OpenCL based machine learning labeling of biomedical datasets

Oscar Amoros, Sergio Escalera, and Anna Puig
University of Barcelona, Avda. Corts Catalanes, 585, 08007, Barcelona, Spain

ABSTRACT

In this paper, we propose a two-stage labeling method of large biomedical datasets through a parallel approach in a single GPU. Diagnostic methods, structures volume measurements, and visualization systems are of major importance for surgery planning, intra-operative imaging and image-guided surgery. In all cases, to provide an automatic and interactive method to label or to tag different structures contained into input data becomes imperative. Several approaches to label or segment biomedical datasets has been proposed to discriminate different anatomical structures in an output tagged dataset. Among existing methods, supervised learning methods for segmentation have been devised to easily analyze biomedical datasets by a non-expert user. However, they still have some problems concerning practical application, such as slow learning and testing speeds. In addition, recent technological developments have led to widespread availability of multi-core CPUs and GPUs, as well as new software languages, such as NVIDIA's CUDA and OpenCL, allowing to apply parallel programming paradigms in conventional personal computers.

Adaboost classifier is one of the most widely applied methods for labeling in the Machine Learning community. In a first stage, Adaboost trains a binary classifier from a set of pre-labeled samples described by a set of features. This binary classifier is defined as a weighted combination of weak classifiers. Each weak classifier is a simple decision function estimated on a single feature value. Then, at the testing stage, each weak classifier is independently applied on the features of a set of unlabeled samples.

In this work, we propose an alternative representation of the Adaboost binary classifier. We use this proposed representation to define a new GPU-based parallelized Adaboost testing stage using OpenCL. We provide numerical experiments based on large available data sets and we compare our results to CPU-based strategies in terms of time and labeling speeds.

Keywords: Image computing architecture, Image-guided procedures

1. INTRODUCTION

There are several biomedical applications that need to extract different objects of interest, such as tissues and organs contained into a volume dataset from MRI, CT, fMRI, and PET input captions. Diagnostic methods, structures volume measurements, and visualization systems require to specify to which anatomical structure each sample/voxel belongs. In the bibliography, Transfer Functions has been used in order to directly associate optical properties or labels to the different data samples according their belonging to a particular structure in the underlying data. However, in general, the anatomical structures are complex, and relationships between them do not allow to separate sufficiently the different structures. In these cases, Transfer Functions alone do not suffice in order to separate different objects and it becomes necessary to use labeling or segmentation methods. In this sense, several approaches to label biomedical datasets have been proposed to discriminate different anatomical structures in an output tagged dataset depending on the used imaging modality. Among existing methods, supervised learning methods for segmentation have been devised to easily analyze biomedical datasets by a non-expert user. In a preprocess, an expert user, such a radiologist, should identify a subset of samples of each anatomical structure. Then, during the learning step, the supervised method defines a classifier to be automatically used in the testing or classification stage. Since the learning phase is carried on a pre-process

Further author information: (Send correspondence to O.A.)
O.A.: E-mail: oamorohu7@alumnes.ub.edu

(before labeling) the same classifier can be used by inexpert users in different classifications several times. Thus, to optimize the classification stage becomes imperative.

Thanks to the recent emerging technologies of multi-core CPUs and GPUs, as well as new software languages, such as NVIDIA's CUDA, and OpenCL, we propose to parallelize the classification step of a well-know supervised method, the Adaboost classifier, in the GPU.

2. PREVIOUS WORK

Many papers in volume rendering literature address classification.[1] Most of them are based on the edition of transfer functions (TF),[2][3],[4].[5] One-dimensional TFs only take into account scalar voxel values. Multi-dimensional TFs take into account vectorial values or combinations of local measure of scalar values (derivatives,[5] position,[6] or statistical signatures[7]). However, the design complexity and the memory requirements increase with the TFs dimensionality.

