Identity verification based on haptic handwritten signatures: genetic programming with unbalanced data

Alsulaiman, Fawaz A.; Valdes, Julio J.; El Saddik, Alsulaiman

Publisher's version / Version de l'éditeur:
https://doi.org/10.1109/CISDA.2012.6291531
(CISDA 2012): Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defence Applications, pp. 1-7, 2012-07-13

Identity Verification based on Haptic Handwritten Signatures: Genetic Programming with Unbalanced Data

Fawaz A. Alsulaiman
School of Electrical Engineering
and Computer Science (EECS)
University of Ottawa
Ottawa, Ontario, Canada
Email: fawaz@mcrlab.uottawa.ca

Julio J. Valdes
National Research Council Canada
Emerging Technologies Division
Information and Communications
Technologies Portfolio
Ottawa, Ontario, Canada
Email: julio.valdes@nrc-cnrc.gc.ca

Abdulmotaleb El Saddik
School of Electrical Engineering
and Computer Science (EECS)
University of Ottawa
Ottawa, Ontario, Canada
Email: elsaddik@uottawa.ca

***Abstract*—In this paper, haptic-based handwritten signature verification using Genetic Programming (GP) classification is presented. The relevance of different haptic data types (e.g., force, position, torque, and orientation) in user identity verification is investigated. In particular, several fitness function are used and their comparative performance is investigated. They take into account the unbalance dataset problem (large disparities within the class distribution), which is present in identity verification scenarios. GP classifiers using such fitness functions compare favorably with classical methods. In addition, they lead to simple equations using a much smaller number of attributes. It was found that collectively, haptic features were approximately as equally important as visual features from the point of view of their contribution to the identity verification process.**

I. Introduction

Haptics, derived from the Greek word "haptesthai" which means the sense of touch, is an emerging area of research that enables the sensing and manipulation of virtual environments through touch. Its applications [1], [2], [3], [4] are wide, and range from surgical simulation to gaming and entertainment. Many of haptics' current applications involve the analysis and interpretation of acquired haptic information in order to possibly reveal certain patterns in the data. Haptic-based biometrics is one example of such applications. Biometric systems provide a solution to ensure that protected services are solely accessible by a legitimate user. This is achieved while relying on users' behavioral and/or physiological characteristics. The two primary uses of biometrics are in user verification and identification. User verification is a one-to-one comparison of a person's biometric template with his or her original sample previously stored in the system. The verification result is a "match" or "no-match". Conversely, user identification is a one-to-many comparison problem where a biometric template is compared against a biometric database with the attempt to identify an unknown individual. The possible use of haptic devices in biometric systems has been suggested in recent years to enhance user identification and verification performance over more traditional techniques, such as those based on handwritten signatures [2], [4].

Haptic data, which depict trajectory, cutaneous as well as kinesthetic information, essentially consist of position, velocity, orientation, torque and force information that are directly acquired from a haptic interface upon a user's interaction with a predefined virtual environment. However, regardless of the intended haptic application, the number of resulting features are usually significantly large (in the thousands range) as the recorded information typically consists of multidimensional time-varying data.

User identity verification and identification using high-dimensional haptic information has been little explored in the literature. Initial work [5] in this area examined users' haptic-based biometric characteristics using a maze application, in particular an environment which enables participants to solve a haptic-enabled virtual maze. The authors applied different techniques, including Hidden Markov Models, spectral data analysis and time warping to explore the possibility of real-time (and continuous) haptic user verification. Their preliminary results demonstrated that a verification rate of up to 78.8% with a 25% False Acceptance Rate (FAR) can potentially be achieved. In [2] haptic datasets are acquired using two different applications in order to explore the feasibility of a haptic-based user authentication system. The haptic-enabled applications corresponded to the aforementioned virtual maze application, and a haptic-based virtual check (virtual equivalent of a real bank check). In subsequent work [4] a verification rate of up to 92% with a 25% FAR was reported.

In [6] the focus was on haptic signature analysis using a visual data mining approach using the previously mentioned virtual maze and virtual check applications, as well as a virtual phone on which users can haptically dial a number (virtual equivalent of a cellular phone). User-behavior-similarity across the three applications (for a single user and using all the acquired features) using 3D visual representations of the recorded data was investigated and 3D virtual spaces were constructed using deterministic optimization technique.

