

# Intelligent Interactive Volume Classification

S. Grau<sup>1</sup>, A. Puig<sup>2</sup>, S. Escalera<sup>2,3</sup> and M. Salomó<sup>2</sup>

<sup>1</sup>Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Spain

<sup>2</sup>Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Spain

<sup>3</sup>Centre de Visió per Computador, Universitat Autònoma de Barcelona, Spain

---

## Abstract

*This paper defines an intelligent and interactive framework to classify multiple regions of interest from the original data on demand, without requiring any preprocessing or previous segmentation. The proposed intelligent and interactive approach is divided in three stages: visualize, training and testing. First, users visualize and label some samples directly on slices of the volume. Training and testing are based on a framework of Error Correcting Output Codes and Adaboost classifiers that learn to classify each region the user has painted. Later, at the testing stage, each classifier is directly applied on the rest of samples and combined to perform multi-class labeling, being used in the final rendering. We also parallelized the training stage using a GPU-based implementation for obtaining a rapid interaction and classification.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Object hierarchies

---

## 1. Introduction

Gathering visual information often requires a high-level of expertise from the final user, who has to mentally work out with a large amount of data, to segment the input models. It is important to alleviate this mental process by providing an automatic and interactive method that allows the user to classify different structures on demand in an intuitive way.

In volume rendering literature many papers address classification by editing transfer functions (TF) [PLB\*01] or Multidimensional TFs [TM04, TLM05]. However, the design complexity and the memory requirements increase with the TFs dimensionality. Moreover, to define the TFs manually or with some friendly interfaces [KKH01], even by skilled users, becomes a hard task. Nevertheless, to recognize semantic structures requires more sophisticated techniques, such as the user-guided volume segmentation [PRH10], where users refine and correct iteratively the results of the probabilistic random walker approach. Previously, in [EPAS11], we proposed an approach that uses AdaBoost and Error Correcting Output Codes (ECOC) framework for labeling a subset of volume classes based on a pre-defined learning step. In this paper we continue this research by defining an iterative process, as the one introduced by [PRH10], divided into three stages for assisting the user in classifying multiple regions of interest on demand from

the original data, without requiring any preprocessing or previous segmentation. In particular, we provide the following contributions addressing the challenge of classifying on demand different structures of interest within an intuitive interaction process: (1) We present the whole iterative framework divided into three main stages; (2) We design an intuitive interface that allows the user to improve iteratively the final classification; (3) We propose a new parallelization of the ECOC and AdaBoost classifiers within the training stage, which is integrated in the GPU volume rendering process.

## 2. Related Work

Volume labeling corresponds to the classification at pixel level, also named segmentation. The special case of segmentation we treat is the learning of different visual regions which may have different visual attributes. In this sense, we have to distinguish from classical unsupervised methods for segmentation based on clustering. In the supervised segmentation process, it is common to model pixel/voxel attributes from user interactions as a prior knowledge to optimize posterior segmentation/classification. In this field, GrabCut approach [RKB04] has become a standard technique to model regional properties and optimize final segmentation in images using Graph Cuts theory [BK04] and an example of application is provided in [HVPE11]. This framework has been recently extended to segment volumes [PYA\*13]. How-

ever, these works involve segmentation dependencies among neighboring pixel/voxels and complex optimization procedures, which make them no suitable for real-time applications.

Probabilistic segmentation of volume models using GPU implementations have been recently proposed [SMH10, PRH10]. However, thinned and complex structures are difficult to be classified with these methods, and their extension to segment multiple structures is not straightforward. Lately, some works have been published based on data-driven and image-driven classification that provide users with higher-level information about data distribution and about the final visualization, respectively. Supervised methods such as neuronal networks [TLM03], decision trees [FPT06] and non-supervised methods [TM04] have been applied to different user interfaces of volume analysis applications.

Several studies have addressed the classification task in image processing with different machine learning approaches. Most of the studies described so far have been limited to binary classifications using GPGPU. Clustering strategies and computation of  $k$ -nearest neighbour similarity classifiers are presented in [GDB08], Geometrical Support Vector Machine classifiers [YGJ\*10, HWS10], or Neural Networks [YSMR10]. Recently we propose the use of AdaBoost classifier [EPR10], which is a strong binary classifier with a high generalization capability. Additionally, in order to deal with multi-class labeling, we combine AdaBoost in an ECOC framework [EPAS11].

