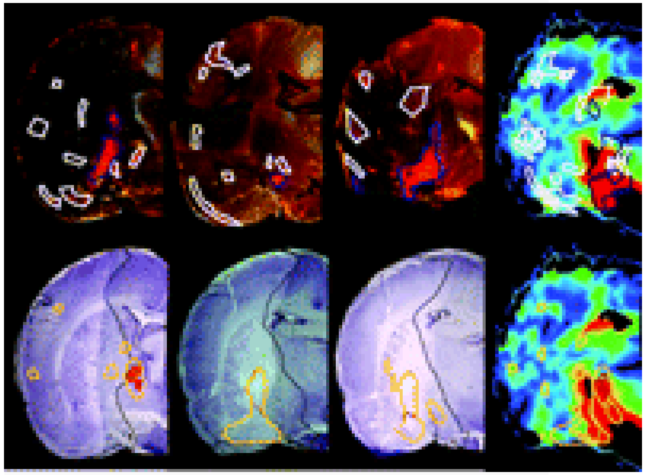
Intelligent GPGPU Classification in Volume Visualization: A framework based on Error-Correcting Output Codes

Dr. Sergio Escalera[1,2], (Computer Vision and Machine Learning)
Dr. Anna Puig[1,3], (Volume Visualization)
Mr. Oscar Amorós[1], and (GPGPU Computing)
Dr. Maria Salamó[1,3] (Data Mining)

[1] Facultat de Matemàtiques, Universitat de Barcelona
[2] Centre de Visió per Computador, Universitat Autònoma de Barcelona
[3] WAI: Volume Visualization and Artificial Intelligence Research

The 19th Pacific Conference on Computer Graphics and Applications
(Pacific Graphics 2011) will be held on September 21 to 23, 2011 in Kaohsiung, Taiwan.

1. Context and motivation



GOAL

Complex classification process

- Increasing dataset complexity
  - Size, modalities, interpretation
- Data understanding needs to interrelate several properties

2. Related work

- Transfer Functions of different dimensionality
- Artificial Intelligence based techniques:
- **Supervised methods:**
  - bayesian networks
  - neural networks
  - decision trees
- **Semi and non-supervised methods** (such as clustering)

Open issue:
Feature representation
Pattern recognition process

2. Related work

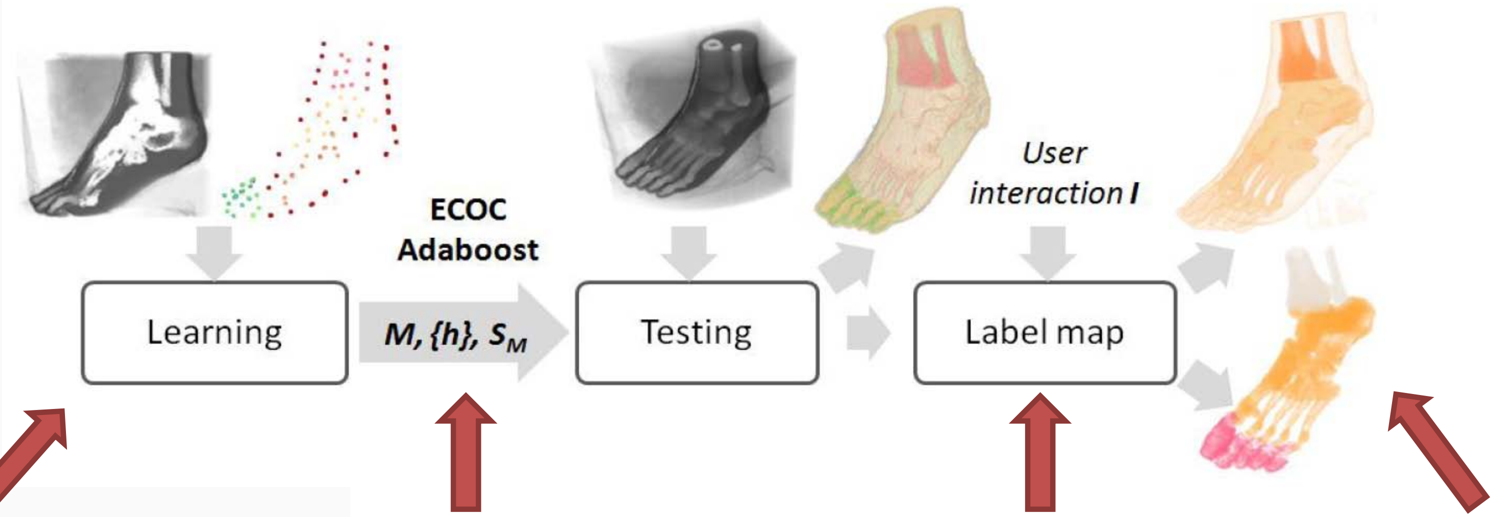
- New GPGPU implementations for binary classifications in image processing applications:
  - Clustering strategies (k-nearest neighbor similarity)
  - Geometrical Support Vector Machine classifier (SVM)
  - Adaboost classifier
  - Neural Networks