Recently, some preliminary works based on learning methods have been published based on data-driven and on image-driven classification, that provide users with higher-level information about data distribution and about the final visualization, respectively. Supervised methods such as bayesian networks,[8],[9] neuronal networks,[10] decision trees[11] and non-supervised methods,[6][12] have been applied in different user interfaces of volume applications. In,[13] clustering-based supervised and non-supervised learning methods are compared for classifying magnetic resonance data.

Additionally, different Machine Learning approaches have been recently implemented using GPGPU, such as clustering strategies and fast computation of k-nearest neighbor similarity classifier in.[14] In,[15–17] different versions of SVM classifier are implemented for GPU. Neural Networks have also been intensively applied for machine learning applications in GPU.[18] Finally, Adaboost classifier has become very popular in last years. Based on a *weak classifier*, Adaboost defines an additive model combining simple decisions in order to define a strong classifier with high generalization capability. In,[19] authors use Adaboost to ensemble weak segmentation hypotheses and majority voting to liver lesion extraction from CT volume. It shows the suitability of Adaboost for segmenting and classifying CT volumes, though the implementation out of GPU and a mathematical morphology post-filtering increase the computational complexity of the method. Thus, the inherent parallel structure of Adaboost, its high performance, and its simplicity in order to train (Adaboost does not require tuning classifier parameters) open a research line to exploit its high potential for GPU applications.

3. ADABOOST BINARY CLASSIFIER

Adaboost classifier is one of the most widely applied methods for labeling in the Machine Learning community. In this paper, we focus on the Discrete version of Adaboost, which has shown robust results in real applications. Given a set of N training samples $(x_1, y_1), .., (x_N, y_N)$, with x_i a vector valued feature and $y_i = -1$ or 1. We define $F(x) = \sum_1^M c_f f_m(x)$ where each $f_m(x)$ is a classifier producing values ± 1 and c_m are constants; the corresponding prediction is $\text{sign}(F(x))$. The Adaboost procedure trains the classifiers $f_m(x)$ on weighed versions of the training sample, giving higher weights to cases that are currently misclassified. This is done for a sequence of weighted samples, and then the final classifier is defined to be a linear combination of the classifiers from each stage.

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

Algorithm 1: Discrete Adaboost training algorithm.

The binary Discrete training Adaboost algorithm is shown in Algorithm 1. E_w represents expectation over the training data with weights $w = (w_1, w_2, .., w_N)$, and $1_{(S)}$ is the indicator of the set S. For a good generalization of $F(x)$, each $f_m(x)$ is required to obtain a classification prediction just better than random. Thus, the most common "weak classifier" f_m is the "decision stump". For each $f_m(x)$ we just need to compute a threshold value and a polarity to take a binary decision, selecting that one that minimizes the error based on the assigned weights. This simple combination of classifiers has demonstrated to reduce the variance error term of the final classifier $F(x)$.

In Algorithm 2, we show the *testing* of the final decision function $F(x) = \sum_1^M c_f f_m(x)$ using the Discrete Adaboost algorithm with 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 time, x^m corresponds to the value of the feature selected by $f_m(x)$ on a test sample x. Note that c_m value is subtracted from $F(x)$ if the hypothesis $f_m(x)$ is not satisfied on the test sample. Otherwise, positive values of c_m are accumulated. Finally decision on x is obtained by sign$(F(x))$.

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

Algorithm 2: Discrete Adaboost testing algorithm.

We propose to define a new and equivalent representation of c_m and $|x|$ that facilitate the paralellization of the testing. We define the matrix $V_{f_m(x)}$ of size $3 \times (|x| \cdot M)$, where $|x|$ corresponds to the dimensionality of the feature space. First row of $V_{f_m(x)}$ 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 $V_{f_m(x)}$ contains the value c_m for the feature $mod(i, |x|)$ if $mod(i, |x|) \neq 0$ or $|x|$, otherwise. The next value of c_m for that feature is found in position $i + |x|$. The positions corresponding to features not considered during training are set to zero. The second and third rows of $V_{f_m(x)}$ for column i contains the values of P_m and T_m for the corresponding Decision Stump.