### 3. Interactive Classification Framework

We define an interactive and intuitive framework of supervised statistical methods to classify multiple regions of interest from the original data on demand, without requiring any preprocessing or previous segmentation. The proposed system is divided into three main stages shown in Fig. 1. First of all, for each region to be learnt, users visualize the input data and intuitively label some samples directly on slices of the volume. Secondly, at the training stage, an ECOC design trains the input samples using a set of Discrete AdaBoost binary classifiers. Finally, at the testing stage, each classifier is independently applied on the rest of unlabeled samples and combined within the ECOC design to perform multi-class labeling in the final rendering. This classification process iteratively improves as users label/unlabel samples, allowing to refine training and testing stages. The advantages of this approach are two-fold. First, it is quite general for being used efficiently to a great variety of data sets. Second, the mental process that the user should perform to convey relevant information is alleviated by the simple nature of the iterative framework. The user only needs to paint some input samples and the framework automatically classifies different structures on demand. Once an iteration is performed, the user may decide to improve the classification by proposing new input samples -and a new iteration starts- or may finish

the process when he considers that the classification is good enough.

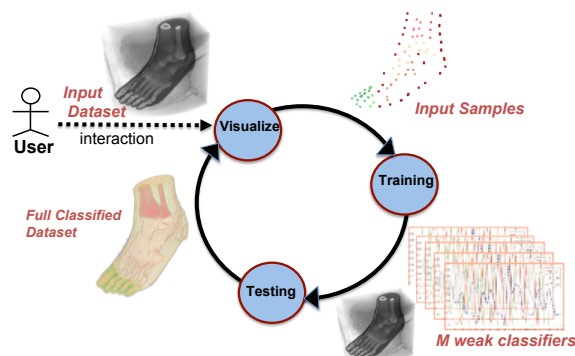


Figure 1: Overview of the proposed framework.

#### 3.1. Visualize Stage: Interactive Sample Definition

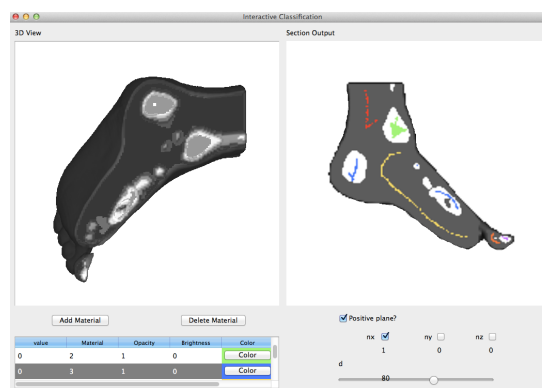


Figure 2: Interface for interactive sample definition.

Figure 2 shows a snapshot of the interface. First of all, the user opens a volume data set which is shown in 3D at the top left side of the interface, see Fig. 2. In that side, the user may rotate the volume as he desires. In the bottom of the interface, multiple UI controls allow users to fine-tune their preferences, will be displayed on the top right side in a 2D volume visualization. This top right side shows the user-preferred inner plane of the volume. Moreover, users are able to define new labels by choosing its name. To emphasize the volume parts, distinct colors for each label are used as visual cues, as shown in Fig. 2 in blue and green colors. As the user moves over the 2D volume, he paints or removes voxels that are associated to the selected label. The user finishes the definition of the samples by selecting the training in the menu too, being able to define which labels he prefers to see in the 3D volume. The result of this stage is shown in the top left side of the interface. Once the user visualizes the result, he may continue improving the results by painting/deleting samples on the top right side.

### 3.2. AdaBoost Classifier

We train the ECOC classifiers using Discrete AdaBoost classifier [FHT98, ETP\*08]. Let  $\mathcal{T}$  be the set of  $k$  training samples,  $\rho$ , we define  $F(\rho) = \sum_1^{\mathcal{M}} c_f f_m(\rho)$  where each  $f_m(\rho)$  is a classifier producing values  $\pm 1$  and  $c_f$  are constants; the corresponding prediction is  $\text{sign}(F(\rho))$ . The AdaBoost procedure trains the classifiers  $f_m(\rho)$  on weighed versions of the training sample, and then the final classifier is defined to be a linear combination of the classifiers from each stage. The training stage is shown in Algorithm 1.  $E_w$  represents expectation over the training data with weights  $w = (w_1, w_2, \dots, w_k)$ , and  $1_S$  is the indicator of the set  $S$ . Finally, Algorithm 2 shows the testing of the final decision function  $F(\rho) = \sum_1^{\mathcal{M}} c_f f_m(\rho)$  using Decision Stump "weak classifier". Each Decision Stump  $f_m$  fits a threshold  $T_m$  and a polarity  $P_m$  over the selected  $m$ -th feature. In testing,  $\rho^m$  corresponds to the value of the feature selected by  $f_m(\rho)$  on a test sample  $\rho$ . Finally decision on  $\rho$  is obtained by  $\text{sign}(F(\rho))$ .