Binary classification

Need of high accurate multi-class labeling

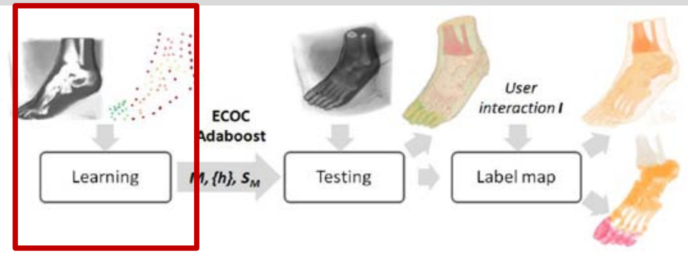
Error-Correcting Output Codes (ECOC) is a general framework to deal with multi-class categorization problems.
ECOC extends any classifier to the multi-class case.

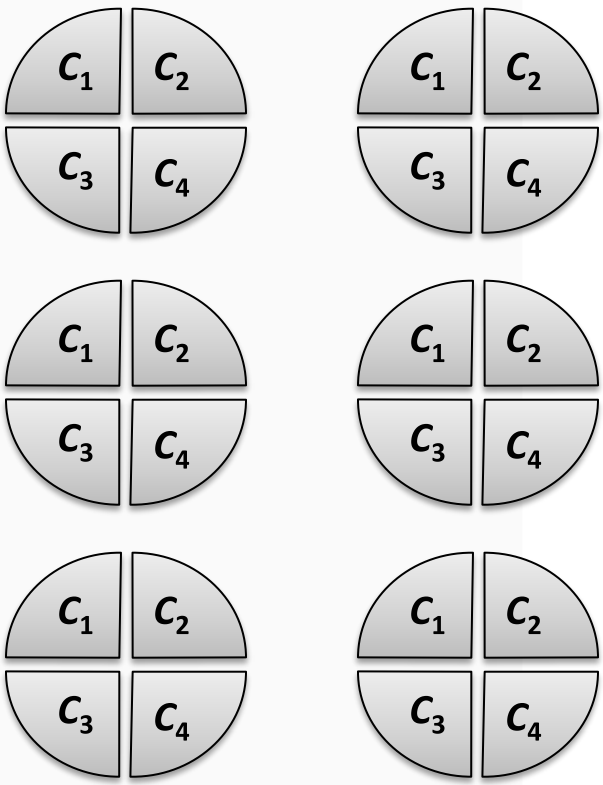
3. Framework

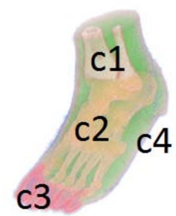


- To define a general framework for multi-classification volume models based on the **Error-Correcting Output Codes**
- To use of **Adaboost** as a case study of this general framework
- To compute an **on-demand an adaptive segmentation** / classification a subset of features of interest
- To obtain an **efficient GPGPU OpenCL implementation of the testing stage of the multi-classifier integrated into the final rendering**

3.1 Multi-class Learning



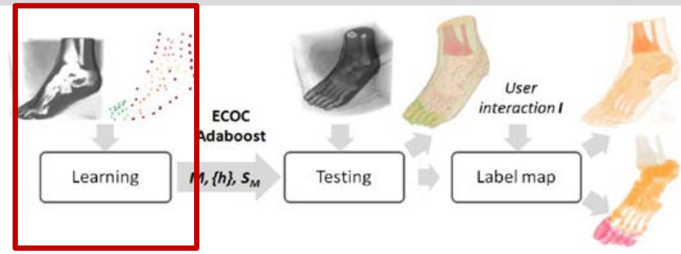




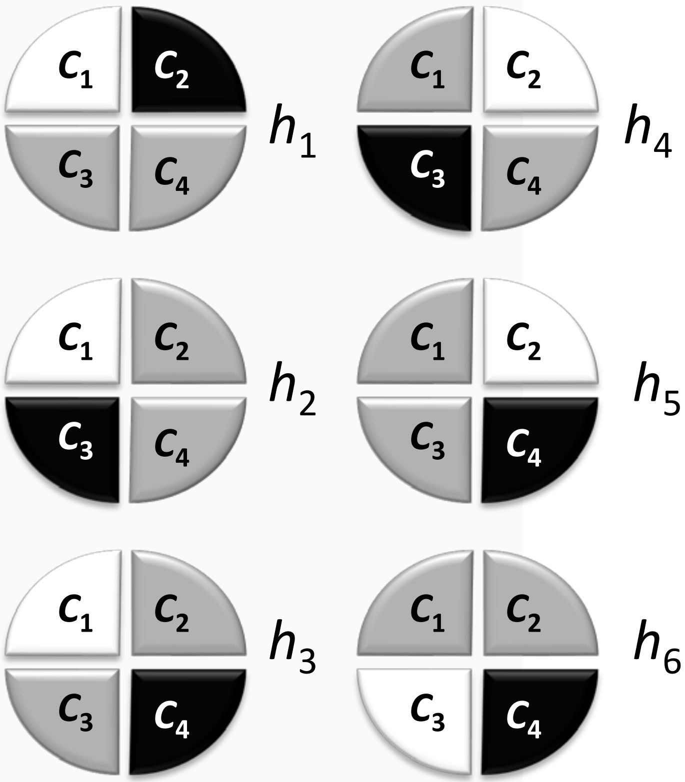
- Multi-class as a combination of binary classifiers (dichotomizers)
- Classical 1vs1 1vsall voting
- Error-Correcting Output Codes

