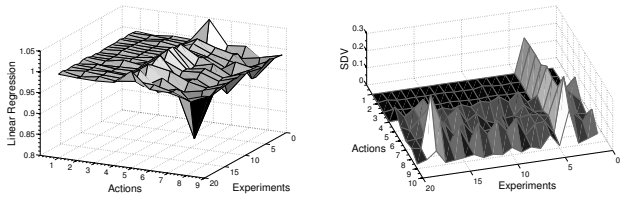
(a) (b)

Fig. 5. Training performances of the classifiers over 20 runs and the average (in bold) performance. (a) Error of the Distributed Time Delay NN, and (b) Fitness of the GP. We acknowledge that the graphs are not comparable and in the analysis they are being treated separately to show the classification performance in terms of learning time, and error minimization.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Fig. 4. Instant action representation of the nine physical actions expressed by 3D kinematic human models. Normal actions: (a) Standing, (b) Handshaking, (c) Waving, (d) Clapping. Aggressive actions: (e) Pushing, (f) Pulling, (g) Slapping, (h) Kicking, (i) Punching.



(a) (b)

Fig. 6. DTDNN simulation performance. (a) Linear regression, and (b) Standard deviation.

In both evaluation methods the results presented by the two action groups (normal and aggressive) where not identical. The evaluation differences from the normal to the aggressive action group was caused because of spatial variances. The spatial clusters used by the normal actions engage an area close to the human body where the distance among all the normal clusters is sufficiently big. On the other hand, the aggressive clusters engage an environment far from the human body whereas the area used by each of the aggressive actions is common. This means that the aggressive clusters mixing and overlapping each other by producing a very complex non linear 3D input space for the network. Eventually, in both evaluation methods the simulation results were good with minor fluctuations from which the network can still achieve good classification accuracy.

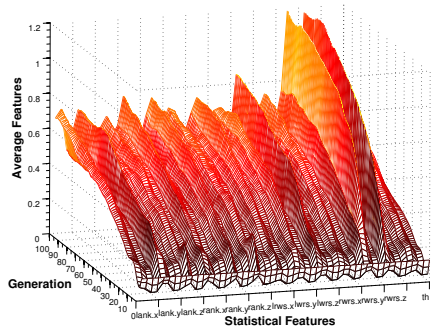
B. GP Analysis

The performance of GP can be mainly attributed to the programming space that has been crafted from the primitive elements that perform statistical feature extraction. We performed a genotypic analysis of individuals during the course of the evolutionary runs to determine which statistical features are actually used within the expression tree structures. Fig. 7(a) presents the evolution of the use of statistical primitives, averaged over 20 independent runs with five-fold cross validation. The axis labeled "Statistical Features" has been decomposed in 13 parts each representing one input time series. Each such part is further decomposed in 4 points each representing the mean, SDV, skewness, and kurtosis of the respective time series. Labeling these on the axis has been omitted for clarity. The graph shows that the average number of statistical features of all input time series is being increased as evolution proceeds, a result that is expected as trees are allowed to grow in depth. However, there are certain features of particular time series whose appearance within individuals is valuable and the selection pressure favors them as evidenced by the average increase of those features while the population is evolving. Fig. 7(b) presents the mean and the SDV (error bars) of the features' existence within a part of the population at the end of the evolutionary run, averaged over 20 trials. The first observation concerns generally the use of statistical features generated by the evolved individuals. Skewness and kurtosis appear to be utilized less than mean and SDV. Features that have been used the most are: $stdDev_y$, $mean_z$ for right wrist and $stdDev_y$, $mean_x$ for the right ankle. For the left part most frequently used features include $mean_z$, $stdDev_y$ for wrist and $stdDev_x$, $mean_y$, $stdDev_y$ for ankle. Generally, features of the right side markers are being utilized more than the respective ones on the left. This result is inline with the action performace session since the right arm and leg have been used the most by the subject-performer.

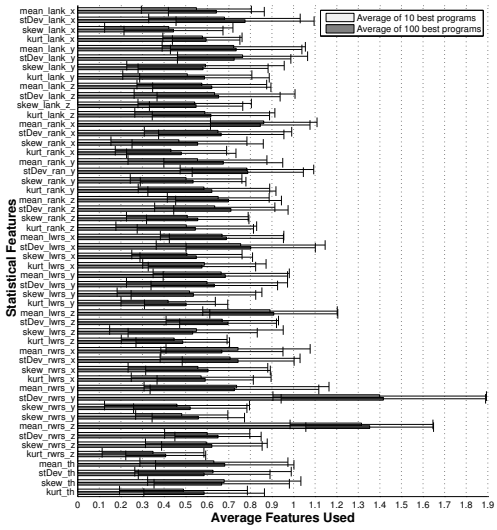
IV. CONCLUSIONS AND FUTURE WORK

From the learning curves depicted in Fig. 5(b) it can be seen that the DTDNN learns faster than GP as this is evidenced by the significant difference in individual evaluations required to reach maximum training performance in the two different methods. While DTDNN has converged in $\approx$400 evaluations by not yielding any significant further improvement, GP resulted a smoother and longer learning fitness curve by typically being minimizing the error until the last generation over $100,000$ evaluations. However, the fact that GP training did not experience any severe form of stagnation is very encouraging and deserves more investigation to induce whether additional processing time will result in a more accurate classifier.

