DOI: 10.1111/j.1467-8659.2011.02043.x

Pacific Graphics 2011
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin
(Guest Editors)

Volume 30 (2011), Number 7

Intelligent GPGPU Classification in Volume Visualization: A framework based on Error-Correcting Output Codes

S. Escalera[1,2], A. Puig[1], O. Amoros[1], and M. Salamó[1]

[1]Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Spain
[2]Centre de Visió per Computador, Universitat Autònoma de Barcelona, Spain

Abstract

In volume visualization, the definition of the regions of interest is inherently an iterative trial-and-error process finding out the best parameters to classify and render the final image. Generally, the user requires a lot of expertise to analyze and edit these parameters through multi-dimensional transfer functions. In this paper, we present a framework of intelligent methods to label on-demand multiple regions of interest. These methods can be split into a two-level GPU-based labelling algorithm that computes in time of rendering a set of labelled structures using the Machine Learning Error-Correcting Output Codes (ECOC) framework. In a pre-processing step, ECOC trains a set of Adaboost binary classifiers from a reduced pre-labelled data set. Then, at the testing stage, each classifier is independently applied on the features of a set of unlabelled samples and combined to perform multi-class labelling. We also propose an alternative representation of these classifiers that allows to highly parallelize the testing stage. To exploit that parallelism we implemented the testing stage in GPU-OpenCL. The empirical results on different data sets for several volume structures shows high computational performance and classification accuracy.

Categories and Subject Descriptors (according to ACM CCS): http://www.acm.org/class/1998/ I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Object hierarchies

1. Introduction

Knowledge expressiveness of scientific data is one of the most important visualization goals. The abstraction process the final user should carry out in order to convey relevant information in the underlying data is normally a difficult task. User has to mentally process a large amount of data contained in several hundreds of slices to find features, boundaries between different structures, regions surrounding pathological areas, semantic structures, and so on. During the last few decades new means of outlining significant features are being developed to gather visual information contained in the data, such as ghost views, Focus+Context approaches [APT08], importance-driven visualizations and automatic viewpoint selections. Some of these methods [VKG05] [KSW06] are useful for the exploration of pre-classified data sets as well as non-classified ones. However, most of them [BHW*07, KBKG08] require to previously define the structures of interest.

In volume rendering literature, many papers addressed classification by directly associating optical and importance properties to the different data values considering their belonging to a particular structure [PLB*01]. Most of them are based on the edition of transfer functions (TF) [BPS97, KD98]. One-dimensional TFs only take into account scalar voxel values, but in some cases they could fail at accurately detecting complex combinations of material boundaries. Multi-dimensional TFs consider vectorial values or combinations of local measures of scalar values (e.g. position, statistical signatures, or derivatives [KD98]). However, the design complexity and the memory requirements increase with the TFs dimensionality. In general, the user coherently assigns similar optical properties to data values corresponding to the same region. Selection of regions or structures is indirectly defined by assigning to zero the opacity since totally transparent samples do not contribute to the final image. Then, the manual TF's definition even by skilled users becomes complicated. In this sense, many works have focused on developing user friendly interfaces that make this definition more intuitive. Special emphasis has been done in the design of interfaces that deal with the definition of a TF or partially automatize it [TM04, KKH01, MAB*97]. Nevertheless, to recognize semantic structures that apply to iden-

tify additional semantic information requires more sophisticated techniques.

Automatic and user-guided segmentation strategies based on image processing are used to obtain classified data sets. Recently, some preliminary works using learning methods have been published based on data driven and on image-driven classification. These classification methods provide users with a high level of information about data distribution and about the final visualization. Supervised methods such as bayesian networks, neural networks [TLM03], decision trees [FPT06] and non-supervised methods [TM04] have been applied in different user interfaces of volume applications. For instance, in [GMK*92], clustering-based supervised and non-supervised learning methods are compared for classifying magnetic resonance data. An integration of interactive visual analysis and machine learning is used in an intelligent interface to explore huge data sets in [FWG09]. Still, the data driven classification problem as a pattern recognition process is an open issue that has been treated from different points of view: template matching, statistical, syntactic or structural, and neural [JDM00]. For instance, supervised statistical learning deals with the classification task by modeling the conditional probability distribution of the different pre-labelled data sample features.

Different Machine Learning (ML) approaches have been recently implemented using GPGPU for binary classifications in image processing applications. Clustering strategies and the computation of a *k*-nearest neighbor similarity classifier is presented in [GDB08]. A Geometrical Support Vector Machine classifier has also been implemented using GPGPU [HWS10]. It extends different GPGPU implementations for Neural Networks [YSMR10]. Adaboost is also a widely applied classifier. Based on a *weak classifier*, Adaboost defines an additive model combining simple weak classifiers to define a strong binary classifier with high generalization capability. Given its inherent parallel structure, its high performance, and its simplicity in order to train – Adaboost does not require tuning classifier parameters, Adaboost has a high potential for GPU applications.

Most of the previous approaches are binary by definition –they only learn to split from two possible labels[†]. In order to deal with multi-class labelling, they need to be combined in some way, for example, by means of a voting or a committee process. In this scope, the Error-Correcting Output Codes (ECOC) is widely applied as a general framework in the ML community in order to deal with multi-class categorization problems. The ECOC framework allows to combine any kind of classifiers, improving classification accuracy by correcting errors caused by the bias and the variance of the learning algorithm [DK95].

[†] Note that we use the terms classification and labelling indistinctly to refer the assignment of labels to data samples.