[Dietterich95] Thomas G. Dietterich, Ghulum Bakiri, Solving Multiclass Learning Problems via Error-Correcting Output Codes, Journal of Artificial Intelligence Research, 1995

3.1 Multi-class Learning

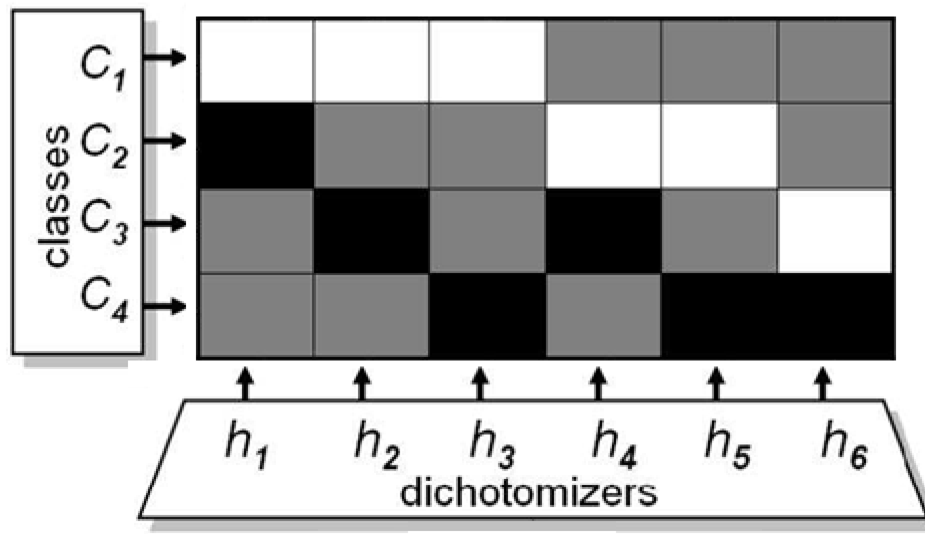


one-versus-one

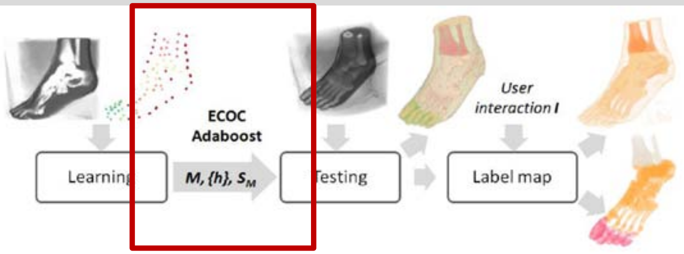


=1 =-1 = 0

Training



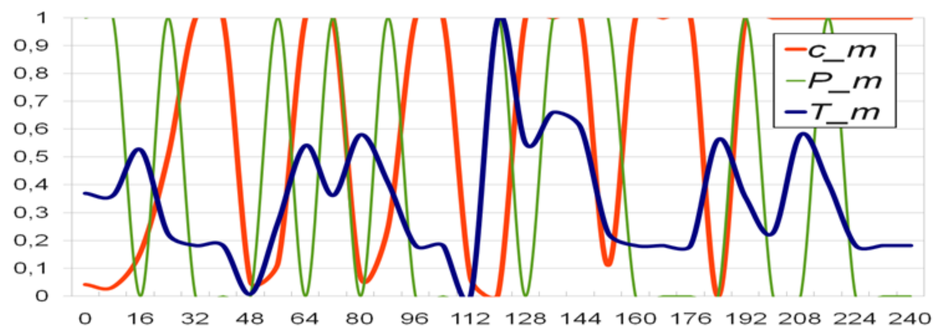
3.1 Multi-class Learning



- **Dichotomizer h_i:**
- **Adaboost: based on a weak classifier:**
- Additive model combining simple decisions to define a strong binary classifier
- Inherent parallel structure
- High performance, simple to train

Algorithm 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 $\mathrm{sign}(F(\rho))$



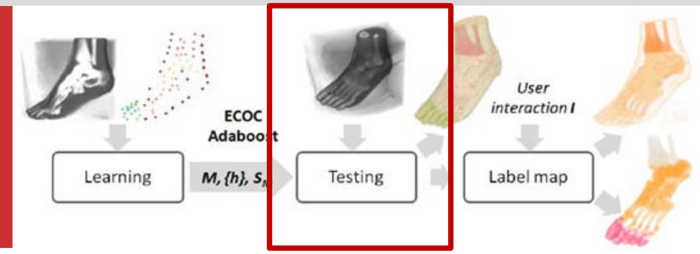
Transfer function representation of the classifier

