Discriminant Feature Selection by Genetic Programming: Towards a domain independent multi-class object detection system

Jacques-André Landry, Luis Da Costa and Thomas Bernier

École de Technologie Supérieure, Université du Québec
Montréal, Québec, Canada

ABSTRACT

In order to implement a multi-class object detection system, an efficient object representation is needed; in this short paper, we present a feature selection method based on the **G**enetic **P**rogramming paradigm. This method allows for the identification of a set of features that best represent the classes in the problem at hand. The idea would then be to have a broad set of features to describe any object, and then to use the presented feature selection method to adapt the description to the actual needs of the classification problem. Furthermore, the tree like solutions generated by the method can be interpreted and modified for increased generality. A brief review of literature, the first implementation of the method and the first results are presented here. The method shows potential to be used as a building block of a detection system, although further experimentation is underway in order to fully asses the power of the method.

Keywords: Feature selection, genetic programming, pattern recognition.

1 INTRODUCTION

Automated visual recognition and detection processes are increasingly prevalent in most scientific fields and in many areas of industry. As the availability of information in electronic form increases, more sophisticated processing methods are required. For visual detection processes, this often takes the form of sample query based image searches, i.e. of the form: "*Given these examples of objects, detect and locate all similar objects in other images*". Examples of this type of problem include target detection ([1–3]) where the task is to find all objects of a certain type in an image; for instance, in [4] the authors searched for tanks, trucks, or helicopters in a map (an aerial image).

In most cases, systems are painstakingly designed and developed in order to detect only a single and specific object or property of an object. We feel that a domain independent resolution method i.e. a method that will work, without adjustment, for any detection problem, is needed. In order to attain this objective, the basic problem of finding a *sound* and *complete* representation for objects have to be addressed. Indeed, if we wish to create a method that works *for any detection problem*, then this representation should also be applicable to any of them. This implies not only a set of characteristics broad enough, but also a procedure for selecting *the best of them for the problem at hand*.

One of the main practical obstacles to finding the *most discriminating* features is the manual tuning of the parameters that a software system would require for the recognition of particular objects. Thus, while knowing a wide range of different features would (in theory) allow us to detect virtually any object in images, it would require the input of an expert in order to select the object features most useful for performing the task in a reasonable amount of time. This shortcoming significantly limits the practicality of any given system.

In this paper we present a concept to automate the selection of features that are pertinent to any given problem. That would be the first step towards building a system solving any detection problem with a high independence from user input. The automatic method chosen is **G**enetic **P**rogramming (**GP**, defined by J. Koza in [5]); the optimal set of features are extracted from the solutions obtained by **GP**, solutions that optimize the process of classification that represents the problem to solve. The presented method will be a component of a global object recognition framework under development by the authors ([6, 7]).

The objectives of the current work is presented in Section 2. A brief revision of relevant work in the field is covered in Section 3. Section 4 contains the methodology used in our experiments. Section 5 presents preliminary experiments demonstrating the viability and performance of the method, which are discussed in Section 6. Finally, a tentative outline for future research is presented in Section 7.

2 OBJECTIVES

The main objective of this investigation is to develop a method that uses **G**enetic **P**rogramming for the selection of the optimal features for a particular classification problem. We will then compare the performance of the proposed method to other classification methods that use the full set of features.

3 STATE OF THE ART

Genetic **P**rogramming (**GP**) is part of a very large body of research called **M**achine **L**earning (**ML**) (*the study of computer algorithms that improve automatically through experience* [8]). **GP** tries to improve a population of programs through their experience with the data on which they are trained. The primary objective of **GP** (and in general **ML**) is to be able to simply define a task and have the machine independently learn to perform it: **GP** provides the framework in which the machine can evolve its own algorithms, representing its solutions as a computer program or as data that can be interpreted as a computer program [9]. The premise of **GP** systems is based on a beam search [10] in which an evaluation metric is used to rank the fitness of solutions; the most promising solutions are retained for further transformation while others are discarded.

GP became very popular amongst computer scientists (perhaps because of the higher conceptual level at which the algorithm operates, as suggested by P. Bentley in [11]), even though it was quickly understood that there were problems with the definition of the genetic operators for **GP** (that often *destroy* the good solutions, instead of creating better ones).