A similar approach was used in [7], [8] in order to explore the within-user variations and between-user variations of haptic-based handwritten signatures. The potential benefits of haptics in graphical password authentication systems; specifically in preventing shoulder surfing attacks was approached in [9], [10]. Their results suggested that on average successful verification probabilities of 90.1% and 92.0% were achieved using Artificial Neural Network and Nearest-Neighbor's classifiers respectively.

In [11] haptic feature vectors were redefined using all haptic data attributes associated with a single signature. The approach was based on rough set theory for feature selection in high-dimensional haptic-based signature datasets. In [1] haptic-based handwritten signatures are analyzed within a visual data mining paradigm while relying on unsupervised construction of virtual reality spaces using classical optimization and genetic programming (GP). GP was used in [12] for feature selection and classification with unbalanced datasets (raw data and random under-sampling). The results demonstrated that, on average, for all datasets, whether imbalanced or under-sampled, a certain numberof perfect classification models were found. In addition, great feature reduction was achieved via genetic programming.

In this paper, GP is used in its dual classification and feature selection capabilities to approach the unbalanced dataset problem. In the present case, a collection of fitness functions are used which specifically consider the information provided by the class distribution to cope with the lack of balance between the classes. Elements considered here are: *i)* finding analytic functions as classifiers in high-dimensional haptic feature spaces and *ii)* a comparative study of the behavior of four fitness functions oriented to unbalanced class distributions.

The rest of the paper is organized as follows. In Section III an introduction to the basic concepts and methods exploited in this work is provided. In Section III-A gene expression programming based classification and feature selection are described. In Section IV the experimental settings are described. In Section V the results are illustrated. Finally, conclusive remarks are outlined in Section VI.

II. Haptic-Signature Data and the Virtual Check Application

The experiments were performed using the Reachin Display [13], which integrates a haptic device with stereo graphics for an immersive and high quality 3D experience. The Reachin visuo-haptic interface enables users to see and touch virtual objects at the same location in space. The haptic stimulus is sensed using the SensAble PHANTOM Desktop force-feedback device, which is equipped with an encoder stylus that provides 6-degree-of-freedom single contact point interaction and positional sensing. Furthermore, the visual stimuli as depicted in Fig. 1, consist of a virtual pen and a virtual check on which users can record their handwritten signature. Conversely, the haptic stimuli are force and frictional feedback that attempt to mimic the tactile sensations felt when signing a traditional paper check. More specifically, the check is

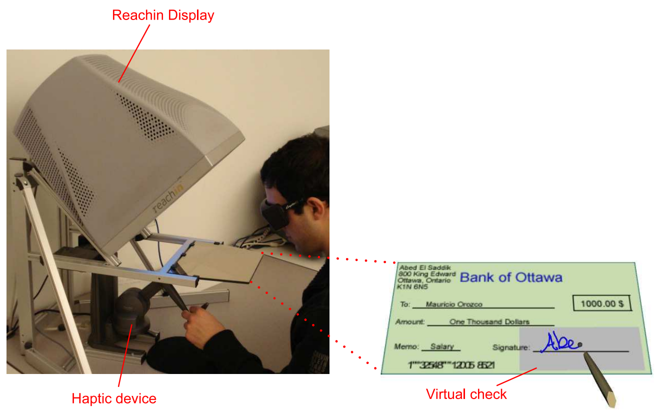

Fig. 1. Haptic-enabled virtual check application.

built on an elastic membrane surface with particular texture features, providing the users with a user-friendly and realistic feel of the virtual object. Moreover, similar to conventional dynamic signature verification technologies, the virtual check application records a wide array of attributes that depict a user's physical and behavioral traits. A group of 13 participants contributed with 10 instances of their signatures, handwritten on the virtual check. Accordingly, there is a one-to-one correspondence between each class a a participant, with a total of 130 haptic-based vectors. Details are given in Section IV-1.

III. Computational Intelligence and Machine Learning Techniques

A. Evolutionary Computation

Evolutionary Computation [14], [15] is a general term for several computational techniques which are inspired and/or based to some extent on the evolution of biological life in nature. An Evolutionary Algorithm (EA) is an iterative and stochastic process that operates on a set of individuals (population). Each individual represents a potential solution to the problem being solved. This solution is represented by means of an encoding/decoding mechanism. An EC-algorithm consists of the following general steps:

1) Build the initial population.
2) Evaluate the objective function (fitness function).
3) While the termination criteria are not met:
   - Apply evolutionary operators (selection, reproduction, crossover, mutation, etc.).
   - Evaluate the objective function.