Note that in the representation of $V_{f_m(x)}$ we loss 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 codified, and thus, the final additive decision model $F(x)$ is equivalent.

4. OPENCL IMPLEMENTATION

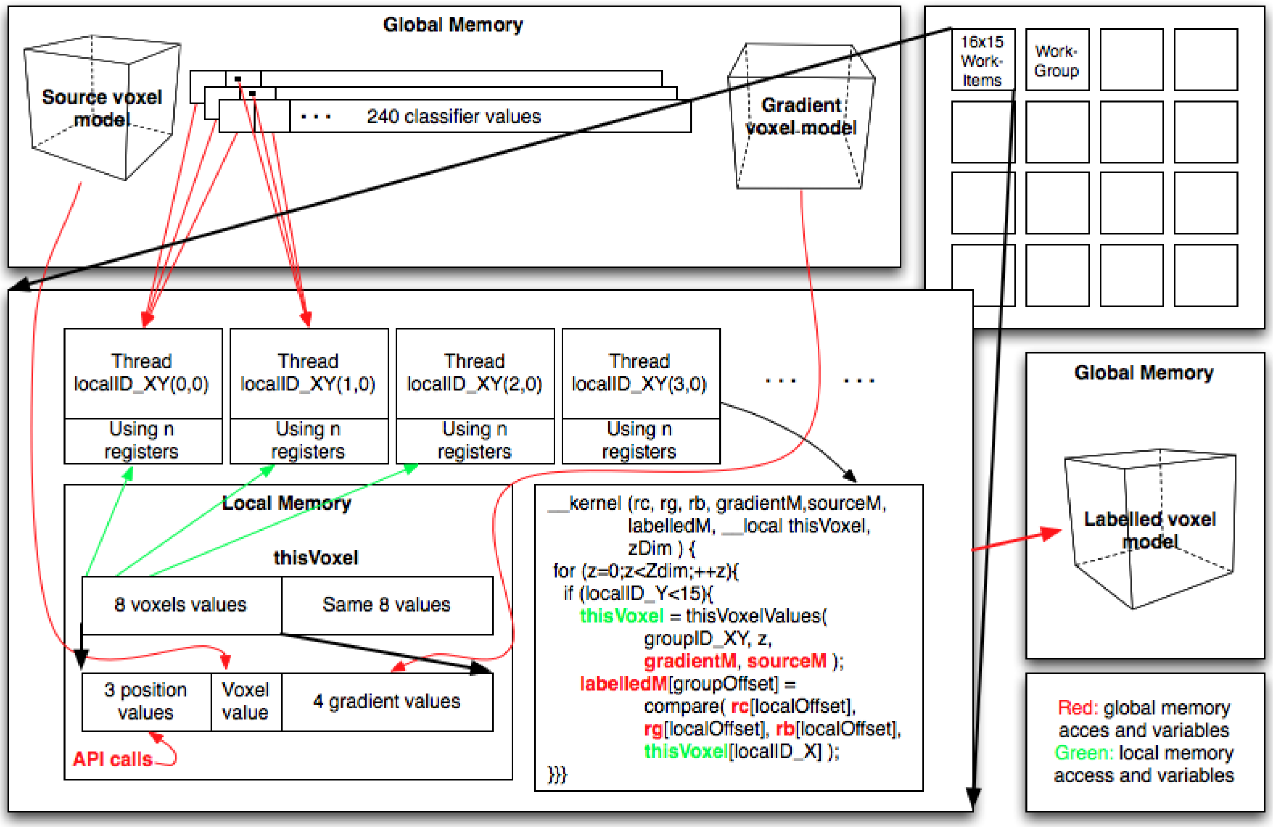
Our proposed OpenCL's implementation is based primarily on the GPU-PCIe accelerator concept, taking into account the PCIe bottleneck and the GPU's main memory or Global Memory. As we deal with large datasets of voxels, we use out-of-core techniques to subdivide the initial dataset into a subset which can fit into GPU main memory. Thus, PCIe bandwidth becomes an essential factor in the overall execution time.