- Adaboost has a high potential for GPU applications

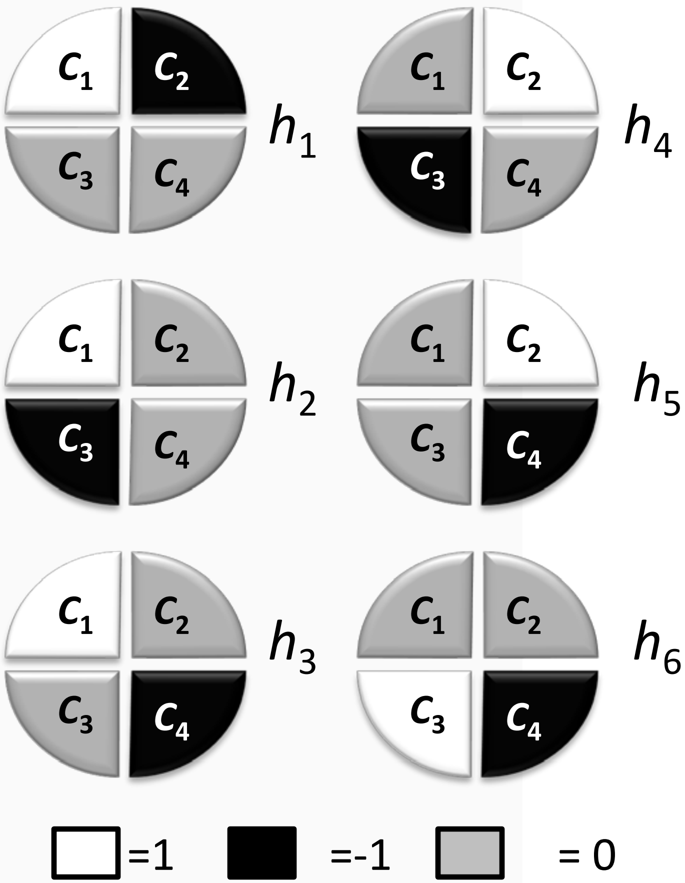
pg2011

Kaohsiung, Taiwan

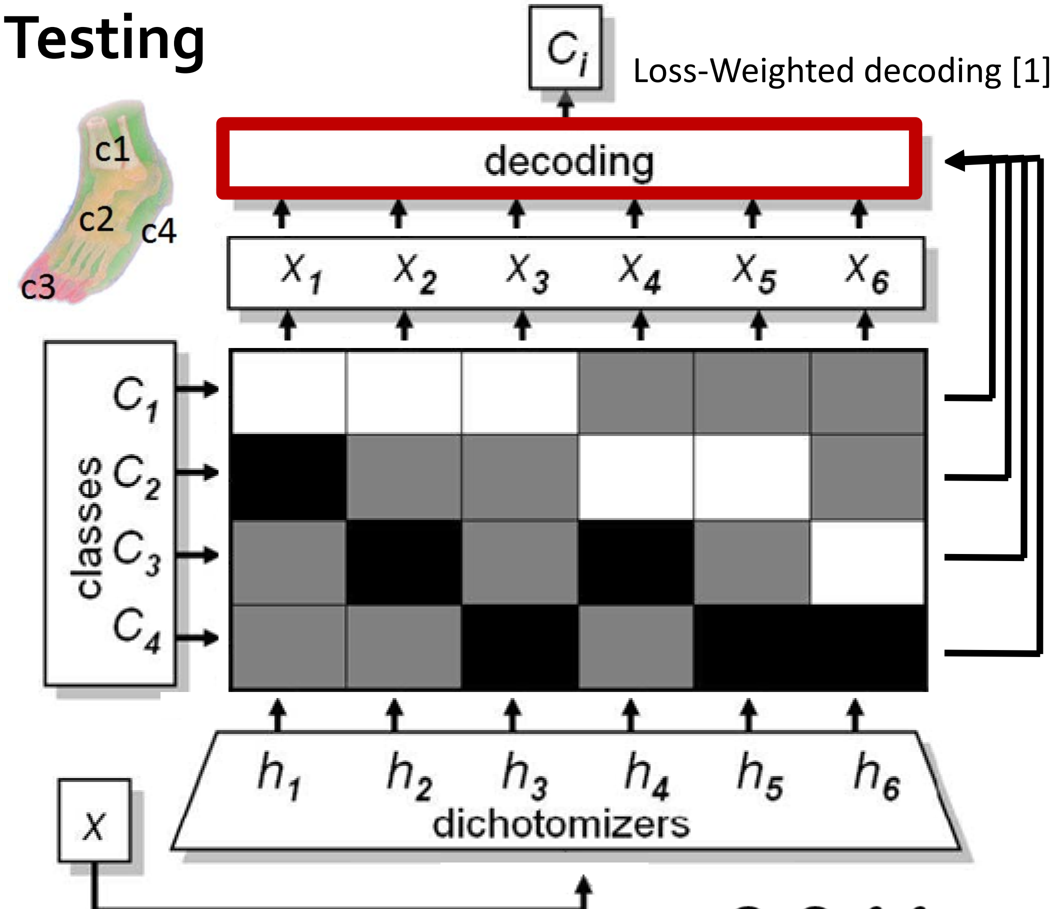
3.2 Testing and label mapping



one-versus-one



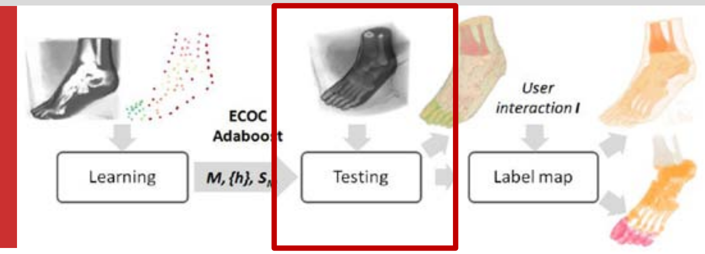
Testing



pg2011 Kaohsiung, Taiwan

[1] Sergio Escalera, Oriol Pujol, and Petia Radeva, On the Decoding Process in Ternary Error-Correcting Output Codes, Transactions in Pattern Analysis and Machine Intelligence, vol. 32, issue 1, pp. 120-134, IEEE Computer Society, New York, ISSN 0162-8828, 2010.

Decoding function details



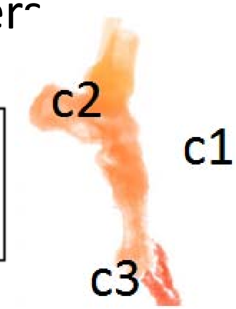
- **Decoding decomposition (Loss-Weighted [1])**, defining a matrix that weights the decoding process of any subgroup of binary classifiers

$\boxed{d = M_W \cdot T}$, T *decoding measure*

Step 1

- Compute the matrix of hypothesis H