B. Genetic Programming: Gene Expression Programming

In particular, evolving functions and programs is the realm of Genetic Programming (GP). Analytic functions, which has a long history of usage in science, are among the most important building blocks for modeling and are a classical way of expressing knowledge. From a data mining perspective, direct discovery of general analytic functions poses enormous challenges because of the (in principle) infinite size of the search spaces.

Within computational intelligence, GP techniques aim at evolving computer programs, which ultimately are functions. Genetic Programming (GP) introduced in [16] and further elaborated in [17], [18] and [19], is an extension of the Genetic Algorithm. GP starts with a set of randomly created computer programs. This initial population goes through a domain-independent breeding process over a series of generations. It employs the Darwinian principle of survival of the fittest with operations similar to those occurring naturally, like recombination of entities (crossover), occasional mutation, duplication and gene deletion. The operations include arithmetic computation (possibly involving many other functions), conditionals, iterations, recursions, code reuse and other kinds of information processing organized into a hierarchy. GP combines the expressive high level symbolic representations of computer programs with the search efficiency of the genetic algorithm. For a given problem, this process often results in a computer program which solves it exactly, or if not, at least provides a fairly good approximation.

There are several approaches to genetic programming. One of them is the Gene Expression Programming (GEP) technique [20], [21]. GEP individuals are nonlinear entities of different sizes and shapes (expression trees) encoded as strings of fixed length. For the interplay of the GEP chromosomes and the expression trees (ET), GEP uses an unambiguous translation system to transfer the language of chromosomes into the language of expression trees and vise versa. The structural organization of GEP chromosomes allows a functional genotype/phenotype relationship, as any modification made in the genome always results in a syntactically correct ET or program. The set of genetic operators applied to GEP chromosomes always produces valid ETs. The chromosomes in GEP itself are composed of genes structurally organized in a head and a tail [20]. The head contains symbols that represent both functions (elements from a function set $\mathcal{F}$) and terminals (elements from a terminal set T), whereas the tail contains only terminals. The function set is an important component because it contains the functional building blocks for assembling analytical expressions. Its choice in cardinality and composition allows the user to introduce external knowledge in the evolutionary process, as the evolved expressions could only contain functional terms comming from $\mathcal{F}$.

In the case of the terminal set, two different alphabets occur at different regions within a gene. For each problem, the length of the head h is chosen, whereas the length of the tail t is a function of h, and the number of arguments of the function with the largest arity. The length of the tail is evaluated given by $t = h(n-1)+1$. As an evolutionary algorithm GEP defines its own set of crossover, mutation and other operators [21].

C. The Unbalanced Dataset Problem

An important issue that must be considered in this work is the unbalanced nature of the exploited haptic datasets. The unbalanced dataset problem generally occurs in two-class domains when the number of instances belonging to one class (the majority class) is significantly larger than the number of instances belonging to the other class (the minority class). This situation is frequently found in many real world applications (e.g. medical diagnostics) and it affects the performance of many classifiers, which tend to predict the majority class with high accuracy, while missing much if not all of the minority class [12].

There are three main approaches to the unbalanced dataset problem:

- re-sampling: In this case a new dataset is generated by resampling the original one, either undersampling the majority class or oversampling the minority one [22], [23], [24].
- cost assignments: Data instances are assigned differential cost values in order to compensate the class imbalance [25], [26], [27].
- use of specific fitness functions: In the context of genetic programming, dynamic class thresholds instead of fixed ones are used to counterbalance the classes. Also, fitness functions including terms depending on the relative class distributions stir the evolutionary process in ways that considers the lack of class balance [28], [29], [30], [31]. This is the approach used in this paper, while resampling was used elsewhere [12]

D. Genetic Programming's Fitness Functions for Unbalanced Datasets

A Genetic programming perspective to the imbalanced dataset problem was introduced in [32], [28] utilizing an overall accuracy as a fitness function that evolves classifiers with a discrimination bias towards the majority class. Four GP fitness functions that consider the performance of the majority class and the minority class separately are

$$\text{BH1} = \sqrt{\frac{hits_{min}}{N_{min}} \times \frac{hits_{maj}}{N_{maj}}} \tag{1}$$

where $hits_{min}$ are the number of correct classification of minority class, $hits_{maj}$ represent the number of correct classificatoin of majority class, N_{min} is the total number of minority class instances and N_{maj} is the total number of majority class. Equation 1 utilize the geometric means of the two objectives, maintaining the classifier performance for both the majority class and the minority class.