However, the number of publications in the field of **GP** are as numerous as the ones in its sister field of **G**enetic **A**lgorithms (**GA**): there are applications in different fields such as biochemistry data mining ([12]), image classification in geoscience and remote sensing ([13]), cellular encoding of artificial neural networks ([14]) and image analysis ([15,16]). **GP** is now used in alternative areas, as J. Koza did when describing the design of analog circuits by means of **G**enetic **P**rogramming ([11], chap. 16).

4 METHODOLOGY

The feature selection method by **G**enetic **P**rogramming was tested by solving a set of classification problems. For each of those, the performance (recognition rate) of the derived classification algorithm was compared to that of using the whole set of features. The classification problems will be described in section 5; the complete set of features is described in 4.1, and the **GP**-representation and operators are presented in 4.2.

4.1 Object representation

Each of the object comparison functions pertains to a single measurable feature of a given object. Each function requires one parameter to identify the model to which the object will be compared and each function returns a value (between 0 and 1) describing the similarity of the object to that model in terms of the selected feature (see Table 1).

Function name	**Value returned** ($\in [0,1]$)	**Weight**
A*rea*	scale adjusted area	5
P*erimeter*	scale adjusted perimeter	5
M*oment*	first order invariant moments	10
R*ed*-**H***istogram*	distribution of red component	25
B*lue*-**H***istogram*	distribution of blue component	25
G*reen*-**H***istogram*	distribution of green component	25
G*ray*-**H***istogram*	distribution of gray component	25
H*ue*	distribution of image hue	25
S*aturation*	distribution of image saturation	25
L*uminosity*	distribution of image luminosity	25
M*ajor* **A***xis* **L***ength*	scale adjusted major axis length	25
C*ross* **S***ection* **D***imension*	cross-sectional widths	50
C*ross* **S***ection* **G***rayscale*	cross-sectional grayscale	50
S*hape*	shape of object contour	100

Table 1. Semantics of the vision functions.

Every function takes as input parameters a reference to an object and a reference to a model; it returns the similarity between the object and the model (1 being *maximum similarity*, 0 meaning *no similarity*). For example, **A***rea*(O_1,m_3) returns a value of similarity between object O_1 and model m_3, referred to as feature *Area*. The *weight* parameter in Table 1 pertains to the cost of using a given feature function: it is an empirical measure of its complexity, inversely proportional to the speed of execution (the *heavier* the function, the *longer* it is to execute on an object). In this paper, both the usefulness of the description of the function and its speed for evaluation are important details. The subset of functions used in this investigation was small in order to lighten the test procedure.

4.2 Building the prediction tree

We present here the grammar used to generate the possible solutions for the **GP** and the genetic operators to use with the **GP** algorithm, as well as a justification for these choices.

Grammar. Each solution to the problem is a program, which can be represented by a tree. This tree is the graphical representation of the expressions generated by the grammar as presented in Table 2.

Node	::	CondExp
	|	Func
	|	ClassifOp
CondExp	::	IF-THEN (Cond,Node)
	|	IF-THEN-ELSE (Cond,Node,Node)
Cond	::	(CondOp, Func, Func)
Func	::	(BinaryArithOp, Func, Func)
	|	(UnaryArithOp Func)
	|	IMFunction
	|	Node
	|	FFunction
NUMCONST	::	value in $\mathcal{R}$
IMFunction	::	MODMAX(FFunction)
	|	MODMIN(FFunction)
ClassifOp	::	VOTE(ModelId, VoteValue)
	|	VOTE(IMFunction, VoteValue)
CondOp	::	GTE | LTE | GT | LT | EQ
BinaryArithOp	::	+ | - | / | *
UnaryArithOp	::	LOG | LOG2 | EXP
ModelId	::	IMFunction
	|	NUMCONST
VoteValue	::	NUMCONST
FFunction	::	*functions from Table* 1

Table 2. Grammar for a solution

All symbols presented in the grammar have a straightforward meaning, although we wish to emphasize the following points:

1. **Feature functions**: referred to as *FFunctions* in Table 2, are in fact the functions presented in Table 1.
2. **Inter-model Comparison**: MODMAX(O_p, $\mathcal{F}$) evaluates the function $\mathcal{F}$ (from Table 1) against all the models and returns the model for which this evaluation is maximum (in other words, the model that *best* describes O_p, in terms of $\mathcal{F}$):

   $$m_n \text{ is such that } \forall m_t,$$
   $$t \neq n \Rightarrow \mathcal{F}(O_p, m_n) \geq \mathcal{F}(O_p, m_t)$$