$$M = \begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix}$$



$$H = \begin{bmatrix} 0.955 & 0.955 & 1.000 & 0.000 \\ 0.900 & 0.800 & 0.000 & 0.000 \\ 1.000 & 0.905 & 0.805 & 0.805 \end{bmatrix}$$

Step 2

- Normalize H to obtain a matrix M_W of PDF-rows

$$M_W = \begin{bmatrix} 0.328 & 0.328 & 0.344 & 0.000 \\ 0.529 & 0.471 & 0.000 & 0.000 \\ 0.285 & 0.257 & 0.229 & 0.229 \end{bmatrix}$$

Step 3

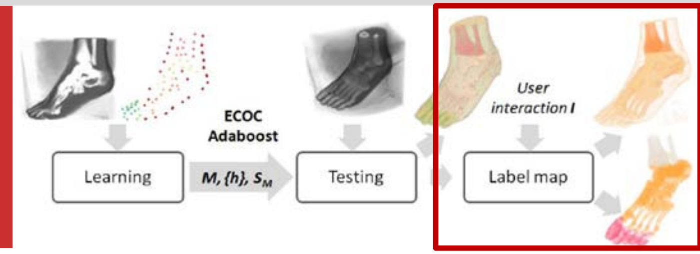
- Use M_W to weight the decoding process in a Loss-based decoding

$$LW(\rho, i) = \sum_{j=1}^{n} M_W(i,j) L(y_i^j \cdot x_j)$$

$$L(\theta) = e^{-\theta}$$

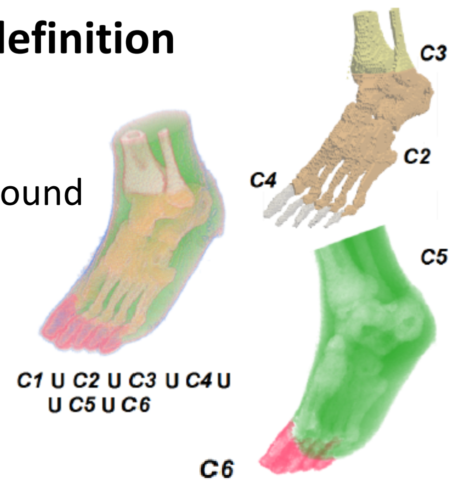
[1] Sergio Escalera, Oriol Pujol, and Petia Radeva, On the Decoding Process in Ternary Error-Correcting Output Codes, Transactions in Pattern Analysis and Machine Intelligence, vol. 32, issue 1, pp. 120-134, IEEE Computer Society, New York, ISSN 0162-8828, 2010.