In this paper, we propose a general framework of supervised statistical classification methods to label on-demand multiple regions of interest (see Fig. 1). This provides an interactive classification/segmentation generic framework that could be used in several real applications, and it can be used with any existing classification/segmentation state-of-the-art method. The framework is composed by a pre-learning stage and an on-demand testing stage included in the renderer. The learning step gets a subset of pre-classified samples to train a set of Adaboost classifiers, which are codified as TFs, and combined in an ECOC design. Each classifier encodes a set of relevant properties in a 1D texture. We deal with several properties, such as density, gradient, space location, etc. of the original data without increasing the dimensionality of the classifier. Next, the testing stage multi-classifies and labels a subset of volume classes based on user interaction. The label mapping defines clusters of the selected classes, and then it assigns optical propeties and importance values to the final output classes to be visualizated. The labels computed in the testing step in the GPU memory can be used in several volume visualization approaches: to skip non-selected regions, to help computing importance values to different context structures, or to select the focus of interest for automatic view selections, just to mention a few. The label mapping reduces the edition of the TFs by only assigning the optical properties of a label. In addition, the labels, as a TF, can be applied directly to the voxels values, or alternatively to the sampling points derived from the interpolation of the nearby voxels. Up to now, the complexity of the learning process and the testing step of a large amount of data do not allow their integration into an interactive classification stage of the user-interface volume pipeline. In this sense, our proposal improves the TF specification process by combining a pre-processing learning step and a rendering integrated GPU-based testing method.

In summary, this paper brings four contributions: First, the definition of a general framework for multi-classification of volumes based on the ECOC approach; Second, the use and parallelization of Adaboost as a case study of this general framework; Third, the computation of an on-demand adaptive classification to a subset of features of interest; Finally, the proposal of a GPGPU OpenCL implementation of the testing stage of the multi-classifier integrated into the final rendering. This work serves as a proof of concept for future embedding of the training step in the visualization pipeline. In this sense, online learning will be performed on-demand in the rendering. Results on diferent volume data sets of the current prototype show that this novel framework exploits GPU capabilities at the same time that achieves high classification accuracy.

The rest of the paper is organized as follow: Next section overviews the ECOC framework and the Adaboost classifier. Section 3 presents the general framework for multi-class volume labelling. Section 4 explains the GPGPU implemen-

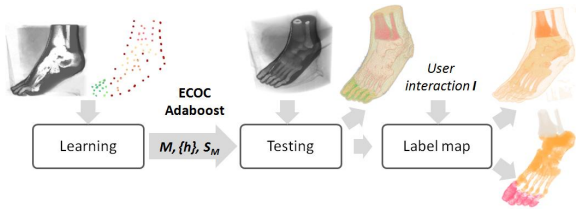


Figure 1: *Overview of the ECOC-based visualization framework.*

tation. Section 5 shows the experimental results, and finally, Section 6 concludes the paper.

2. Background

Our automatic volume labelling system bases on the combination of a set of trained binary classifiers. We consider the general ECOC framework to deal with multi-class labelling. As a case study, we use Adaboost to train the sets of binary classifiers. Next, we briefly review these two methodologies.

2.1. Error-Correcting Output Codes

Given a set of N classes (volume structures or regions with certain properties) to be learnt in an ECOC framework, n different bi-partitions (groups of classes) are formed, and n binary problems over the partitions are trained. As a result, a codeword of length n is obtained for each class, where each position (bit) of the code corresponds to a response of a given classifier h (coded by +1 or -1 according to their class set membership, or 0 if a particular class is not considered for a given classifier). Arranging the codewords as rows of a matrix, we define a *coding matrix* M, where $M \in \{-1,0,+1\}^{N\times n}$. Fig. 2(a) and (b) show a volume data set example and a coding matrix M, respectively. The matrix is coded using 15 classifiers $\{h_1,...,h_{15}\}$ trained using a few voxel samples for each class of a 6-class problem $\{c_1,...,c_6\}$ of respective codewords $\{y_1,...,y_6\}$. The classifiers h are trained by considering the pre-labelled training data samples $\{(\rho_1,l(\rho_1)),...,(\rho_k,l(\rho_k))\}$, for a set of k data samples (voxels in our case), where ρ is a data sample and $l(\rho_k)$ its label. For example, the first classifier h_1 is trained to discriminate c_1 against c_2, without taking into account the rest of classes. Some standard coding designs are one-versus-all, one-versus-one, and random [ASS02]. Mainly, they differ on the definition of the sub-groups of classes in the partitions of each binary problem. Due to the huge number of bits involved in the traditional coding strategies, new problem-dependent designs have been proposed [ETP*08]. These strategies take into account the distribution of the data in order to define the partitions of classes of the coding matrix M.

During the decoding or testing process, applying the n binary classifiers, a code X is obtained for each data sample ρ in the test set. This code is compared to the base codewords $(y_i, i \in [1,..,N])$ of each class defined in the matrix M, and the data sample is assigned to the class with the *closest* codeword (e.g. in terms of distance). In fig. 2(b), the new code X is compared to the class codewords $\{y_1,...,y_6\}$ using the Hamming Decoding [ASS02], $HD(X,y_i) = \sum_{j=1}^{n}(1 - \text{sign}(x^j \cdot y_i^j))/2$, where X^j corresponds to the j-th value of codeword X, and the test sample is classified by class c_1 with a measure of 0.5. The decoding strategies most widely applied are Hamming and Euclidean [ASS02], though other decoding designs have been proposed [EPR10].