We overview our proposed GPU-OpenCL algorithm in Figure 1. The eight features considered at each sample by our binary classifier are: the spatial location (x, y, z), the sampled value (v), and its associated gradient value (gx, gy, gz, $|g|$). Our binary classifier, for each feature, we have a total of N possible C_m values, with $N = 3 \cdot M$.

We create a matrix of Work-Groups that covers the x and y size of the dataset fitted into GPU global memory, whereas the component z is computed in a inner loop at each kernel. Each WorkGroup classifies one voxel. Inside each Workgroup, we define $N \cdot 8$ threads (or WorkItems). Each thread computes a single operation with the 3 channels or weights of the weak classifier. These $N \cdot 8$ values will be reduced at the end of the execution and compared to a reduced addition. The final label at each voxel is directly computed by this comparison.

5. RESULTS

In order to present the results, we considerate different three-dimensional data from CT and MRI image modalities with different sizes. For each data set we trained a Discrete Adaboost classifier with 32 Decision Stumps.


Global Memory
Source voxel model
... 240 classifier values
Gradient voxel model
16x15 Work-Items
Work-Group
Thread localID_XY(0,0) Using n registers
Thread localID_XY(1,0) Using n registers
Thread localID_XY(2,0) Using n registers
Thread localID_XY(3,0) Using n registers
... ...
Local Memory
thisVoxel
8 voxels values
Same 8 values
3 position values
Voxel value
4 gradient values
API calls
Global Memory
Labelled voxel model
Red: global memory acces and variables
Green: local memory access and variables

```
__kernel (rc, rg, rb, gradientM,sourceM,
          labelledM, __local thisVoxel,
          zDim ) {
for (z=0;z<Zdim;++z){
  if (localID_Y<15){
    thisVoxel = thisVoxelValues(
              groupID_XY, z,
              gradientM, sourceM );
    labelledM[groupOffset] =
              compare( rc[localOffset],
              rg[localOffset], rb[localOffset],
              thisVoxel[localID_X] );
}}}
```

Figure 1. The proposed OpenCL implementation.

For each voxel we considered eight features: three spatial coordinates, voxel value, three spatial derivatives, and gradient magnitude. Finally, we codified the Discrete Adaboost testing classifier as described in previous sections in Matlab, C++, OpenMP, GSGL, and OpenCL codes. We measure the performance in terms of the mean execution time from 500 code runs on the same hardware. Our preliminary results confirm our expectations of time consuming and interactivity of the classification stage. In Table 1, we show the averaged times of the five implementations with the different sized datasets. Our proposed OpenCL-based optimization has a speed up of 89.91x over a C++ CPU-based algorithm and a speed up of 8.01x over the GLSL GPU-based algorithm.

Dataset	Size	Mathlab	CPU-based	OpenMP	GLSL	OpenCL
Foot	128x128x128	18.32s	9.63s	8s	1.32s	0.1256s
Hand	244x124x257	67.29s	26s	20s.	2.86s	0.1653s
Thorax	400x400x400	114.28s	33.76s	25s	4.41s	1.9253s

Table 1. Testing step times in seconds of the different datasets with the five implementations. GLSL and OpenCL times has been obtained using the GTX470 graphic card.

We also achieved PCIe transfers 16 times faster than the previous GLSL version. The introduction of gradient calculation into the GPU also added a speedup having 0,0005 seconds of execution time for 128^3 voxel models in a NVIDIA Geforce GTX 470. The execution times for the classification kernel are less than half that of the GLSL version.