   MODMIN(O_p, $\mathcal{F}$) has the same behavior, but instead returns the model that *worst* describes O_p, in terms of $\mathcal{F}$:

   $$m_n \text{ is such that } \forall m_t,$$
   $$t \neq n \Rightarrow \mathcal{F}(O_p, m_n) \leq \mathcal{F}(O_p, m_t)$$
3. ***Vote* nodes**: for each object O_p being evaluated, a tree will issue a prediction concerning whether it is considered an instance of a model M. This output is called a *vote*; it is syntactically noted as vote(V, M), and it can take three values:
   (a) V = 1: object O_p *belongs* to class M
   (b) V = 0: object is *unknown* (equivalent to no vote having occurred)
   (c) V = -1: object O_p *does not* belong to class M

For example, we present in Figure 1 a string that is recognized by the grammar (that is *syntactically correct*), along with its equivalent decision tree.

4.3 Genetic operators

We discuss in the following section the genetic operators of mutation, crossover and selection, and the fitness function that imposes constraints on the evolutionary path of the algorithm. We are showing how these operators are defined in order to respect the grammar presented in Table 2.

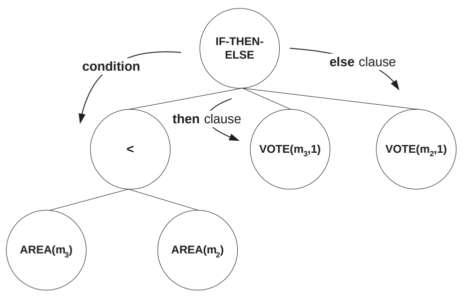

Fig. 1. Prediction tree example. String: IF-THEN-ELSE (($<$ AREA(m_3) AREA(m_2)) VOTE(m_3, +1) VOTE(m_2, +1))

Mutation. Mutation is a random change to a solution. The position of the change in the solution (node) and the change itself are both randomly determined. Three different types of mutation are defined:

- *Mutation of node and sub-tree*. A given node in the solution (and all its children) can be replaced by an entirely different sub-tree; a node is selected at random in the solution and is replaced with another randomly generated branch that respects the grammar.
- *Mutation of node*: any given node can be changed to any other node which shares the same parameters. A node is replaced with another of similar type.
- *Mutation of parameters*: A node's parameters are replaced; these parameters depend on the definition of the actual node. For example, the two parameters of a *Vote* node can be changed (the model to which the vote is applied, and its value), but it should be clear that each node type is particular. The only existing restriction is that the changes respect the grammar.

Crossover. Crossover is the main process through which subsequent populations of solutions are reproduced and models genetic exchange in sexual reproduction. Two solutions reproduce by exchanging sub-trees from randomly selected positions in each, within the context of the grammar. In short, we implemented the crossover operator as defined by J. Koza in [5]; the only particularity being that the newly generated trees (the offspring) must comply with the grammar in Table 2.

Fitness function. The fitness measure of each solution could be based on both the classification accuracy and the execution speed of the solution. In our current implementation, we elected to use only the classification performance to evaluate fitness, although the use of the weights (described in Table 1) was explored. Therefore, the fitness of an individual of the population (f_i) is defined using Eq. (1):

$$f_i = \frac{\text{number of correctly classified objects}}{\text{number of objects}} \tag{1}$$

Selection. The selection of the parent solutions occurs with a probability determined by the fitness of the solutions using Eq. (2)

$$p_i = \frac{f_i}{\sum\limits_{n=0}^{N} f_n} \tag{2}$$

where p_i is the *probability of selecting solution i*, f_i the *fitness of solution i*, and N is the *population size*.