2.2. Adaboost classifier

In this work, we train the ECOC binary classifiers h using Adaboost classifier. Adaboost is one of the main preferred binary classifiers in the ML community based on the concept of Boosting [FHT98]. Given a set of k training samples, we define $h_i \sim F_i(\rho) = \sum_{m=1}^{\mathcal{M}} c_m f_m(\rho)$, where each $f_m(\rho)$ is a classifier producing values ± 1 and c_m are constants; the corresponding classifier is $\text{sign}(F(\rho))$. The Adaboost procedure trains the classifiers $f_m(\rho)$ on weighed versions of the training sample, giving higher weights to cases that are currently misclassified [FHT98]. Then, the classifier is defined to be a linear combination of the classifiers from each stage. The binary Discrete Adaboost algorithm used in this work is shown in Algorithm 2.2. E_w represents expectation over the training data with weights $w = (w_1, w_2, .., w_k)$, and $1_{(S)}$ is the indicator of the set S (1 or 0 if S is or not satisfied). Finally, Algorithm 2.2 shows the testing of the final decision function $F(\rho) = \sum_{m=1}^{\mathcal{M}} c_m f_m(\rho)$ using Adaboost with Decision Stump "weak classifier". A Decision Stump is a simple directional threshold over a particular feature value. Each Decision Stump f_m fits a threshold value T_m and a polarity (directionaly over the threshold) P_m over the selected m-th feature. In testing time, ρ^m corresponds to the value of the feature selected by $f_m(\rho)$ on a test sample ρ. Note that c_m value is subtracted from $F(\rho)$ if the classifier $f_m(\rho)$ is not satisfied on the test sample. Otherwise, positive values of c_m are accumulated. Finally decision on ρ is obtained by $\text{sign}(F(\rho))$.

Algorithm 1 Discrete Adaboost training algorithm.

1: Start with weights $w_i = 1/k, i = 1,..,k$.
2: Repeat for $m = 1,2,..,(M)$:
(a) Fit the classifier $f_m(\rho) \in -1,1$ using weights w_i on the training data.
(b) Compute $\text{err}_m = E_w[1_{(l(\rho)\neq fm_{(\rho)}]}, c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
(c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(l(\rho_i)\neq fm_{(\rho_i)}]}, i = 1,2,..,k$, and normalize so that $\sum_i w_i = 1$.
3: Output the classifier $F(\rho) = \text{sign}[\sum_{m=1}^{\mathcal{M}} c_m f_m(\rho)]$.

3. General framework for multi-class volum labelling

Here, we present our automatic system for multi-class volume labelling. The system performs the following stages: a)

Algorithm 2 Discrete Adaboost testing algorithm.

1: Given a test sample ρ
2: $F(\rho) = 0$
3: Repeat for $m = 1, 2, .., \mathcal{M}$:
(a) $F(\rho) = F(\rho) + c_m(P_m \cdot \rho^m < P_m \cdot T_m)$;
4: Output sign$(F(\rho))$

All-pairs ECOC multi-class learning, b) ECOC submatrix definition, c) Adaptive decoding, and d) Label mapping.

3.1. All-pairs multi-class learning

Given a set of pre-labelled samples for each volume structure, we choose the one-versus-one ECOC design of $N(N-1)/2$ classifiers to train the set of all possible pairs of labels. An example of a one-versus-one ECOC coding matrix for a 6-class foot problem is shown in Fig. 2(a) and (b). The positions of the coding matrix M coded by +1 are considered as one class for its respective classifier h_j, and the positions coded by -1 are considered as the other one. For example, the first classifier is trained to discriminate c_1 against c_2; the second one classifies c_1 against c_3, etc., as follows:

$$h_1(x) = \begin{cases} 1 & \text{if } X \in \{c_1\} \\ -1 & \text{if } X \in \{c_2\} \end{cases}, .., \quad h_{15}(x) = \begin{cases} 1 & \text{if } X \in \{c_5\} \\ -1 & \text{if } X \in \{c_6\} \end{cases} \quad (1)$$

The selection of the one-versus-one design has two main benefits for our purpose. First, though all pairs of labels have to be split in a one-versus-one ECOC design, the individual problems that we need to train are significantly smaller compared to other classical ECOC classifiers (such as one-versus-all). As a consequence, the problems to be learnt are usually easier since the classes have less overlap. Second, as shown in Fig. 2, considering binary problems that split individual labels allow us for combining groups of labels on-demand just by the selection of a subgroup of previouly trained classifiers, as we show next.

3.2. ECOC submatrix definition

Given a volume that can be decomposed into N different possible labels, we want to visualize in rendering time the set of labels requested by the user. For this purpose, we use a small set of ground truth voxels described using spatial, value, and derivative features to train the set of $N(N-1)/2$ Adaboost binary problems that defines the one-versus-one ECOC coding matrix M of size $N \times n$. Then, let us define the interaction of the user as the set $I = \{I_0, .., I_z\} = \{\{c_i, .., c_j\}, ..\{c_k, .., c_l\}\}$, where $I, |I| \in \{1, .., N\}$ is the set of groups of labels selected by the user, and I_0 contains the background (always referred as c_1) plus the rest of classes not selected for rendering, $I_0 = \{c_i\}, \forall c_i \notin \{I_1, .., I_z\}, \bigcup_{c_i \in I} = \{c_1, .., c_N\}, \bigcap_{c_i \in I} = \emptyset$. Then, the submatrix $S_M \in \{-1, 0, +1\}^{N \times Z}$ is defined, where $Z \leq n$ is the number of classifiers selected from M that satisfies the following constraint,

$$h_i | \exists j, M_{ji} \in \{-1, 1\}, c_j \in I \setminus I_0 \quad (2)$$

For instance, in a 6-class problem of 15 one-versus-one ECOC classifiers (see Fig. 2(b)), the user defines the interaction $\{c_3, c_6\}$, resulting in the interaction model $I = \{\{c_1, c_2, c_4, c_5\}, \{c_3, c_6\}\}$ in order to visualize two different labels, one for background and other one for those voxels with label c_3 or c_6. Then, from the original matrix $M \in \{-1, 0, +1\}^{6 \times 15}$, the submatrix $S_M \in \{-1, 0, +1\}^{6 \times 9}$ is defined, as shown in Fig. 2(c). Note that the identifier i of each classifier h_i in S_M refers to its original location in M.