---

#### Algorithm 1: Discrete AdaBoost Training ( $\mathcal{T}, \theta$ ).

---

- 1: Start with weights  $w_i = 1/k, i = 1, \dots, k$ .
  - 2: **for**  $m = 1, 2, \dots, \mathcal{M}$  **do**
  - 3:   Fit the classifier  $f_m(\rho) \in -1, 1$  using weights  $w_i$  on the training data.
  - 4:   Compute  $\text{err}_m = E_w[1_{(l(\rho) \neq f_m(\rho))}], c_m = \log((1 - \text{err}_m)/\text{err}_m)$ .
  - 5:   Set  $w_i \leftarrow w_i \exp[c_m \cdot 1_{(l(\rho_i) \neq f_m(\rho_i))}], i = 1, 2, \dots, k$ , and normalise so that  $\sum_i w_i = 1$ .
  - 6: **end for**
  - 7: Output the classifier  $\text{sign}[\sum_{m=1}^{\mathcal{M}} c_m f_m(\rho)]$ .
- 

---

#### Algorithm 2: Discrete AdaBoost Testing ( $\rho, h_i$ ).

---

- 1: Given a test sample  $\rho$
  - 2:  $F(\rho) = 0$
  - 3: **for**  $m = 1, 2, \dots, \mathcal{M}$  **do**
  - 4:    $F(\rho) = F(\rho) + c_m (P_m \cdot \rho^m < P_m \cdot T_m)$ ;
  - 5: **end for**
  - 6: Output  $\text{sign}(F(\rho))$
- 

### 4. Parallelization

For the training and testing stages we have considered the general Error-Correcting Output Codes framework to deal with multi-class classification, and as a case study, we used AdaBoost to train the sets of binary classifiers, for details about the algorithms see [EPAS11] where the testing stage was parallelized but not the training as it was pre-defined.

In this paper we have proposed an interactive way to paint samples and train the classifiers. Additionally, we propose to parallelize the training stage using GPU for obtaining a rapid interaction and real-time classification. For optimizing the training stage, we use an equivalent LUT-based representation of AdaBoost classifier.

### 4.1. AdaBoost Parallelization

We define the matrix  $\mathcal{L} = \{P, T, c\}$  of size  $3 \times (|\rho| \times W_c)$ , where  $P = \{P_1, \dots, P_m\}$  defines the polarity of the  $m$  features,  $T = \{T_1, \dots, T_m\}$  is a threshold for the  $m$  features, and  $c = \{c_1, \dots, c_m\}$  are the class problems. The  $|\rho|$  corresponds to the dimensionality of the feature space and  $W_c$  to the number of weak classifiers used in AdaBoost training step that correspond in a one-versus-one coding design to  $\frac{N(N-1)}{2}$  where  $N$  are the different possible labels in a volume. Therefore, we have three rows of  $(|\rho| \times W_c)$  size. First row of  $\mathcal{L}$  codifies the weight values of weak classifiers. In this way, each position  $i$  of the first row of  $\mathcal{L}$  contains the weight value for the feature computed using the modulus as  $\text{mod}(i, |\rho|)$ . The next weight value for that feature is found at position  $i + |\rho|$ . The positions corresponding to features not considered during training are set to zero. The second and third rows of  $\mathcal{L}$  for column  $i$  contains the values of polarity  $P_m$  and threshold  $T_m$  used in a Decision Stump weak classifier during training.