From Table II, which is what constitutes the evaluation assessment reports of this project, it can be seen that the configurations of both DTDNN and GP classifier representations managed to perform well in terms of recognizing action patterns without having though identical performances. More analytically, the table shows the simulation performance of both the DTDNN and the GP over 20 experiments. The performance of the DTDNN tested under three thresholds, has shown increasing classification accuracy even if the thresholds selected were kept very small. The type of the these thresholds is a small amount of tolerance around the mean of the output/classified time series. On the other hand,



(a)



(b)

Fig. 7. (a) Evolution of statistical features usage in population (avg. of 20 runs), (b) Average number and SDV of features in evolved individuals.

the GP instead of thresholds uses numbers of programs to test the classification accuracy.

Ultimately, GP reaches higher classification accuracy than DTDNN. Our hypothesis as to why this is so is that the inclusion of time series statistical primitives in the representation language of evolvable individuals creates a programming space that offers the capability of feature extraction from the input time series. Searching such a space is natural to result in obtaining programs that are composed of both linear and non-linear combinations of such statistical features. On the other hand, dynamic ANNs are in effect forced to consider each sample of the time series separately and due to their representation and learning mechanism they lack of any form of feature extraction and utilization. We strongly believe that this does not imply that we are being unfair to one methodology because both dynamic ANNs and GP are being presented with the same 13^{th}-dimentional input vector.

For future work we are planning to study stateful program representations by allowing the time series to be fed sample-by-sample to the evolvable individual as well as to develop security-like scenarios so that the robot to perform relevant actions according to the recognized human behaviours.

TABLE II

COMPARISON OF CLASSIFICATION ACCURACY

DTDNN		
Threshold %	**Mean Accuracy %**	**SDV**
±0.4	69.0	9.3
±0.8	79.7	3.6
±1.2	86.6	3.5
GP		
No. of Programs	**Mean Accuracy %**	**SDV**
1	92.5	4.5
3	93.2	3.9
5	92.9	3.6
10	91.3	5.2
20	88.9	7.6
50	87.2	8.9
100	79.4	5.7
150	75.2	6.4

REFERENCES

[1] Koza J.R. *Genetic Programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, 1992.

[2] Loveard T. and V. Ciesielski. Representing classification problems in genetic programming. In *Proceedings of the Congress on Evolutionary Computation*, pages 1070–1077. IEEE NNC, EPS, IEE, IEEE Press, 2001.

[3] Ye G., Corso J.J., and Hager G.D. *Real-Time Vision for Human-Computer Interaction*, chapter 7: Visual Modeling of Dynamic Gestures Using 3D Appearance and Motion Features, pages 103, 111, 113. Springer-Verlag, 2005.

[4] Tripathi S. Panchanathan K., Kahol P. Recognizing human movements through human anatomy based coupled hidden markov models. *International Journal on Systemics, Cybernetics and Informatics*, Pentagram Publications, India:25–31, 2006.

[5] Weinland D.l., Ronfard R., and Boyer E. Motion history volumes for free viewpoint action recognition. In *IEEE International Workshop on modeling People and Human Interaction (PHI'05)*, pages 1, 3, 5, 2005.

[6] Muni D.P., Pal N.R., and Das J. Genetic programming for simultaneous feature selection and classifier design. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 36(1):106–117, Feb 2006.

[7] Lv F. and Nevatia R. Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In *9th European Conf. on Computer Vision (ECCV'06)*, volume 4, pages 359–361, 2006.

[8] Demuth H., Beale M., and Hagan M. Neural network toolbox users guide. Technical report, The MathWorks, 2006.

[9] Simpson J.J. and McIntire T.J. A recurrent neural network classifier for improved retrievals of areal extent of snow cover. In *IEEE Transactions on Geoscience and Remote Sensing*, volume 39, pages 2135, 2136, 2138, October 2001.

[10] Song A., Loveard T., and Ciesielski V. Towards genetic programming for texture classification. In *Proceedings of the 14th International Joint Conference on AI*, volume 2256, pages 461–472, Adelaide, Australia, Dec, 10-14 2001. Springer-Verlag.

[11] Waibel A., Hanazawa T., Hinton G., Shikano K., and Lang K. Phoneme recognition using time-delay neural networks. In *IEEE Acoustics Speech and Signal Processing*, pages 329–331, 1989.

[12] Boden M. A guide to recurrent neural networks and backpropagation. Technical report, In The DALLAS project, Report from the NUTEK-Supported Project AIS-8: Application of Data Analysis with Learning Systems, Sweden, 2002.

[13] Souza L.F.R., Rebolho D.C., Caporali A.S., Belo E.M., Marques F.D., and Ortolan R.L. Application of time-delay neural networks for the identification of a hingeless helicopter blade flapping and torsion motions. *the Brazilian Society of Mechanical Sciences*, 27, n. 2:100, 2005.

[14] Smart W. and Zhang M. Probability based genetic programming for multiclass object classification. In *PRICAI 2004: 8th Pacific Rim International Conference on AI*, volume 3157, pages 251–261. Springer, Aug, 9-13 2004.