3.3. Adaptive decoding

The proposed ECOC submatrix S_M encodes the minimum required number of binary classifiers from a one-versus-one coding matrix M to label the sets of structures defined by the user in I. However, this novel definition of ECOC submatrices requires a readjustment of the decoding function δ applied. For instance, if we look at the matrix M of Fig. 2(b), one can see that each row (codeword) of M contains the same number of positions coded by zero. If one applies a classical Hamming decoding, obviously a bias in the comparison with a test and a matrix codeword is introduced to those positions comparing membership $\{-1, 1\}$ to 0. Note that the zero value means that the class has not been considered, and thus, it makes no sense to include a decoding value for those positions. However, since in the one-versus-one design all the codewords contain the same number of zeros, the bias error introduced for each class is the same, and thus, the decoding measurements using classical decoding functions are comparable. In our case, this constraint does not hold. Look at the submatrix S_M of Fig. 2(c). The selection of submatrices often defines coding matrices with different number of positions coded to zero for different codewords. In order to be able to successfully decode a submatrix S_M we take benefit from the recent Loss-Weighted proposal [EPR10], which allows the decoding of ternary ECOC matrices avoiding the error bias introduced by the different number of codeword positions coded by zero. The Loss-Weighted decoding is defined as a combination of normalized probabilities that weights the decoding process. We define a weight matrix M_W by assigning to each position of the codeword codified by $\{-1, +1\}$ a weight of $\frac{1}{n-z}$, being z the number of positions of the codeword coded by zero. Moreover, we assign a weight of zero to those positions of the weight matrix M_W that contain a zero in the coding matrix M. In this way, $\sum_{j=1}^{n} M_W(i, j) = 1, \forall i = 1, ..., N$. We assign to each position (i, j) of a performance matrix H a continuous value that corresponds to the performance of the classifier h_j classifying the samples of class c_i as shown in eq. 3. Note that this equation makes H to have zero probability at those positions corresponding to unconsidered classes. Then, we normalize

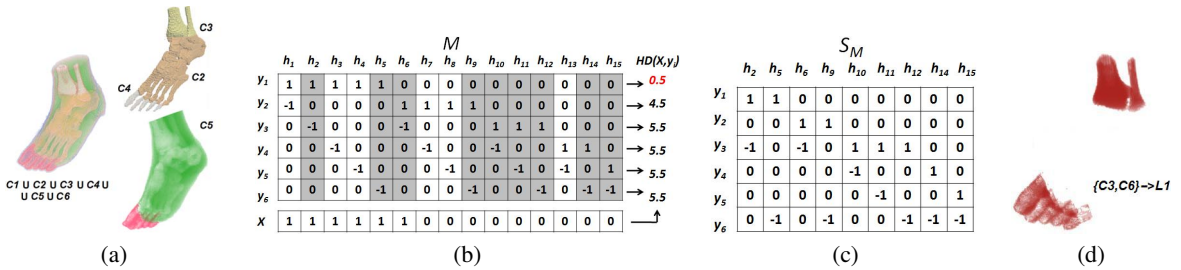


Figure 2: *(a) True labels for a foot volume of six classes; (b) One-versus-one ECOC coding matrix M for the 6-class problem. An input test codeword X is classified by class c_1 using the Hamming Decoding; (c) Submatrix S_M defined for an interaction set $I = \{\{c_1,c_2,c_4,c_5\},\{c_3,c_6\}\}$; (d) Visualization in the new label space.*

each row of the matrix H so that M_W can be considered as a discrete probability density function (eq. 5). In fig. 3, a weight matrix M_W for a 3-class toy problem of four classifiers is estimated. Fig. 3(a) shows the coding matrix M. The matrix H of Fig. 3(b) represents the accuracy of the classifiers testing the instances of the training set. The normalization of H results in a weight matrix M_W shown in Fig. 3(c). Once we compute the weight matrix M_W, we include this matrix in a loss-function decoding formulation, $L(\theta) = \mathbf{e}^{-\theta}$, where θ corresponds to $y_i^j \cdot f(\rho, j)$, weighted using M_W, as shown in eq. 6. The summarized algorithm is shown in table 3.3.

Loss-Weighted strategy: Given a coding matrix M,
1) Calculate the performance matrix H,

$$H(i,j) = \frac{1}{m_i}\sum_{k=1}^{m_i} \varphi(h^j(\rho_k^i), i, j) \quad (3)$$

$$\text{based on} \quad \varphi(x^j,i,j) = \begin{cases} 1, & \text{if} \quad X^j = y_i^j, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

2) Normalize H: $\sum_{j=1}^{n} M_W(i,j) = 1, \quad \forall i = 1,...,N$:

$$M_W(i,j) = \frac{H(i,j)}{\sum_{j=1}^{n} H(i,j)}, \quad \forall i \in [1,...,N], \quad \forall j \in [1,...,n] \quad (5)$$

3) Given a test data sample ρ, decode based on,

$$\delta(\rho,i) = \sum_{j=1}^{n} M_W(i,j) L(y_i^j \cdot f(\rho,j)) \quad (6)$$

Table 1: *Loss-Weighted algorithm.*

3.4. Label mapping

Given the submatrix S_M and the user interaction model I, after classification of a voxel ρ applying the Loss-Weighted decoding function, the obtained classification label $c_i, i \in \{1,..N\}$ is relabelled applying the mapping,

$$L_M(I,c_i) = \begin{cases} l_1 & \text{if } c_i \in I_1 \\ \ldots & \\ l_z & \text{if } c_i \in I_z \end{cases}$$

where $l_i, i \in \{1,..,z\}$ allows to assign RGBα or importance values to all voxels that belong to the corresponding selected classes in I_i. These functions are useful to provide flexibility to our framework in order to be applied in several visualization tasks, such as importance-driven visualizations or automatic viewpoint selections. As an example, applying the interaction model $I = \{\{c_1,c_2,c_4,c_5\},\{c_3,c_6\}\}$ for the 6-class problem of Fig. 2(a), we obtain the submatrix S_M of Fig. 2(c). Applying the Loss-Weighted decoding over S_W, and the mapping function $\{\{c_1,c_2,c_4,c_5\},\{c_3,c_6\}\} \longrightarrow \{0,1\}$, the new volume representation is shown n Fig. 2(d).