5 EXPERIMENTAL RESULTS

As presented the feature selection method consists in generating the best classifier for a problem, and then choosing as *the best* features those that were taken into account by the classifier. In this section we show a few experiments designed to evaluate the performance of the evolved classifier functions, in terms of their ability to perform *detection* (subsection 5.1). The training and testing sets used for our experiments were composed of images of several hundred grains and legumes (hard wheat, soft wheat, buckwheat, linseed, peas, lentils, and others) for tests 1 and 3 and of dozens of coins (canadian pennies, nickels, dimes, quarters) for test 3. The sets are shown on the last page in Figures 4, 5, 6 and 7. It is important to note that a training set consists of many example images of the objects. The set should include both positive and negative examples of every class for which the system is being trained as well as some examples of objects considered as unknown, i.e. not pertaining to any of the selected classes. An expert must then validate the set of examples, i.e. the correct classifications must be determined and recorded for each of the example objects.

5.1 Classifier performance

Three separate experiments were performed in order to test the ability of the method to evolve classifiers under varying conditions. They were presented to one or more of the following classification methods, sharing the same training and test sets:

1. **the GP method**: the **GP** algorithm was allowed to generate a classifier; with the following parameters held constant: *Population size* = 1,000; *Elitism rate* = 1; *Probability of mutation* = 30; *Tree maximum depth* = 10; and *Number of generations* = 50
2. **a human group**: Ten (10) untrained people were allowed to observe the differences between the various species of grain in the training sets for one (1) minute. They were then sequentially shown the test sets for performing the classification task.
3. **the *K*-NN algorithm**: *K*-**N**earest **N**eighbors classifier system (*K*-**NN**) (as described in [17], pages 174–200). In each of the 3 experiments, the value of minimum acceptance and the value of sampling were expressed as ($<$examples-to-sample$>$, $<$min-accept.$>$).

Experiment 1 - three classes. In this experiment, we aimed at obtaining a function to classify objects of three classes, with a training set composed only of objects appearing in the test set. This experiment was divided in 4 parts, each with its own training and test sets. For each, a classifier tree was generated, but we only present the resulting tree for the first part due to space limitations.

part 1: The test set for this instance is composed of 293 objects (Fig. 4a). The tree generated by the **GP** method is shown in Fig. 2; its classification performance, along with human group and the *K*-**NN** method, are shown in Table 3.

From the functions defined in Table 1, the tree in Fig. 2 used *Perimeter, Hue, Major Axis Length, Cross Section Dimension and Cross Section Grayscale* as feature-nodes. This means that only 5 of the 14 feature functions have to be computed when using this classifier; this, in terms of the weights presented in Table 1, represents only 155 out of 420 (37%). A speed-up of 63%. It is worthwhile mentioning that the subset of functions (14 out of 100+) was small and that speed-up would dramatically increase as the number of functions included increases.

Method	Accuracy	
GP method		94%
Human	best	69.92%
	avg.	54.44%
K-**NN**	(3,1)	92.9%
	(10,5)	94%

Table 3. Classification performances (Exp. 1-1)

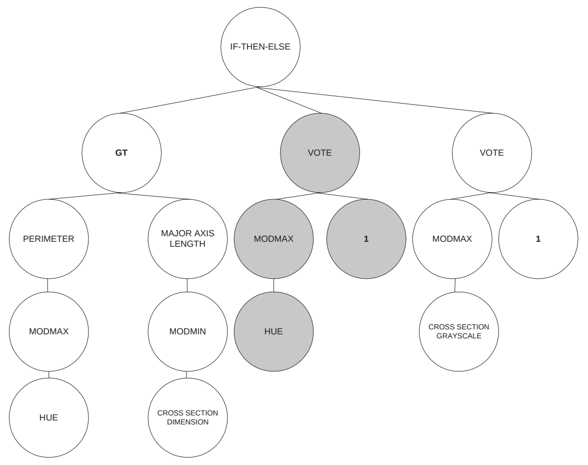

Fig. 2. GP-tree solution trained on training set 1
Classification accuracy on training set 1: 92.9% and on test set 1: 94.0%

part 2: The test set for this instance of experiment 1 is composed of 266 objects (Fig. 4b). From the functions defined in Table 1, the solution tree used *Perimeter, Hue, Green Histogram, Cross Section Dimension and Cross Section Grayscale* as feature-nodes. Only 5 of the 14 feature functions have to be evaluated when using this classifier; in terms of weights (Table 1) it is a speed-up of 63%, as the weights of the function in the **GP**-classifier represents only 155 out of 420. The classification performances of the human group and the *K*-**NN**, along with the **GP**-method, are shown in Table 4.