3.2 Testing and label mapping



Submatrix definition

C1 = Background



M

	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}	
y_1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	→ 0.5
y_2	-1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	→ 4.5
y_3	0	-1	0	0	0	-1	0	0	0	1	1	1	0	0	0	→ 5.5
y_4	0	0	-1	0	0	0	-1	0	0	-1	0	0	1	1	0	→ 5.5
y_5	0	0	0	-1	0	0	0	-1	0	0	-1	0	-1	0	1	→ 5.5
y_6	0	0	0	0	-1	0	0	0	-1	0	0	-1	0	-1	-1	→ 5.5
x	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	

S_M

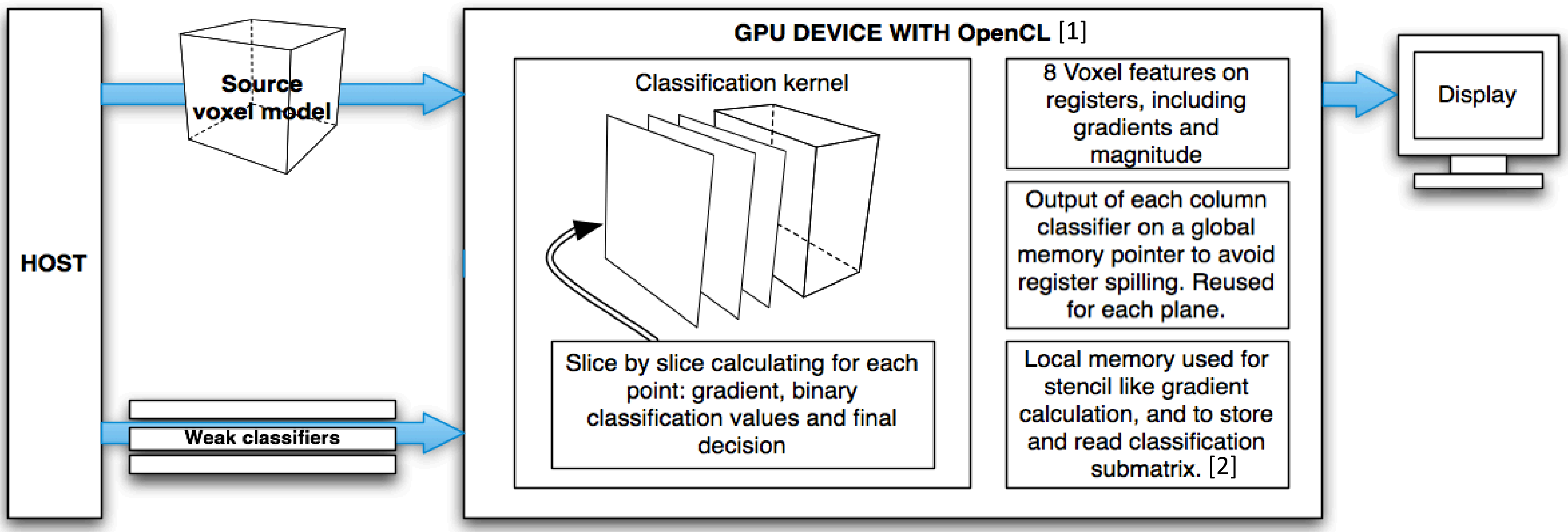
	h_2	h_5	h_6	h_9	h_{10}	h_{11}	h_{12}	h_{14}	h_{15}
y_1	1	1	0	0	0	0	0	0	0
y_2	0	0	1	1	0	0	0	0	0
y_3	-1	0	-1	0	1	1	1	0	0
y_4	0	0	0	0	-1	0	0	1	0
y_5	0	0	0	0	0	-1	0	0	1
y_6	0	-1	0	-1	0	0	-1	-1	-1

{C3,C6}->L1

$$I = \{\{c_1, c_2, c_4, c_5\}, \{c_3, c_6\}\}$$

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

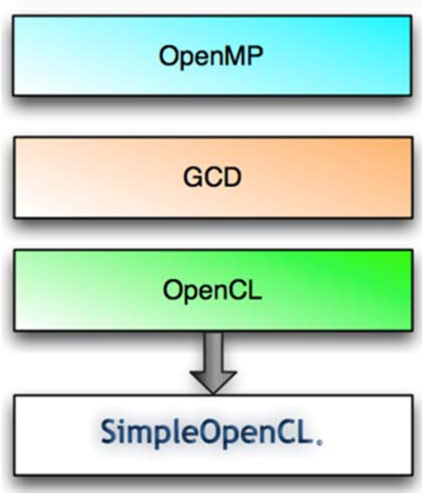
3.3 GPGPU Implementation



[1] SimpleOpenCL, http://code.google.com/simple-opencl/
[2] Paulius Micikevicius "3D Finite Difference Computation on GPUs using CUDA"

4. Results

Data: Thorax (400x400x400), Foot (128x128x128), Brain (256x256x256)
Features: Standard features (8): x,y,z, gradient components and magnitude, and intensity value.
Implementations: C++ ⇨ CPU



OpenMP ⇨ CPU

GCD ⇨ CPU

OpenCL ⇨ CPU + GPU

Development eased by using SimpleOpenCL.
Project available on **Google Code**
http://code.google.com/p/simple-opencl/

Measurements: mean execution time from 500 runs, and accuracy from stratified ten-fold cross-validation with 5% stratified sampling.