4. Implementation

Here, we describe our proposed classifier representation and analyze its parallelization possibilities.

4.1. Adaboost Look up table representation

We propose to define a new and equivalent representation of c_m and $|\rho|$ that facilitate the parallelization of the testing. We define the matrix $V_{f_m(\rho)}$ of size $3 \times (|\rho| \cdot M)$, where $|\rho|$ corresponds to the dimensionality of the feature space. First row of $V_{f_m(\rho)}$ codifies the values c_m for the corresponding features that have been considered during training. In this sense, each position i of the first row of $V_{f_m(\rho)}$ contains the value c_m for the feature $mod(i,|\rho|)$ if $mod(i,|\rho|) \neq 0$ or $|\rho|$, otherwise. The next value of c_m for that feature is found in position $i + |\rho|$. The positions corresponding to features not considered during training are set to zero. The second and third rows of $V_{f_m(\rho)}$ for column i contains the values of P_m and T_m for the corresponding Decision Stump. Note that in the representation of $V_{f_m(\rho)}$ we lose the information of the order in which the Decision Stumps were fitted during the training step. However, though in different order, all trained "weak classifiers" are represented, and thus, the final additive decision model $F(\rho)$ is equivalent. In our proposal, each "weak classifier" is codified in a channel of a 1D-Texture.

4.2. GPU and Multi-core CPU implementations

In order to make the application useful in practice, we aim to improve the testing stage execution times as much as possible. Our proposed Adaboost codification allows not only a coarse or fine grain parallelization, but also exposes a big amount of data independent operations, susceptible of being

$$M = \begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \end{bmatrix} \quad H = \begin{bmatrix} 0.955 & 0.955 & 1.000 & 0.000 \\ 0.900 & 0.800 & 0.000 & 0.000 \\ 1.000 & 0.905 & 0.805 & 0.805 \end{bmatrix} \quad M_W = \begin{bmatrix} 0.328 & 0.328 & 0.344 & 0.000 \\ 0.529 & 0.471 & 0.000 & 0.000 \\ 0.285 & 0.257 & 0.229 & 0.229 \end{bmatrix}$$

(a) (b) (c)

Figure 3: *(a) Coding matrix of four classifiers for a 3-class toy problem, (b) Performance matrix, and (c) Weight matrix.*

pipelined on the CPU. The critical section in our application is a triple *for-loop* that traverses all the data. Each iteration can be divided into two steps: a binary classification and a multi-class final decision. The former is devoted to perform the binary classification for each classifier. The latter is the multi-class decision made by using the ECOC matrix and the results of the initial step. Each iteration is data independent with each other so they can be executed concurrently.

One of the languages we have considered is ANSI C + OpenMP because not all computers have an OpenCL capable GPU. Additionally, depending on the configuration of the system, it is possible to have a Multicore-CPU that is faster than the GPU. OpenMP execution times can be affected by other applications or threads running in the system. For this reason, we have also considered testing Apple's GCD (Grand Central Dispatch) [GCD] API, that passes the management of the threads to the system and takes in account the work load of the CPU cores to reduce context switching. At the programmer level GCD substitutes the threads with queues. It allows the programmer to only focus on deciding which part of the code will be synchronous or not regarding the main program, using lots of lightweight queues that will feed a few system controlled threads.

On the other hand, we have also analyzed OpenCL due to the huge speedups it can deliver when running on GPU's and the possibility of OpenGL integration. Nevertheless, productivity continues to be a major concern. Then, CUDA could be an option but we preferred to stick with OpenCL's compatibility and portability between GPU's and CPU's. Besides effective GPU-memory bandwidth, one typical recommendation for better performance is to use as much *Work Items* as possible. However, this is true up to a certain extent. In our case, the best solution has been to maintain a correlation of one Work Item per one sample. It would have been feasible to use more than a Work Item per sample in the above mentioned steps in order to achieve a more fine-grained parallelization and a more scalable code. Nevertheless, the problem is that at the end of the two steps we only obtain a value per sample. As a consequence, the extra Work Items need to communicate through local memory and so, the performance is drastically reduced. In addition, maintaining the same Work Group and Work Item dimensions through all the process allowed us to code a single kernel, skipping lots of global and local memory reads and writes, and also integrate an initial gradient calculation step following the Micikevicius GPU-stencil algorithm [Mic09].

5. Simulations and Results

This section describes the experimental setup and shows the performance evaluation in terms of classification accuracy and execution time.

5.1. Setup

Data: We used three data sets, *Thorax* data set[‡] of size $400\times400\times400$ represents a MRI phantom human body; *Foot* and *Brain*[§] of sizes $128\times128\times128$ and $256\times256\times159$ are CT scans of a human foot and a human brain, respectively.

Methods: We use the one-versus-one ECOC design, Discrete Adaboost as the base classifier, and we test it with different number of decision stumps. For each voxel sample ρ, we considered eight features: x, y, z coordinates, the respective gradients, g_x, g_y, g_z, the gradient magnitude, $|g|$ and the density value, v. The system is compared in C++, OpenMP, GCD, and OpenCL codes.

Hardware/Software: The details of the different CPU and GPU hardware configurations used in our simulations are shown in Tables 2 and 3, respectively. The viewport size is 700×650. We used the MoViBio software developed by the GIE Research Group at the UPC university [mov].

Measurements: We compute the mean execution time from 500 code runs. For the accuracy analysis, we performed 50 runs of cross-validation with a 5% stratified samplings.