Method	Accuracy	
GP method		98.1%
Human	best	97.73%
	avg.	94.08%
K-**NN**	(3,1)	100%
	(10,5)	100%

Table 4. Classification performances (Exp. 1-2)

part 3: The test set for this instance of experiment 1 is composed of 333 objects (Fig. 4c). The classification accuracy on the training set (Fig. 4c) was 93.0%, and on the test set (Fig. 5c): 90.7%. From the functions defined in Table 1, the classification tree used *Major Axis Length, Hue, Green Histogram, Red Histogram, Blue Histogram, Gray Histogram, Moment, Luminosity, Saturation, Cross Section Dimension and Cross Section Grayscale* as feature-nodes. This means that 11 of the 14 feature functions have to be computed when using this classifier; this, in terms of the weights expressed in Table 1, represents 310 out of 420 (74%), a speed-up of 26%. The classification performances of the human group and the *K*-**NN**, along with the **GP**-method, are shown in Table 5.

part 4: The test set for this instance of experiment 1 is composed of 305 objects (Fig. 4d). The classification accuracy on the training set was 97.0%, and on the test set: 94.0%. From the functions defined in Table 1, the classification tree used *Green Histogram, Gray Histogram, Moment, Perimeter, Shape, Luminosity, Saturation and Cross Section Dimension* as feature-nodes. This means that only 8 of the 14 feature functions have to be evaluated when using this classifier; this, in terms of the weights expressed in Table 1, represents only 265 out of 420 (63%). A speed-up of 37%. The classification performances of the human group and the *K*-**NN**, along with the **GP**-method, are shown in Table 6.

Method	Accuracy	
GP method		90.7%
Human	best	96.24%
	avg.	81.43%
K-**NN**	(3,1)	90.23%
	(10,5)	89.47%

Table 5. Classification performances (Exp. 1-3)

Method	Accuracy	
GP method		94%
Human	best	100%
	avg.	89.92%
K-**NN**	(3,1)	99.15%
	(10,5)	98.29%

Table 6. Classification performances (Exp. 1-4)

Experiment 2 - four classes. The second experiment was designed to demonstrate the classifier's ability to distinguish between objects with few distinctive features and in which the training set and test set images were obtained using different acquisition means (scanner vs. camera). Four different denominations of coins were used for building the training and the test sets; the training set (Fig. 6a) was acquired using a flatbed scanner, while the test set (Fig. 6b) was acquired using a digital camera. The mean of acquisition had significant effect on the shadows and on the specular reflection off each of the coins. For the training of this experiment, the system reached 100% accuracy after 31 generations. The resulting solution (Fig. 3) was relatively compact and is shown in Fig. 3; the performance results are shown in Table 7.

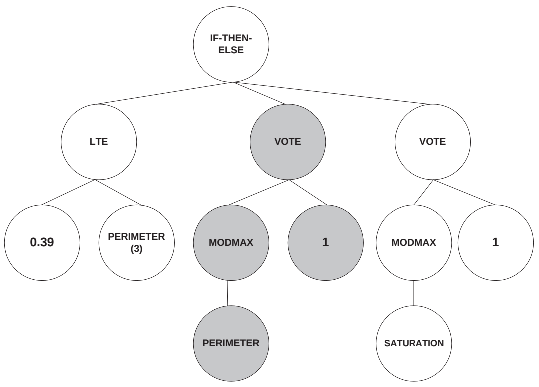

Fig. 3. GP solution to the coins problem

Method	Accuracy	
GP method		86.3%
K-**NN**	(3,1)	39.2%
	(10,5)	41.2%

Table 7. Results from experiment 2

As evident from Fig. 3, only the features *Perimeter and Saturation* were used in this solution, with a weight of only 30 out of 420 (an increase in speed of 93%).

Experiment 3 - six classes. This experiment is a "*six-class*" problem and is performed to further test for generality of the **GP** solution. A solution was evolved to accurately classify objects into six classes. The evolved solution was then applied to the same images used in experiment 1.

The purpose of this experiment was to demonstrate the generality of the method for multiple classes when the objects appearing in the test images belong only to some of the classes for which it was trained. The results are shown in Table 8. Note that this table presents the results when training for 6 classes (**GP**-6) as well as when training for only 3 classes (**GP**-3) while the test images contained objects from only 3 classes.