Hardware:

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

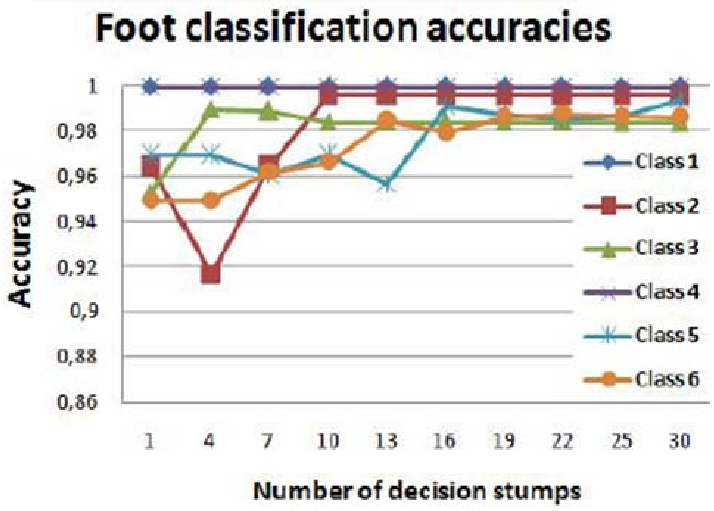
Different CPU configurations used for evaluation

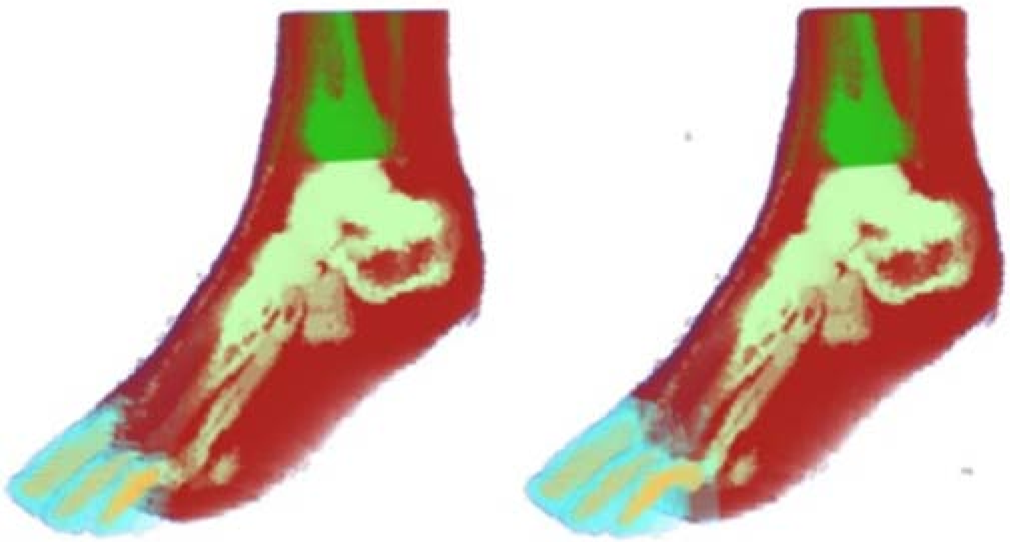
GPU's	ATI	NVIDIA
Processing Elements	720	448
Stream or CUDA cores	144	448
Compute Units	9	14
Max PE per WI	5f / 0d	1f / 1d
PE available per CU	80f / 0d	32f / 4d
Warp size	64 Work Items	32 Work Items
Memory type	GDDR 5	GDDR 5
Global Memory size	1GB	1.28GB
Local Memory size	32KB	16KB or 48KB

Different GPUs architectures used for evaluation, where 'd' and 'f' stands for double and float, respectively