In our proposal, each sample corresponds to a voxel. Let  $VM$  be a voxel model  $dim_x \times dim_y \times dim_z$  sized. For all  $v_{xyz} \in VM$ , we defined the feature vector  $\rho$  as:  $x$ ,  $y$ , and  $z$  coordinates, the respective gradients  $g_x$ ,  $g_y$ , and  $g_z$ , the gradient magnitude  $|g|$ , and the density value  $d$ . The  $\mathcal{T}$  training data set is the  $k$  training samples introduced by the proposed user interface (see section 3). Each training sample,  $(\rho, l(\rho))$  corresponds to a feature vector  $\rho$  plus its associated user-defined label  $l(\rho)$  and it is defined as  $(\rho, l(\rho)) = [x, y, z, d, g_x, g_y, g_z, |g|, l(\rho)]$ . Then, in the first row of the proposed AdaBoost Look Up Table  $\mathcal{L}$ , we codify eight successive values for  $\rho^s, s \in [1, \dots, 8]$  corresponding to the eight sample features. For example,  $\rho^1$  corresponds to the  $x$  coordinate. Once trained each of the AdaBoost weak classifiers (Decision Stumps in our case), in the second and third rows of  $\mathcal{L}$  we codify the values of polarity and threshold for their corresponding feature positions, respectively. Therefore, the Discrete Testing Algorithm (proposed in [EPAS11]) can be reinterpreted using voxels as the test data set  $\mathcal{T}_e$  and codifying the dichotomizers  $h_i$  with the  $\mathcal{L}$  Look Up Table. 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 (dichotomizers) over the partitions are trained (using AdaBoost in our case).

### 4.2. Training Parallelization

The parallelization of training stage of the ECOC framework can be classified as a parallel irregular program [KBCEP09], and some parallelization patterns could be applied to optimize the Discrete AdaBoost Training of one dichotomizer  $h_i$ . However, the computation of the weak classifiers  $w_i$ , involves dependencies through the successive iterations of the main loop of the training algorithm that exhibit read and write memory conflicts that cannot be avoided easily.

At each iteration a new  $w_i$  is calculated in function of the previously computed *weak classifiers*.

The total cost of the sequential training of one dichotomizer  $h_i$  is  $O(|\mathcal{T}| \times \mathcal{M})$ , where  $|\mathcal{T}|$  is the cardinality of the training data set  $\mathcal{T}$ . Taking into account that  $|\mathcal{T}|$  is a small value, i.e., is the number of training samples introduced by the user at each cycle of the process, we let this process in its serial version. Nevertheless, we can easily compute in parallel the  $n$  dichotomizers associated to the  $n$  binary independent problems. Then, each thread computes the corresponding  $\mathcal{L}_i = \{P_i, T_i, c_i\}$  from the same  $\mathcal{T}$  training data set. This operation is data independent among dichotomizers, as each thread concurrently calculates the AdaBoost Look Up Table without write-memory access conflicts. This parallelization is specially suitable when the user deals with a high number of classes, but in the worst case, the upper bound becomes the serial cost. Thus, the complete process has a total cost of  $O(|\mathcal{T}| \times \mathcal{M})$ .

## 5. Simulations and Results

This section describes the experimental setup and shows the performance evaluation of the proposed framework.

### 5.1. Setup

**Data:** We used three data sets, *Thorax* data set of size  $400^3$  represents a MRI phantom human body; *Foot* and *Brain* data sets of sizes  $128^3$  and  $256 \times 256 \times 159$  are CT scans of a human foot and a human brain, respectively.

**Methods:** We use the one-versus-one ECOC design with Discrete AdaBoost with 30 decision stumps as the base classifier. For each voxel sample  $\rho$ , 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 implemented in GPU.

**Measurements:** We measure the number of samples selected by the user and the accuracy obtained by the testing stage at each iteration. The accuracy  $acc \in [0, \dots, 1]$  is estimated as the number of correctly classified voxels in relation to the number of voxels of the data set, removing background voxels based on value. As a proof of concept and without loss of generality three independent users label, for each data set, a set of voxels that belongs to a predefined set of structures using five iterations. Each data set is the input data of the training stage. The number of voxels considered at iteration  $t + 1$  also include the selected at iteration  $t$ . As a reference measure we estimate the performance when the training used all the volume samples, named as Reference in the tables. Moreover, to measure the maximum accuracy of our classifier, we use data sets that have been previously segmented by an expert (Ground Truth). Tables show the average of the three users.

### 5.2. Analysis

We implemented the system in the GPU, using OpenCL in both training and testing stages. In general, the number of selected voxels is between 1880 in the first iteration to 14200 at the last one. Time performance for training is interactive, being 5 seconds in the worst case. On the other hand, testing runs in real time for all datasets.