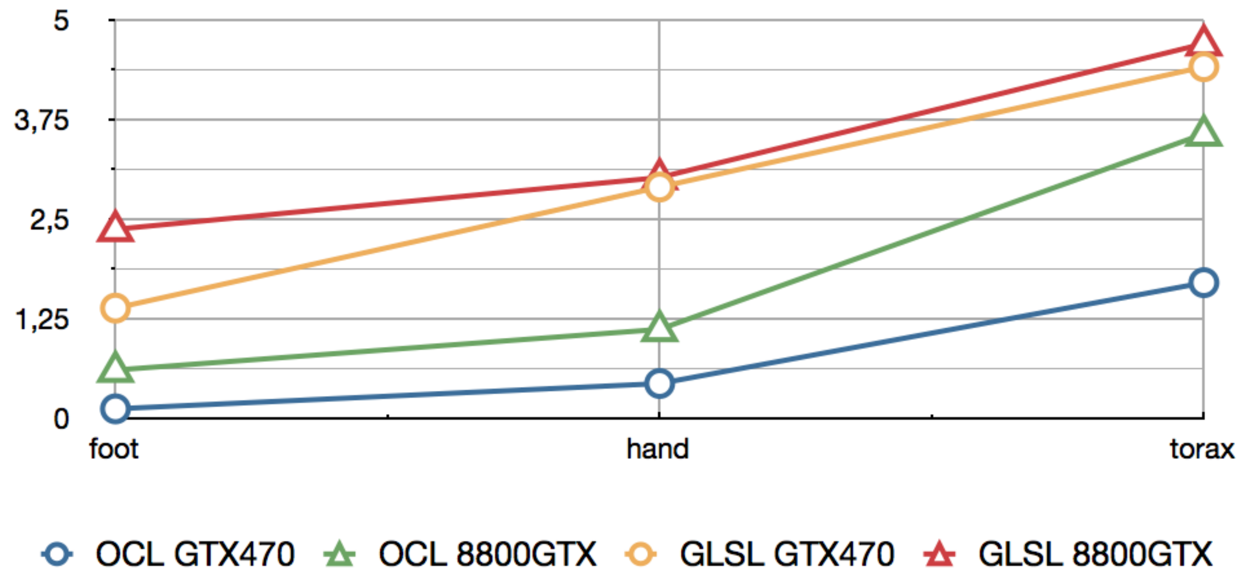


Figure 2. Performance of GLSL and OpenCL implementations with different GPUs.

5.1 GPU-Hardware comparison

As seen in Figure 2 there is a difference in performance evolution when increasing the number of voxels to be classified between the Geforce 8800GTX and the Geforce GTX470 of our GLSL and OpenCL implementations. Basically besides the better performance, the GTX470 appears to have a more linear behavior.

At hardware level there are tree main differences between the 8800GTX and GTX470 cards. The number of Compute Units (CU), 16 for the 8800GTX in contrast to 14 for the GTX470, the number of Processing Elements (PE) 8 for the 8800GTX and 32 for the GTX470, and the L1 and L2 caches only present in the Fermi architecture (GTX470).

Having less and faster CU's with 8 maximum instantiated WG per CU in both cards, should result in a faster execution. The difference between execution times could decrease with more voxels due to the greater coarse grain parallelism of the 8800GTX, but the results show the opposite behavior. The reason can be the caches avoiding repeated GM reads that happen with our design. When reading 1 unsigned char, 64 of them are loaded into the L2 cache and the L1 cache and the next 63 reads will provably be from L2 or L1 caches. That alleviates the GM bottleneck, and allows to exploit the increased throughput of having dual issue 32 PE Compute Units. That is coherent with the results observed.

6. CONCLUSIONS

Our representation of the weak-classifiers guarantees the equivalence to the classical approach. In addition, our breakthrough is a proved optimization of the labeling process involved in several biomedical applications. This optimization increases the interactively of these processes, and also can be easily integrated in the clinical routine.

In this paper, we have presented an optimization of the Adaboost binary classifier based on GPU-OpenCL. We have defined an alternative representation of the Adaboost binary classifier and we have used this proposed representation to write a new GPU-OpenCL parallelized Adaboost testing stage. The numerical experiments based on large available data sets and the performed comparisons with CPU-implementations confirm our expectations. As a future work, we plan to extend this optimization to a multi-classification strategy.