The second fitness introduced by [28] is conceptually based on a similar bases as Equation 1 with the addition of a third objective which is maintaining an overall classification accuracy rather than just focusing on the classification accuracy of majority and minority class. Thus, [28]

$$\text{BH2} = \frac{hits_{min}}{N_{min}} + \frac{hits_{maj}}{N_{maj}} + \frac{hits}{N} \tag{2}$$

where $hits$ represents the overall number of correct classification and N represents the total number of instances.

The next two fitness functions were introduced in [32]

$$\text{PA1} = \frac{\frac{hits_{maj}}{N_{maj}} + \frac{hits_{min}}{N_{min}}}{2} \times 100\% \tag{3}$$

$$\text{PA2} = \frac{\left(\frac{hits_{maj}}{N_{maj}}\right)^2 + \left(\frac{hits_{min}}{N_{min}}\right)^2}{2} \times 100\% \qquad (4)$$

These two fitness functions are reported to have better minority class performance on several real and synthetic data.

IV. Experimental Settings

1) Haptic Dataset: The haptic-based handwritten signatures are obtained from 13 different participants, where 10 signatures are collected per individual. The data acquired depict various distinct haptic features as a function of time. A number of haptic data types are considered that characterize the instantaneous state of the haptic system, including, three-dimensional position, force (pressure exerted on the virtual check), torque, and angular orientation. Also, haptic data are recorded at 100 Hz. As the data is time-varying, the resulting number of attributes per signature is in fact the number of haptic data types considered (position, force, torque, etc.) times the number of samples recorded per data type during each signature acquisition. This evidently leads to significantly large feature vectors that encompass thousands of haptic-based attributes. In order to ensure accurate discrimination between the signatures, the obtained feature vectors were normalized to a common length of 10000. Essentially, the acquired haptic data types are re-sampled (upsampled/downsampled) when necessary to ensure a common feature vector length across all instances. The latter feature vector length was selected in such a manner to minimize the information loss that is most apparent when downsampling is performed. Consequently, the computed preprocessed dataset contains 130 instances, where each consists of 10000 features.

As aforementioned, in this paper we are concerned with biometric identity verification, which is a two-class classification problem, where a dichotomizer assigns class labels A (accept) or B (reject) to observed feature vectors x. The preprocessed dataset is therefore rearranged into 13 distinct datasets (one for each class), where in each only the instances of a single class are labeled *accept* (A = 1), whereas the remaining instances are labeled *reject* (B = 0). It is clear that the obtained datasets are highly unbalanced as the number of instances belonging to the *accept* class is much smaller than the number of instances associated with the *reject* class. Nonetheless, the obtained biometric identity verification datasets were then first divided into 60% training and 40% test sets.

A. Gene Expression Programming Algorithm Settings

The GEP experiments were performed using the implementation from [33], which extends the ECJ environment [34]. The parameter values used in the experiments are shown in Table I. It can be seen that the mathematical expressions are composed of one chromosome per individual and each chromosome is made from 3 gene expressions, all linked by the addition operator. The gene expressions can be formed using constants, the independent variables associated with each problem, and any of the functions from the function set. Weights can be associated to the functions in $\mathcal{F}$ in order to bias the evolutionary process towards the use of certain functions more than others. This is another mechanism that allows the introduction of external knowledge. In the present case all functions had the same weights.

TABLE I
Experimental settings of the GEP algorithm.

GEP Parameter	Experimental Values
No. generations	100000
Population size	1000
No. chromosomes / individual	1
No. genes / chromosome	3
Gene head size	8
Linking function	addition
No. constants / gene	2
Bounded range of Constants	[0,10]
Inversion rate	0.1
Mutation rate	0.044
is-transposition rate	0.1
ris-transposition rate	0.1
One-point recombination rate	0.3
Two-point recombination rate	0.3
Gene recombination rate	0.1
Gene transposition rate	0.1
rnc-mutation rate	0.01
dc-mutation rate	0.044
dc-inversion rate	0.1
Function set (e.g., {function$_1$(weight)) function$_2$(weight), ...})	{add(1), sub(1), mult(1),div(1)}

For each experiment a set of 100 runs was performed with random seeds. It is important to mention that during each run, 1000 different expressions are used for providing a rich genetic diversity at the onset of the evolution.