CPU's	AMD Phenom 2 955	Intel Core 2 Duo P8800	Intel Core i5 750
Frequency	3.2GHz	2.66GHz	2.66GHz to 3.2GHz
Cores	4	2	4
Threads per core	1	1	1
L3 cache	6MB	0MB	8MB
SSE level	4A	4.1	4.2

Table 2: *Different CPU configurations used for evaluation.*

GPU's	ATI	NVIDIA
Processing Elements	720	448
Stream or CUDA cores	144	448
Compute Units	9	14
Max PE per WI	5f / 0d	1f / 1d
PE available per CU	80f / 0d	32f / 4d
Warp size	64 Work Items	32 Work Items
Memory type	GDDR 5	GDDR 5
Global Memory size	1GB	1.28GB
Local Memory size	32KB	16KB or 48KB

Table 3: *Different GPUs architectures used for evaluation, where 'd' and 'f' stands for double and float, respectively.*

‡ http://www.voreen.org

§ http://www.slicer.org/archives

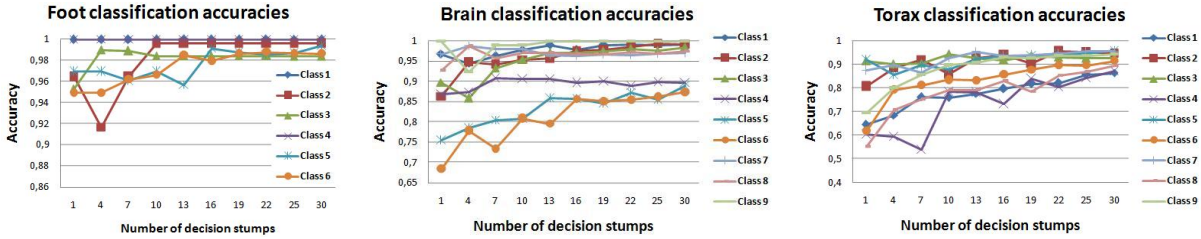
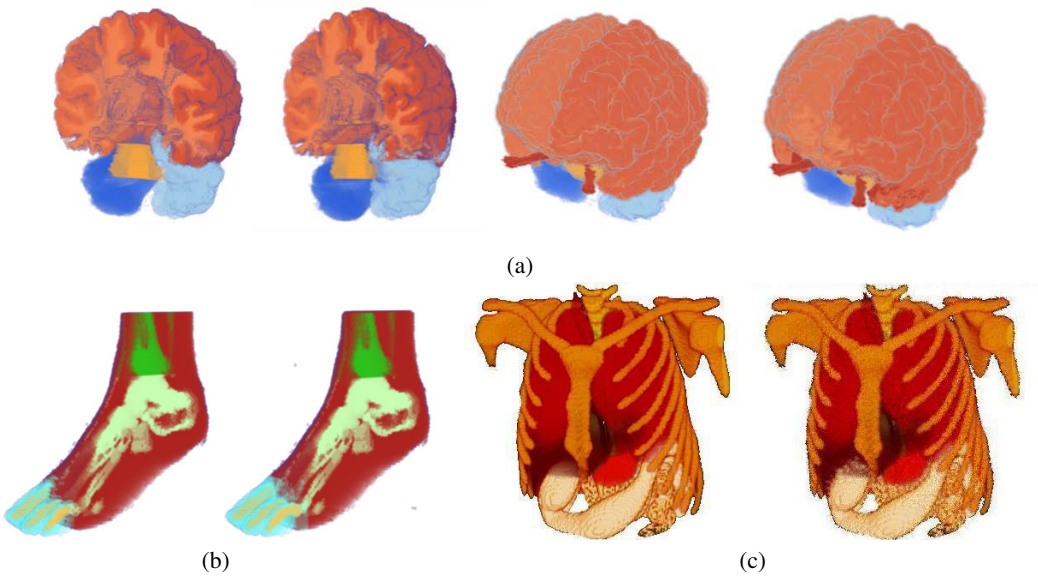


Figure 4: *Classification accuracies for the different data set structures and number of decision stumps in the Adaboost-ECOC framework.*



(a)

(b) (c)

Figure 5: *Comparison of the pre-labelled (left) and the classified data set (right) for (a) the* Brain*, (b) Foot, and (c)* Thorax *data sets, respectively.*

5.2. Classification accuracy analysis

Fig. 4 shows the classification accuracy of the framework for the data sets considering different number of decision stumps $\mathcal{M}$. The accuracies are shown individually for each volume structure. From Fig 4 one can see that even for different complexity of volume structures, most of the categories obtain upon 90% of accuracy.

In the case of the *Foot* data set, accuracy achieves near 100% for all the categories. In Fig. 5(b), we show the same section of the full pre-labelled *Foot* data set for six classes and the obtained classification with our proposal. In Fig. 6, we show selections and different shadings of the visualizations to demonstrate the flexibility of submatrix testing and the label mapping for different interaction models I.

In case of the *Brain* data set, we have trained nine classes of different structures of the brain that shares the same density and gradient values, achieving accuracies between 90% and 100%, where the lowest value is for the classe c_6, with values near 90%. The highest accuracy corresponds to the class c_9. In Fig. 5(a), we show two sections of the pre-labelled brain structures together with the results obtained by our multi-classifiers. Note that classification inaccuracies do not present significant artifacts in the rendering. In Fig. 7 we also present some illustrative visualizations of this multi-classified data set applying different user interactions I. Fi-

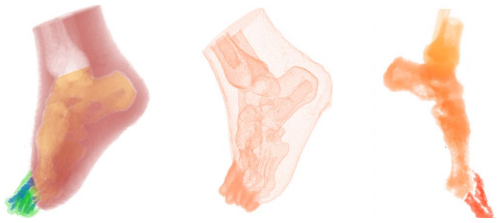


Figure 6: *Case study of the 6-class foot volume: c_1 is the background, c_2, c_3, c_4 are the bone structures of the palm, ankle and tooth, repectively, and c_5 and c_6 are the soft tissue of the palm/ankle and tooth, respectively. Color Plates show the user interaction sets (from left to right) $I = \{\{c_1\},\{c_2\},\{c_4\},\{c_5\},\{c_6\}\}$, $I = \{\{c_1\},\{c_2,c_3\},\{c_5\},\{c_6\}\}$, and $I = \{\{c_1\},\{c_2\},\{c_3\},\{c_4\}\}$, respectively.*

nally, Fig. 5(c) shows an example of a full Thorax volume classification for different structures[¶].