Tables 1, 2 and 3 depict the results for the *Brain*, *Foot*








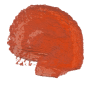

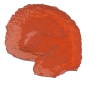

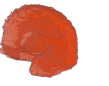

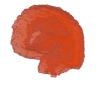






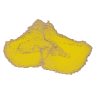







and *Thorax* data sets, respectively. For each data set we have defined different structures (labels) denoted as L1, L2, and so on. Specifically, they are in the *Brain* (the right hemisphere, cerebellum, and left hemisphere), in the *Foot* (ankle-muscle, palm-muscle, toes-muscle, ankle-bone, palm-bone, and toes-bone), and in the *Thorax* (column, bones, heart, and soft tissue, respectively). Some of them are not shown due to space limitations. First row of each table show the renderings obtained for the whole volume (ALL) and for each one of the structures along the five iterations. Each column represents one iteration. Additionally, in the last two columns, we introduced the result of the Reference and the Ground Truth data sets. Note that below each color plate we show the corresponding accuracy  $acc$ .

One can see at Tables 1, 2 and 3 that the percentage of correctly classified samples increases with the number of user iterations, achieving almost the same accuracy as the reference value at the fifth iteration. As it has been seen, even for different complexity of volume structures, most of the categories yield high classification rates with few user iterations. In all data sets the achieved mean accuracy at fifth iteration is upon 90%. In the case of the *Foot* data set, accuracy achieves above 99% from the second to the last iteration. Moreover, our proposal is able to accurately segment thinned and complex structures, such as toes' bones.

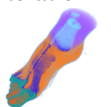
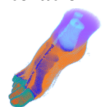
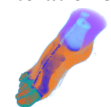
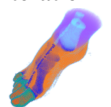

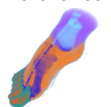
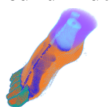
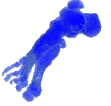
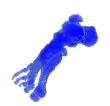
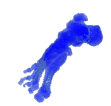
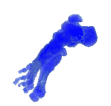
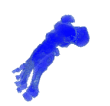

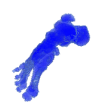














Our analysis states that the proposed iterative framework requires few voxels to obtain high accuracies, achieving close or equal results to those obtained by learning a whole labeled volume, see Figure 3(a) and 3(b). As shown in Fig. 3(c), the users almost achieved the same performance than training the whole data just selecting between 0,1% in the *Thorax* data set to less than 4% of the voxels of the *Foot* volume in the last iteration. This augment is largely produced by the increment of the labels to be learnt, although the difficulty of learning some of the structures (e.g., L6) may affect it moderately. The most difficult label to be learnt in the *Foot* is L6, which corresponds to the toes-bone structure. In particular, this structure only uses a 14% of the voxels of the toes for training. Note that at the first iteration it lacks one of the fingers whereas at the last one the image delineates almost a perfect segmentation in relation to the ground truth data.

## 6. Conclusions

We proposed an interactive and iterative process for assisting the user in classifying multiple regions of interest on demand. An intuitive interface has been designed to help the user in the process of defining relevant volume structures. From the set of user selected voxels a training stage is performed based on voxel characteristics. The system uses the Error-Correcting Output Codes framework and AdaBoost base classifier to learn the selected properties and classify the data set. To obtain a rapid interaction and classification, we implemented the training and testing stages using the GPU. The empirical results on different data sets show high classi-

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Reference	Ground Truth
ALL							
acc	0,868	0,908	0,952	0,959	0,960	0,968	
L1							
acc	0,899	0,913	0,953	0,958	0,958	0,975	
L2							
acc	0,878	0,926	0,950	0,954	0,954	0,958	
L3							
acc	0,834	0,898	0,951	0,962	0,964	0,964	

**Table 1:** Visualization of the Brain data set along user iterations for different volume structures.

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Reference	Ground Truth
ALL							
acc	0,983	0,990	0,992	0,992	0,989	1,000	
L4							
acc	0,999	0,999	0,999	0,999	0,998	1,000	
L5							
acc	0,887	0,887	0,948	0,948	0,949	1,000	
L6							
acc	0,781	0,781	0,854	0,855	0,967	1,000	

**Table 2:** Visualization of the Foot data set along user iterations for different volume structures.

fication accuracy and fast computation of the system, being a reliable tool for semi-automatic volume segmentation.

## ACKNOWLEDGMENTS

Work partially funded by TIN2011-24220 and MICINN Grant TIN2009-14404-C02 Spanish research projects.