ACKNOWLEDGMENTS

This work has been partially funded by the project CICYT TIN2008-02903, by the research centers CREB of the UPC and the IBEC and under the grant SGR-2009-362 of the Generalitat de Catalunya. This work has also been supported in part by projects TIN2009-14404-C02, CONSOLIDER INGENIO CSD 2007-00018, and the CASE department of Barcelona Supercomputing Center.

REFERENCES

[1] Pfister, H., Lorensen, B., Baja, C., Kindlmann, G., Shroeder, W., Avila, L., Raghu, K., Machiraju, R., and Lee, J., "The transfer function bake-off," *IEEE Computer Graphics & Applications* **21**(3), 16–22 (2001).

[2] Bajaj, C. L., Pascucci, V., and Schikore, D., "The contour spectrum," in [*IEEE Visualization'97*], 167–174 (1997).

[3] Fujishiro, I., Azuma, T., and Takeshima, Y., "Automating transfer function design for comprehensible volume rendering based on 3d field topology analysis.," in [*IEEE Visualization*], 467–470 (1999).

[4] Levoy, M., "Display of surfaces from volume data," *IEEE Computer Graphics & Applications* **8**, 29–37 (May 1988).

[5] Kindlmann, G. and Durkin, J., "Semi-automatic generation of transfer functions for direct volume rendering," in [*IEEE Symposium on Volume Visualization*], 79–86, IEEE Press (October 1998).

[6] Tzeng, F.-Y. and 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).

[7] Tzeng, F. Y., Lum, E., and Ma, K. L., "An intelligent system approach to higher-dimensional classification of volume data," *IEEE Transactions on Visualization and Computer Graphics* **11**, 273–284 (2005).

[8] Laidlaw, D. H., Fleischer, K. W., and Barr, A. H., "Partial-volume bayesian classification of material mixtures in MR volume data using voxel histograms," *IEEE Trans. Med. Ima* **17**, 74–86 (1998).

[9] Leemput, K. V., Maes, F., Vandermeulen, D., and Suetens, P., "Automated model-based tissue classication of mr images of the brain," *IEEE Transactions on Medical Imaging* **18(10)**, 897–908 (1999).

[10] Tzeng, F. Y., Lum, E., and Ma, K. L., "A novel interface for higher dimensional classification of volume data," in [*Visualization 2003*], 16–23, IEEE Computer Society Press (2003).

[11] Ferré, M., Puig, A., and Tost, D., "Decision trees for accelerating unimodal, hybrid and multimodal rendering models," *The Visual Computer* **3**, 158–167 (2006).

[12] Christiaan Gribble, Xavier Cavin, M. H. and Hansen, C., "Cluster-based interactive volume rendering with simian," *Technical Paper, INRIA Lorraine* (2003).

[13] Gerig, G., Martin, J., Kikinis, R., Kubler, O., Shenton, M., and Jolesz, F., "Unsupervised tissue type segmentation of 3-d dual-echo mr head data.," *Image and Vision Computing* **10**(6), 349–36 (1992).

[14] Garcia, V., Debreuve, E., and Barlaud, M., "Fast k nearest neighbor search using gpu," in [*In Proceedings of the CVPR Workshop on Computer Vision on GPU*], (2008).

[15] Catanzaro, B., Sundaram, N., and Keutzer, K., "Fast support vector machine training and classification on graphics processors," in [*Proceedings of the 25th international conference on Machine learning*], 104–111 (2008).

[16] Yang, D., Getao, L., Jenkins, D., Peterson, G., and Li, H., "High performance relevance vector machine on gpus," in [*Symposium on Application Accelerators in High Performance Computing*], (2010).

[17] Herrero, S., Williams, J., and 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*], **425**, 2–11 (2010).

[18] Yudanov, D., Shaaban, M., Melton, R., and 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).

[19] Shimizu, A., Narihira, T., Furukawa, D., Kobatake, H., Nawano, S., and Shinozaki, K., "Ensemble segmentation using AdaBoost with application to liver lesion extraction from a CT volume," *The Midas Journal* (2008).