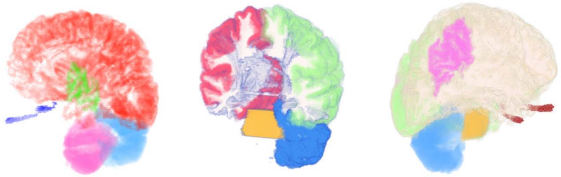


Figure 7: *Case study of the 9-class* Brain *volume: the classes corresponds to different structures of the brain such as hemispheres, optical nerves and cerebellum with different shading techniques.*

5.3. Execution time analysis

We compared the time performance of our GPU parallelized testing step in relation to the CPU-based implementations based on sequential C-code, OpenMP and GCD approaches. In table 4, for each implementation we show an averaged time of the 500 executions. We employ different sized data sets with several number of classes to be labelled and distinct user-selections. Our proposed OpenCL-based optimization has an average of speed up of 109x, 31.11x, and 31.14x over the sequential C-coded algorithm, the OpenMP and the GCD based algorithms, respectively. The results are remarkably promising considering that the whole tested data sets fit in the GPU memory card. We are planning to expand our implementation using well-known bricking techniques. In table 4, we can observe that time values are proportional to three features: the data set sizes, the number of classes,N, and the number of classifiers, Z, used for the selected classes. Specifically, the most influential feature in the time value is the size of the data set (see rows 6, 10 and 16 of table 4). Thus, including more classifiers hardly affect the time performance of the system. In addition, we obtain real-time in the *Foot* data set.

In Table 5 we observe time values of our OpenCL implementation for different parallel platforms: the ATI Radeon, the Nvidia Geforce GTX470, and the AMD Quad Core. We conclude that the best performance is achieved on the Nvidia graphic card whose time ranges from a few seconds in the worst case to some miliseconds for the best one. In addition, we executed the OpenCL code in the AMD CPU and we obtained speed ups of 1.02x, 1.06x, and 1.2x over the corresponding OpenMP implementation for 9 classes and 36 classifiers of the three data sets, *Foot*, *Brain* and *Thorax*, respectively. We observe that the greater dataset, the better OpenCL performance.

¶ Note that the proposed multi-class GPU-ECOC framework is independent of the classification/segmentation strategy applied. Thus, different feature sets as well as labelling strategies could be considered, obtaining differences in classification accuracy.

Data set	N	Sel. classes	Z	CPU	OpenMP	GCD	OpenCL
Foot	3	2	2	0.387	0.111	0.111	0.008
	3	3	3	0.577	0.165	0.165	0.002
	4	3	5	0.948	0.271	0.271	0.020
	4	4	6	1.139	0.325	0.325	0.038
	6	6	15	4.986	0.760	0.769	0.062
	9	9	36	8.319	1.787	1.777	0.091
Brain	9	2	15	39.396	11.190	11.177	0.358
	9	4	26	68.485	19.475	19.615	0.649
	9	6	33	87.558	24.947	24.875	0.848
	9	9	36	96.859	27.642	27.557	1.263
Thorax	2	2	1	26.849	7.604	7.600	2.694
	8	2	13	321.768	94.589	94.579	2.011
	8	4	22	564.577	160.801	160.784	3.532
	8	8	28	754.203	220.430	220.388	6.007
	9	7	35	923.225	260.751	259.489	5.955
	9	9	36	971.915	270.751	269.751	7.763

Table 4: *Testing step times in seconds of the different datasets using a submatrix S_M depending on the subset of selected classes. The whole matrix M is used when all classes are selected. The CPU, OpenMP and GCD times are tested on a Intel Core i5 750 (see Table 2). OpenCL times are tested on a GTX 470 (see Table 3).*

6. Conclusions and future work

In volume visualization, users usually face with the problem of manually defining regions of interest. To cope with this problem, we proposed an automatic framework for general multi-class volume labelling on-demand. The system is decomposed into a two-level GPU-based labelling algorithm that computes in time of rendering voxel labels using the ECOC framework with the Adaboost classifier. After a training step using few volume voxel features from different structures, the user is able to ask for different volume visualizations and optical properties. Additionally, to exploit the inherent parallelism of the proposal, we implemented the testing stage in C++, OpenMP, GCD, and GPU-OpenCL. Our empirical results indicate that the proposal have the potential to deliver worthwile accuracy and speeds up execution time. Overall, the proposal of this paper presents a novel, automatic, and general-purpose multi-decision framework that performs real-time computation.

Data set	N	Selected classes	Z	ATI	NVIDIA	AMD Quad
Foot	3	2	2	0.028	0.008	0.236
	4	3	5	0.066	0.020	0.495
Brain	9	2	15	1.498	0.358	7.114
	9	4	26	2.636	0.649	12.393
	9	6	33	3.403	0.848	16.047
	9	9	36	3.818	0.959	18.446
Thorax	8	2	13	8.750	2.011	56.374
	8	4	22	14.860	3.532	95.657
	8	6	27	18.340	4.467	118.978
	8	8	28	19.150	4.752	125.454

Table 5: *Different timings for different parallel architectures obtained by the ATI-Radeon, Nvidia GTX470 and AMD Phenom 2 955.*

There exists several points in this novel framework that deserve future analysis. We plan to analyze the effect of classifier generalization reducing the initial number of voxels in the ground truth training set, look for different feature space

representations including contextual information, study the use of different base strategies in the ECOC framework from both learning and segmentation points of view, as well as new parallelization optimizations. In order to avoid possible memory usage limitations for larger volume data sets, we will also analyze bricking strategies. Finally, we plan to embed the training step in the visualization pipeline so that online learning and forbidden steps can be performed on-demand in the rendering.