## References

- [BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI* 26, 9 (2004), 1124–1137. 1
- [EPAS11] ESCALERA S., PUIG A., AMOROS O., SALAMÓ M.: Intelligent gpgpu classification in volume visualization: A framework based on error-correcting output codes. *Computer Graphics Forum* 30, 7 (2011), 2107–2115. 1, 2, 3
- [EPR10] ESCALERA S., PUJOL O., RADEVA P.: On the decoding process in ternary error-correcting output codes. *PAMI* 32 (2010), 120–134. 2
- [ETP\*08] ESCALERA S., TAX D., PUJOL O., RADEVA P., DUIN R.: Subclass problem-dependent design of error-correcting output codes. In *PAMI* (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
- [GDB08] GARCIA V., DEBREUVE E., BARLAUD M.: Fast k nearest neighbor search using gpu. In *In Proceedings of the CVPR Workshop on Computer Vision on GPU* (2008). 2
- [HVPE11] HERNÁNDEZ-VELA A., PRIMO C., ESCALERA S.: Automatic user interaction correction via multi-label graph cuts. *HICV, ICCV* (2011). 1



	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Reference	Ground Truth
ALL							
acc	0,879	0,896	0,906	0,922	0,922	0,922	
L1							
acc	0,944	0,950	0,921	0,917	0,934	0,985	
L3							
acc	0,890	0,867	0,866	0,854	0,844	0,940	
L4							
acc	0,890	0,911	0,908	0,931	0,929	0,929	

Table 3: Visualization of the Thorax data set along user iterations for different volume structures.

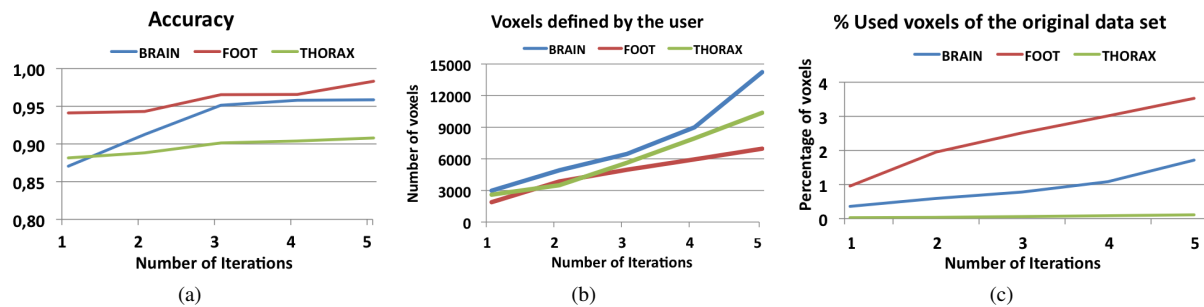


Figure 3: (a) Mean accuracy results of volume data sets based on user interactions. (b)-(c) Mean absolute and relative number of voxels selected by the user at each iteration in relation to the size of each data set.

[HWS10] HERRERO S., WILLIAMS J., SANCHEZ A.: Parallel multiclass classification using svms on gpus. In *ACM General-Purpose Computation on Graphics Processing Units* (2010), vol. 425, pp. 2–11. 2

[KBCP09] KULKARNI M., BURTSCHER M., CASCAVAL C., PINGALI K.: Lonestar: A suite of parallel irregular programs. In *ISPASS* (2009), pp. 65–76. 3

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

[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

[PRH10] PRASSNI J., ROPINSKI T., HINRICHS K.: Uncertainty-aware guided volume segmentation. *Visualization and Computer Graphics* 16, 6 (2010), 1358–1365. 1, 2

[PYA\*13] PUNITHAKUMAR K., YUAN J., AYED I. B., LI S., BOYKOV Y.: A convex max-flow approach to distribution based figure-ground separation. *SIAM Imag. Sciences* (2013). 1

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.* 23, 3 (2004), 309–314. 1

[SMH10] SAAD A., MÖLLER T., HAMARNEH G.: Probexplorer: uncertainty-guided exploration and editing of probabilistic medical image segmentation. In *Eurographics-VGTC Visualization* (2010), pp. 1113–1122. 2

[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

[TLM05] TZENG F., LUM E., MA K.: An intelligent system approach to higher-dimensional classification of volume data. *Visualization and Computer Graphics 11* (2005), 273–284. 1

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

[YGJ\*10] YANG D., GETAO L., JENKINS D., PETERSON G., LI H.: High performance relevance vector machine on gpus. In *App. Accelerators in High Perf. Computing* (2010). 2

[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). 2