V. Results

For each of the 13 subsets, 100 independent GEP runs were conducted (100 different analytical functions are generated for each class). It is important to mention that only the GEP model with the best discrimination performance is selected (based on the training results).

A. Classification

Classification performance is analyzed in terms of the verification success rate (VR), False Acceptance Rate (FAR), and False Rejection Rate (FRR). The VR, FAR, and FRR results using genetic programming models evolved with various fitness functions (NH, BH1,BH2, PA1, PA2) are presented in Table II. NH is the Number of Hits which represents the number of instances that are classified correctly and it is used as the baseline fitness function.

The classification results are based on the average performance of all generated analytical functions i.e., 100 analytical functions per subset (class). The best GP individuals for each subset are considered those whose classification error is within the first quartile of the error distribution. Then, selection is based on each model's classification accuracy of test data. The average classification performance of the first quartile of the top GP models is illustrated in Table III. The GP results shows an important decrease in the number of variables used and in the number of operations (that is, smaller model complexity).

The number of variables refers to the actual number of occurrences of a given attribute within an expression. The number of operations includes all occurrences of functions from $\mathcal{F}$ within an expression (duplication of identical operator counts).

TABLE II
CLASSIFICATION RESULTS BASED ON THE AVERAGE PERFORMANCE OF ALL GP-GENERATED ANALYTICAL FUNCTIONS (100 ANALYTICAL FUNCTIONS PER CLASS) WITH VARIOUS FITNESS FUNCTIONS.

Fit.Func	*60%*				
	VR (%)	*FAR (%)*	*FRR (%)*	*No. of Variables*	*No. of Operations*
NH	91.49	5.65	42.85	7.67	22.63
BH1	89.27	8.44	38.15	6.22	8.46
BH2	89.77	7.92	37.98	6.18	8.26
PA1	89.67	8.00	38.29	6.30	8.57
PA2	89.83	7.94	36.87	6.04	8.23

TABLE III
CLASSIFICATION RESULTS BASED ON THE AVERAGE PERFORMANCE OF THE FIRST QUARTILE OF THE BEST GP-GENERATED ANALYTICAL FUNCTIONS (PER CLASS) WITH VARIOUS FITNESS FUNCTIONS FOR 13 USERS.

Fit.Func	*60%*				
	VR (%)	*FAR (%)*	*FRR (%)*	*No. of Variables*	*No. of Operations*
NH	95.90	1.93	30.13	7.39	22.88
BH1	92.89	6.34	16.31	5.97	7.74
BH2	91.84	6.56	27.31	5.84	7.32
PA1	91.34	7.19	26.31	6.10	8.14
PA2	93.25	5.90	16.92	5.73	7.87

The average performance over all generated GP models is considered in Table II. It is observed an additional decrease in the number of variables and in the number of operations with the fitness functions that tackle the problem of unbalanced data sets. GP-generated models using fitness functions (BH1,BH2, PA1, PA2) exhibited a decrease in FRR and a slight increase in FAR. However, as FRR of NH is high (42.85%) a decrease would be more favorable. However, when the average of the best first quartile models are considered III, an important decrease of FRR can be observed. Moreover, (BH1,BH2, PA1, PA2) require a much smaller number of operations (8) compared to those required when the default NH fitness function is used (22.63). A slight decrease in the number of variables is also observed. In addition, the use of fitness functions (BH1, PA2) resulted in a substantial decrease of FRR.

An example of a GP classifier obtained when using the BH1 fitness function (Eq. 1 for the first experimental subject (class 1) is

$$class1 = \begin{cases} 1 & \text{if } (Ty_{113} + Pz_{105} * Px_{211} - Px_{618} \\ & \quad +Tx_{737} + Px_{211} * Pz_{883} \\ & \quad) > 0.5 \\ & \\ 0 & otherwise \end{cases} \tag{5}$$

where Tx and Ty are the torques along the x and y axis respectively, Px and Py are the positions along the x and y axis. The subindexes indicate relative time units for that particular feature when producing the signature. As a white box model, it clearly shows both the influence of the different attributes in the final class decision and that of the individual functions in the final result. Note that, for example, division is an element of the function set, but it was not used in the classifier. Note the simple functional structure of the expression, which on the other hand, has an excellent performance: 100% accurate on training and 96.15% with the testing set.