ACKNOWLEDGMENTS

This work has been partially funded by the projects TIN2008-02903, TIN2009-14404-C02, CONSOLIDER INGENIO CSD 2007-00018, by the research centers CREB of the UPC and the IBEC and under the grant SGR-2009-362 of the Generalitat de Catalunya.

References

[APT08] ABELLAN P., PUIG A., TOST D.: Focus+context rendering of structured biomedical data. In *EG VCBM 2008 : Eurographics workshop on visual computing for biomedicine* (2008), pp. 109–116. 1

[ASS02] ALLWEIN E., SCHAPIRE R., SINGER Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. *JMLR 1* (2002), 113–141. 3

[BHW*07] BURNS M., HAIDACHER M., WEIN W., VIOLA I., GROELLER E.: Feature emphasis and contextual cutaways for multimodal medical visualization. In *Eurographics / IEEE VGTC Symposium on Visualization 2007* (May 2007), pp. 275–282. 1

[BPS97] BAJAJ C. L., PASCUCCI V., SCHIKORE D.: The contour spectrum. In *IEEE Visualization'97* (1997), pp. 167–174. 1

[DK95] DIETTERICH T., KONG E.: Error-correcting output codes corrects bias and variance. In *S. Prieditis and S. Russell* (1995), of the 21th International Conference on Machine Learning P., (Ed.), pp. 313–321. 2

[EPR10] ESCALERA S., PUJOL O., RADEVA P.: On the decoding process in ternary error-correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32* (2010), 120–134. 3, 4

[ETP*08] ESCALERA S., TAX D., PUJOL O., RADEVA P., DUIN R.: Subclass problem-dependent design of error-correcting output codes. In *IEEE Transactions in Pattern Analysis and Machine Intelligence* (2008), vol. 30, pp. 1–14. 3

[FHT98] FRIEDMAN J., HASTIE T., TIBSHIRANI R.: Additive logistic regression: a statistical view of boosting. *Annals of Statistics 28* (1998). 3

[FPT06] FERRÉ M., PUIG A., TOST D.: Decision trees for accelerating unimodal, hybrid and multimodal rendering models. *The Visual Computer 3* (2006), 158–167. 2

[FWG09] FUCHS R., WASER J., GRÖLLER M. E.: Visual human-machine learning. *IEEE TVCG 15*, 6 (Oct. 2009), 1327–1334. 2

[GCD] GCD: Grand central dispatch (gcd). 6

[GDB08] GARCIA V., DEBREUVE E., BARLAUD M.: Fast k nearest neighbor search using gpu. 2

[GMK*92] GERIG G., MARTIN J., KIKINIS R., KUBLER O., SHENTON M., JOLESZ F.: Unsupervised tissue type segmentation of 3-d dual-echo mr head data. *Image and Vision Computing 10*, 6 (1992), 349–36. 2

[HWS10] HERRERO S., WILLIAMS J., SANCHEZ A.: Parallel multiclass classification using svms on gpus. In *ACM International Conference Proceeding Series, Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units* (2010), vol. 425, pp. 2–11. 2

[JDM00] JAIN A. K., DUIN R. P., MAO J.: Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22* (2000), 4–37. 2

[KBKG08] KOHLMANN P., BRUCKNER S., KANITSAR A., GRÖLLER M.: *The LiveSync Interaction Metaphor for Smart User-Intended Visualization*. Tech. Rep. TR-186-2-08-01, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Jan. 2008. 1

[KD98] KINDLMANN G., DURKIN J.: Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium on Volume Visualization* (October 1998), IEEE Press, pp. 79–86. 1

[KKH01] KNISS J., KINDLMANN G., HANSEN C.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proc. of the conference on Visualization 2001* (2001), IEEE Press., pp. 255–262. 1

[KSW06] KRÜGER J., SCHNEIDER J., WESTERMANN R.: Clearview: An interactive context preserving hotspot visualization technique. *IEEE Trans. on Visualization and Computer Graphics (Proc. Visualization / Information Visualization 2006) 12*, 5 (sep- oct 2006). 1

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P. A., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design galleries: a general approach to setting parameters for computer graphics and animation. *Computer Graphics 31*, Annual Conference Series (1997), 389–400. 1

[Mic09] MICIKEVICIUS P.: 3d finite difference computation on gpus using cuda. In *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units* (2009), pp. 79–84. 6

[mov] http://movibio.lsi.upc.edu/website. 6

[PLB*01] PFISTER H., LORENSEN B., BAJA C., KINDLMANN G., SHROEDER W., AVILA L., RAGHU K., MACHIRAJU R., LEE J.: The transfer function bake-off. *IEEE Computer Graphics & Applications 21*, 3 (2001), 16–22. 1

[TLM03] TZENG F. Y., LUM E., MA K. L.: A novel interface for higher dimensional classification of volume data. In *Visualization 2003* (2003), IEEE Computer Society Press, pp. 16–23. 2

[TM04] TZENG F.-Y., MA K.-L.: A cluster-space visual interface for arbitrary dimensional classification of volume data. In *Proceedings of the Joint Eurographics-IEEE TVCG Symposium on Visualization 2004* (May 2004). 1, 2

[VKG05] VIOLA I., KANITSAR A., GRÖLLER M. E.: Importance-driven feature enhancement in volume visualization. *IEEE Trans. on Visualization and Computer Graphics 11*, 4 (2005), 408–418. 1

[YSMR10] YUDANOV D., SHAABAN M., MELTON R., REZNIK L.: Gpu-based simulation of spiking neural networks with real-time performance and high accuracy. In *WCCI-2010, Special Session Computational Intelligence on Consumer Games and Graphics Hardware CIGPU-2010* (2010). 2