Image	Type	Accuracy	Image	Type	Accuracy
a	**GP**-6	78.95%	c	**GP**-6	88.72%
	GP-3	94.0%		**GP**-3	90.7%
	K-**NN**(3,1)	84.2%		*K*-**NN**(3,1)	90.2%
	K-**NN**(10,5)	76.7%		*K*-**NN**(10,5)	82.7%
b	**GP**-6	86.79%	d	**GP**-6	87.18%
	GP-3	98.1%		**GP**-3	94%
	K-**NN**(3,1)	87.7%		*K*-**NN**(3,1)	84.6%
	K-**NN**(10,5)	81.1%		*K*-**NN**(10,5)	87.2%

Table 8. Training and Test accuracy-Experiment 3

6 DISCUSSION

The **GP**-method performs an implicit reduction over the whole set of features, as shown by the results from the three experiments. We think this is significant when the runtime speed is as important as the actual classification rate: if only a subset of the features are used in the "winning" solution of the GP method, only those features have to be computed and the overall classification time, for a rather large problem, would be considerably smaller. In this paper we tested this idea by generating classifiers for different problems and comparing them to other classification methods.

The **GP** method produces decision trees whose performances are comparable to that of *K*-**NN** (Tables 3, 4, 5, 6, 7 and 8). Although an important difference can be seen in the coin problem (Table 7), where the **GP**-method clearly outperforms *K*-**NN**, it is not clear that the **GP** method performance is superior for these classification problems. It is however interesting to note that, while the **GP** method performs a reduction of features in the training stage, the computational complexity of the *K*-**NN** method (lower-bounded at $O(dn^2)$ at best for d features and n prototypes [17]) causes its runtime to be considerably greater than that of the **GP** method. And this can be a great drawback in designing a real-time multicategory classification system. We think that this would have been even more predominant had we selected a larger number of functions to describe each object.

Our experiments did not show clearly that our method is superior to a trained human being. It is implicit that these tasks of classification are better suited for machines than for humans, but it is our belief that we have to set up more experimentation in order to show how and when the **GP** method is superior to a well trained expert. The **GP**-method shows a more *reliable* performance for the classification problems: its classification performance for the test sets is very similar to that achieved for the training set. As such, the time invested in the decision tree synthesis gives the solution the ability to better generalize the solutions; an interesting extension of this observation would be to measure the difficulty of different classification problems and to observe the performance depending on this difficulty. This would provide a better understanding of which method is more effective to use with which type of problems.

7 CONCLUSIONS AND FUTURE WORK

The automatic generation of a solution alone brings the detection process to an entirely new level in that any number or combination of solutions can be tested without any user intervention. Furthermore, entirely non-intuitive solutions are equally tested without any bias, often leading to effective solutions that would possibly have been overlooked by a user. Another advantage of this method is that it produces solutions which are both readable and editable. This allows the user to understand the generated solution thus leading to developmental improvements and also permits editing of the solution in order to further enhance performance, or to combine solutions for added generality.

We consider this to be a first step towards a full domain-independent object detection system. We have to continue moving towards this objective, namely by exploring the influence of the genetic parameters in the generation of solutions and by doing a more comprehensive experimentation.