B. Relevance of Haptic Data Types in Identity Verification

The frequency distributions of different haptic data types that appear in GP-generated models for NH,BH1,BH2,PA1,PA2 fitness functions are shown as Box plots in Figs. 2.

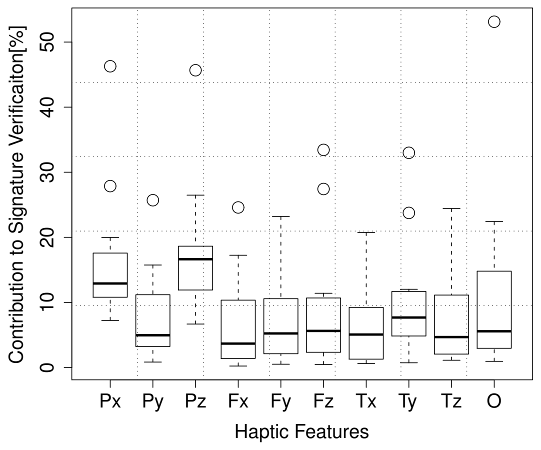

Fig. 2. Box plots illustrating the frequency of haptic data types found in GP-generated models when unbalanced dataset are considered with NH fitness function.

Each box includes lines indicating the median and the mean, in addition to the first and third quartiles. Lines extending from each end of the box (whiskers) are provided to show the extent of the frequency of different haptic data types found in the generated GP-models. Outliers are represented using the 'o' symbol and consist of frequency values that go beyond the ends of the whiskers.

From Figs. 2, 3, 4, 5, 6, it can be noticed that position information (P_x, P_y, and P_z) appeared in GP models with a probability of approximately equal to 0.5. Consequently, the remaining haptic-specific data types combined, i.e., force, torque and orientation (F_x, F_y, F_z, T_x, T_y, T_z, and O) appeared in GP-models with a similar probability (approximately equal to 0.5). Accordingly, the importance of haptic specific features is seen as almost equal to that of visual attributes. This result clearly shows that the inclusion of haptic information enrich signatures with additional non-visible behavior elements.

In general, among the visual features, Pz (Position of the pen along the z-axis) is the one used most frequently within the

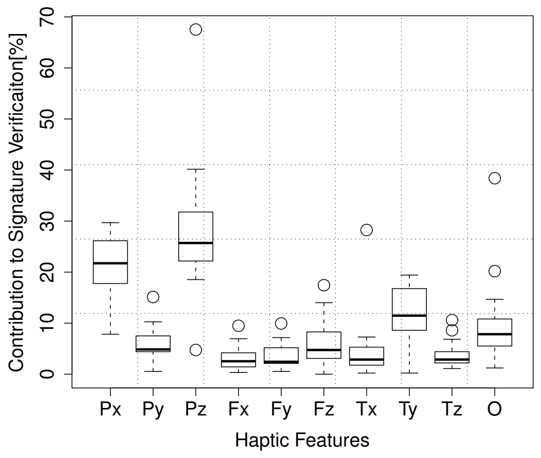

Fig. 3. Box plots illustrating the frequency of haptic data types found in GP-generated models when BH1 are considered as a fitness function

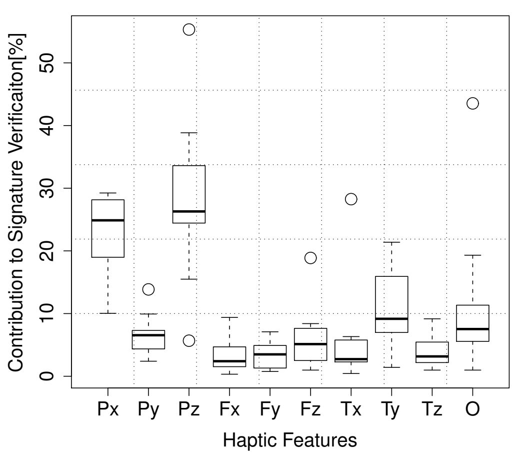

Fig. 5. Box plots illustrating the frequency of haptic data types found in GP-generated models when PA1 are considered as a fitness function

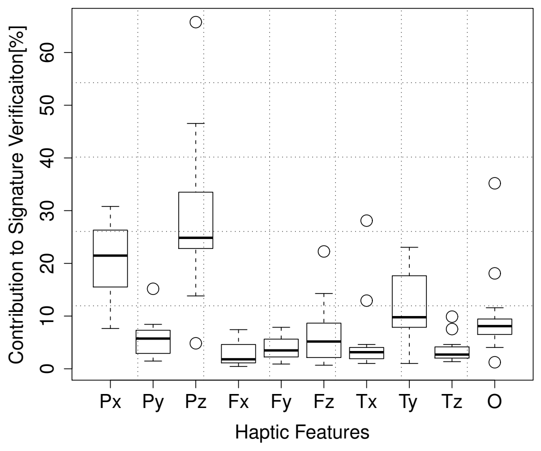

Fig. 4. Box plots illustrating the frequency of haptic data types found in GP-generated models when BH2 are considered as a fitness function

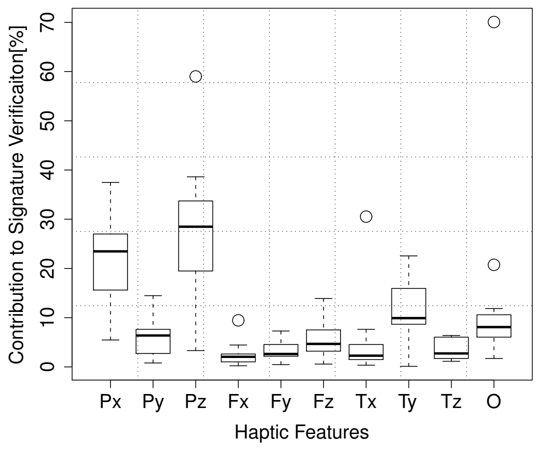

Fig. 6. Box plots illustrating the frequency of haptic data types found in GP-generated models when PA2 are considered as a fitness function

GP models. From the point of view of pure haptic variables, Ty (torque long the y-axis) was the most frequently appearing in the GP models, followed by O (angular rotation).

VI. Conclusion

The problem of Haptic-based handwritten signature verification is addressed via Genetic Programming (GP) in this paper. As the problem of identity verification is a two-class classification problem where a positive class represents a single user while the negative class represents other users. Thus, an unbalanced dataset problem emerges. In this paper, the unbalanced dataset problem is addressed through the utilization of fitness functions to drive the evolution process towards analytical functions that consider both minority class and minority class with similar importance. Future work will include comparisons with other feature selection and classification approaches, additional fitness functions and studies of their performance on other datasets.

References

[1] N. Sakr, F. A. Alsulaiman, J. J. Valdés, A. E. Saddik, and N. Georganas, "Exploring the underlying structure of haptic-based handwritten signatures using visual data mining techniques," in *Proc. of IEEE Haptics Symposium*, March 2010, pp. 467–474.

[2] A. E. Saddik, M. Orozco, Y. Asfaw, S. Shirmohammadi, and A. Adler, "A novel biometric system for identification and verification of haptic users," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, no. 3, pp. 895–906, June 2007.

[3] N. Sakr, F. A. Alsulaiman, J. J. Valdés, A. E. Saddik, and N. Georganas, "Feature selection in haptic-based handwritten signatures using rough sets," in *Proc. of IEEE World Congress on Computational Intelligence*, July 2010, pp. 1–8.

[4] M. Orozco, M. Graydon, S. Shirmohammadi, and A. E. Saddik, "Experiments in haptic-based authentication of humans," *Journal of Multimedia Tools and Applications*, vol. 37, no. 1, pp. 73–92, 2008.

[5] M. Orozco, Y. Asfaw, S. Shirmohammadi, A. Adler, and A. E. Saddik, "Haptic-based biometrics: A feasibility study," in *Proc. Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2006, pp. 256–271.

[6] R. Iglesias, M. Orozco, J. Valdés, and A. E. Saddik, "Behavioral features for different haptic-based biometric tasks," in *Proc. of IEEE International Workshop on Haptic, Audio and Visual Environments and Games*, 2007, pp. 102–106.

[7] R. Iglesias, M. Orozco, F. Alsulaiman, J. Valdés, and A. E. Saddik, "Characterizing biometric behavior through haptics and virtual reality," in *Proc. of the 42nd Annual IEEE International Carnahan Conference*, 2008, pp. 174–179.