References

1. Gader, P.D., Miramonti, J.R., Won, Y., Coffield, P.: Segmentation free shared weight neural networks for automatic vehicle detection. Neural Networks **8(9)** (1995) 1457–1473
2. Waxman, A.M., Seibert, M.C., Gove, A., Fay, D.A., Bernandon, A.M., Lazott, C., Steele, W.R., Cunnigham, R.K.: Neural processing of targets in visible, multispectral ir and sar imagery. Neural Networks **8(7/8)** (1995) 1029–1051
3. Won, Y., Gader, P.D., Coffield, P.: Morphological shared-weigth networks with applications to automatic target recognition. IEEE Transactions on Neural Networks **8(5)** (1997) 1195–1203
4. Zhang, M., Ciesielski, V., Andreae, P.: A domain independent approach to multiclass object detection using genetic programming. EURASIP Journal of Applied Signal Analysis **Special Issue on Evolutionary Techniques** (2003)
5. Koza, J.: Genetic Programming-On the programming of Computers by Means of Natural Selection. MIT Press, Cambridge, London (1992)
6. Bernier, T.: An adaptable recognition system for biological and other irregular objects. Phd thesis, McGill (2001)
7. Bernier, T., Landry, J.A.: A new method for representing and matching shapes of natural objects. Pattern Recognition **36(8)** (2003) 1711–1723
8. Mitchell, T.: Machine Learning. McGraw Hill, New York (1996)
9. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming An Introduction. Morgan Kauffman Publishers, San Francisco (1998) An interesting introduction to genetic programming.
10. Langley, P.: Elements of Machine Learning. Morgan Kauffman, San Francisco,CA (1996)
11. Bentley, P.J.: Evolutionary Design By Computers. Morgan Kaufmann, San Francisco, CA (1999)
12. Pei, M., Goodman, E.D., Punch, W., Ding, Y.: Further research on feature selection and classification using genetic algorithms for classification and feature extraction. In: Proc. of the Annual Meeting of the Classification Society of North America, Denver, CO. (1995)
13. Daida, J., Hommes, J., Bersano-Begy, T., Ross, S., Vesecky, J.: Algortihm discovery using the genetic programming paradigm: Extracting low-contrast curvilinear features from sar images of artic ice. In P.J., A., Kinnear Jr., K., eds.: Advances in genetic Programming 2. MIT Press, Cambridge, MA. (1996) 417–442
14. Gruau, F.: Automatic definition of modular neural networks. Adaptiv Behavior **3(2)** (1195) 151–183

15. Poli, R.: Genetic programming for image analysis. In Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., eds.: Genetic Programming 1996: Proceedings of the First Annual Conference. MIT Press (1996) 363–368
16. Poli, R.: Genetic programming for feature detection and image segmentation. In Fogarty, T.C., ed.: Evolutionary Computing. Springer-Verlag (1996) 110–125
17. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. 2nd edn. John Wiley and sons, Inc., New York (2001)

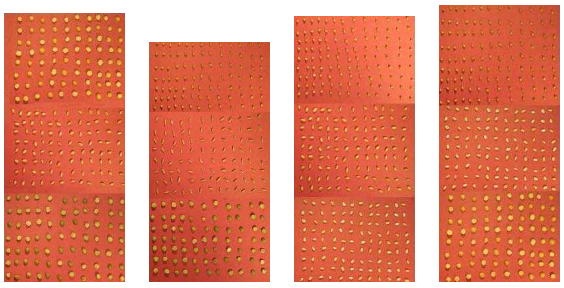
(a) (b) (c) (d)

Fig. 4. Training sets-Experiment 1

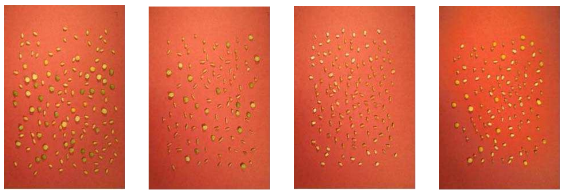
(a) (b) (c) (d)

Fig. 5. Test sets-Experiment 1 and 3

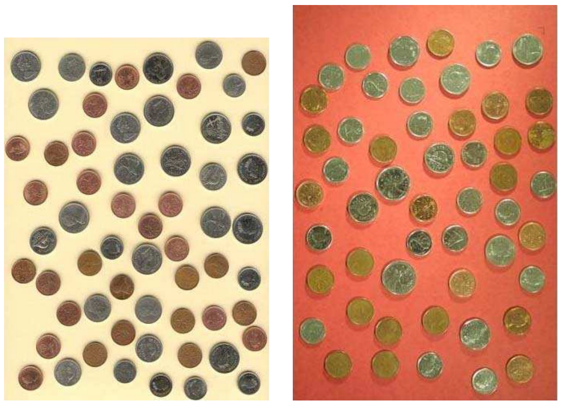
(a) (b)

Fig. 6. Training and Test sets-Experiment 2

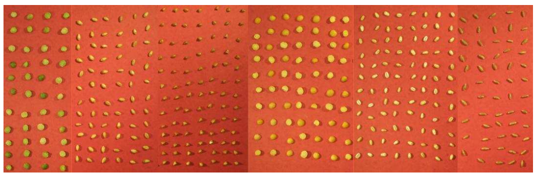

Fig. 7. Training set-Experiment 3