[8] R. Iglesias, M. Orozco, J. Valdés, and A. E. Saddik, "Visualizing human behavioral features based on signature haptic data," in *Proc. of the International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2008, pp. 451–456.

[9] B. Malek, M. Orozco, and A. E. Saddik, "Novel shoulder-surfing resistant haptic-based graphical password," in *Proc. of Eurohaptics Conference*, 2006, pp. 179–184.

[10] M. Orozco, B. Malek, M. Eid, and A. E. Saddik, "Haptic-based sensible graphical password," in *Proc. Virtual Concept*, 2006.

[11] N. Sakr, F. A. Alsulaiman, J. J. Valdés, A. E. Saddik, and N. Georganas, "Relevant feature selection and generation in high dimensional haptic-based biometric data," in *Proc. of International Conference on Data Mining (DMIN)*, July 2009, pp. 71–77.

[12] F. Alsulaiman, N. Sakr, J. Valdés, A. E. Saddik, and N. Georganas, "Feature selection and classification in genetic programming: Application to haptic-based biometric data," in *Proc. of IEEE Symposium on Computational Intelligence for Security and Defence Applications*, Ottawa, Canada, July 2009, pp. 1–7.

[13] R. T. ab. Reachin Display., "http://www.reachin. se/products/."

[14] D. Fogel, "What is evolutionary computation?" *Spectrum, IEEE*, vol. 37, no. 2, pp. 26, 28–32, Feb. 2000.

[15] K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[16] G. John and P. Langley, "Hierarchical genetic algorithms operating on populations of computer programs," in *Proc. of the 11th International Joint Conference on Artificial Intelligence*, 1989, pp. 768–774.

[17] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[18] ——, *Genetic programming II: Automatic discovery of reusable programs*. MIT Press, 1994.

[19] F. B. I. J. Koza, D. Andre and M. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, 1999.

[20] C. Ferreira, "Gene expression programming: A new adaptive algorithm for problem solving," *Journal of Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.

[21] ——, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Springer Verlag, Germany, 2006.

[22] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training set: One sided selection," in *Proc. of the 14th International Conference on Machine Learning*, Nov. 2004, pp. 179–186.

[23] L. Tomek, "Two modifications of cnn," *IEEE Transactions on Systems, Man and Communications*, vol. 6, no. 11, pp. 769–772, 1976.

[24] N. V. Chawla, K. W. Bowyer, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[25] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk, "Reducing misclassification costs," in *Proc. of the 11th International Conference of Machine Learning*, 1994, pp. 217–225.

[26] N. Japkowics and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis Journal*, vol. 6, no. 5, pp. 429–449, 2002.

[27] J. V. Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proc. of the 24th International Conference on Machine Learning*, 2007, pp. 935–942.

[28] U. Bhawan, M. Johnston, and M. Zhang, "Differentiating between individual class performance in genetic programming fitness for classification with unbalanced data," in *Proc. of the 11th Conference on Congress on Evolutionary Computation*, 2009, pp. 2802–2809.

[29] J. Doucette and M. Heywood, "Gp classification under imbalanced data sets: Active sub-sampling and auc approximation," in *Proc. of the European Conference on Genetic Programming*, 2008, pp. 266–277.

[30] U. Bhowan, M. Zhang, and M. Johnston, "Genetic programming for classification with unbalanced data," *Lecture Notes in Computer Science, Genetic Programming*, vol. 6021, pp. 1–13, 2010.

[31] ——, "A comparison of classification strategies in genetic programming with unbalanced data," *Lecture Notes in Computer Science, AI 2010: Advances in Artificial Intelligence*, vol. 6464, pp. 243–252, 2011.

[32] G. Patterson and M. Zhang, "Fitness functions in genetic programming for classification with unbalanced data." in *In Proceedings of the 20th Australian Joint Conference on Artificial Intelligence.*, 2007.

[33] J. Valdés, A. Barton, and R. Orchard, "Exploring medical data using visual spaces with genetic programming and implicit functional mappings," in *Proc. of Genetic and Evolutionary Computation*, 2007, pp. 2953–2960.

[34] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, E. Popovici, J. Harrison, J. Bassett, R. Hubley, and A. Chircop, "Ecj. a java-based evolution computing research system," 2007, evolutionary Computation Laboratory, George Mason University.