4. Results

Accuracy foot data set



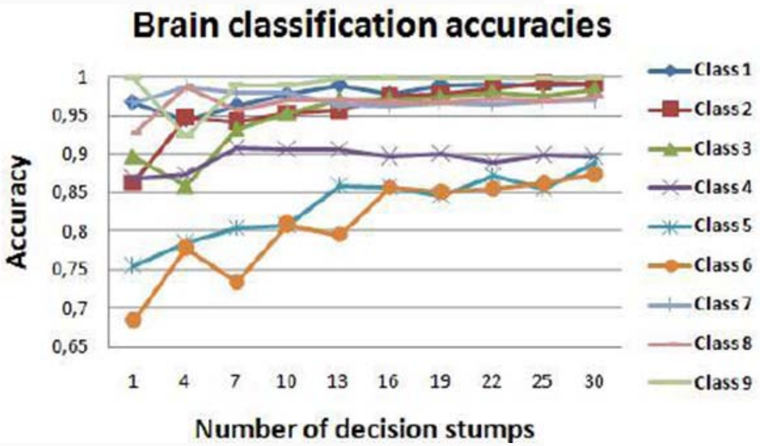


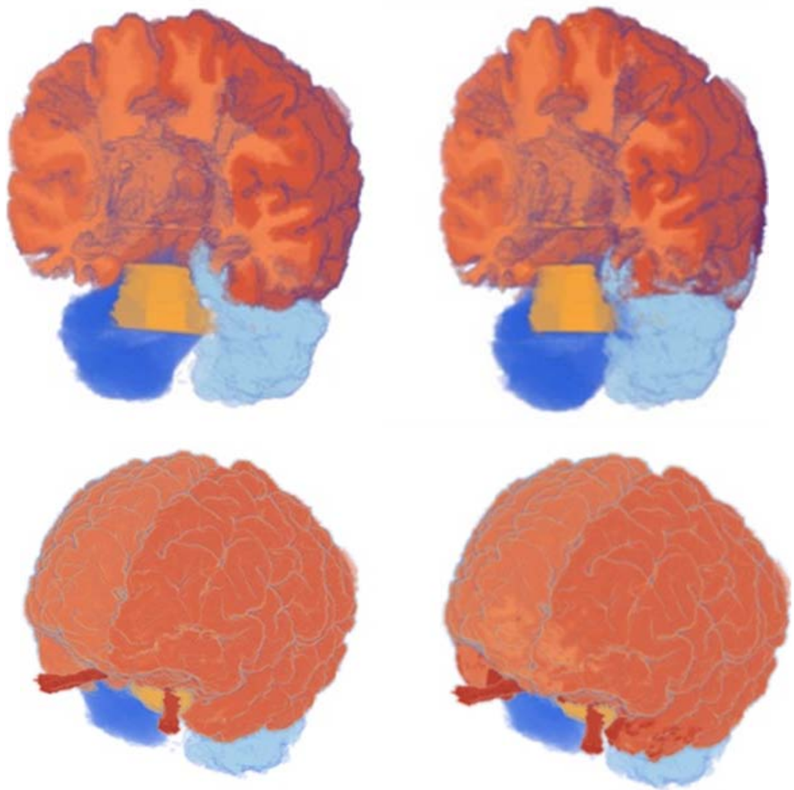
Ground truth data

Labeled data

4. Results

Accuracy brain data set

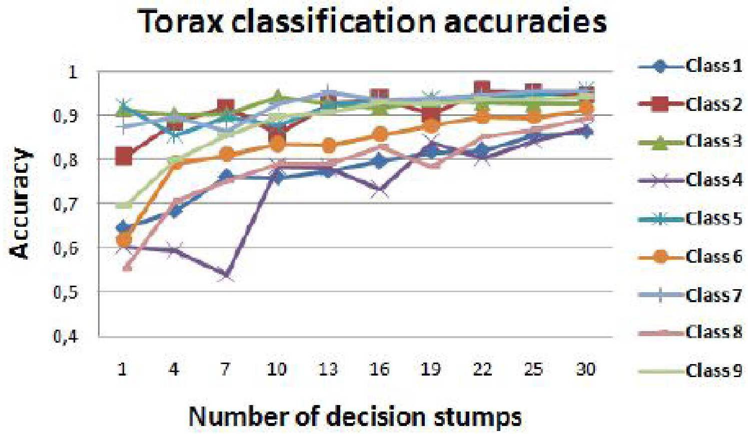


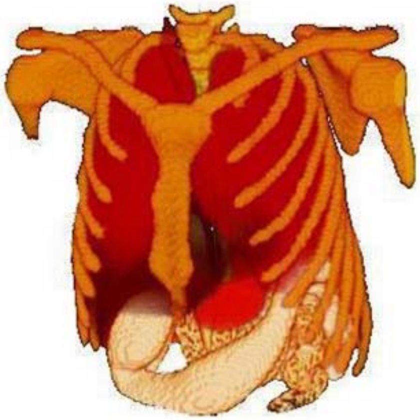


Ground truth data | Labeled data

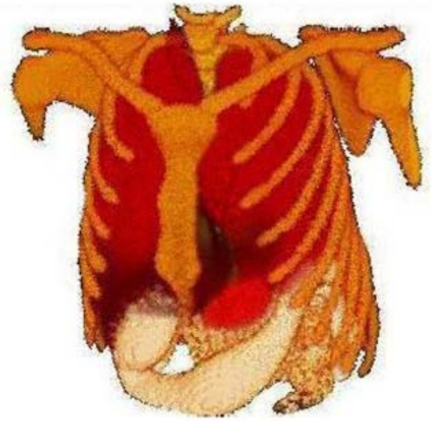
4. Results

Accuracy thorax data set





Ground truth data



Labeled data

4. Results

Execution time

CPU-GPU

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	0.959
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	4.752
	9	7	35	923.225	260.751	259.489	5.955
	9	9	36	971.915	270.751	269.751	7.763

125 FPS

OpenCL on different hardware

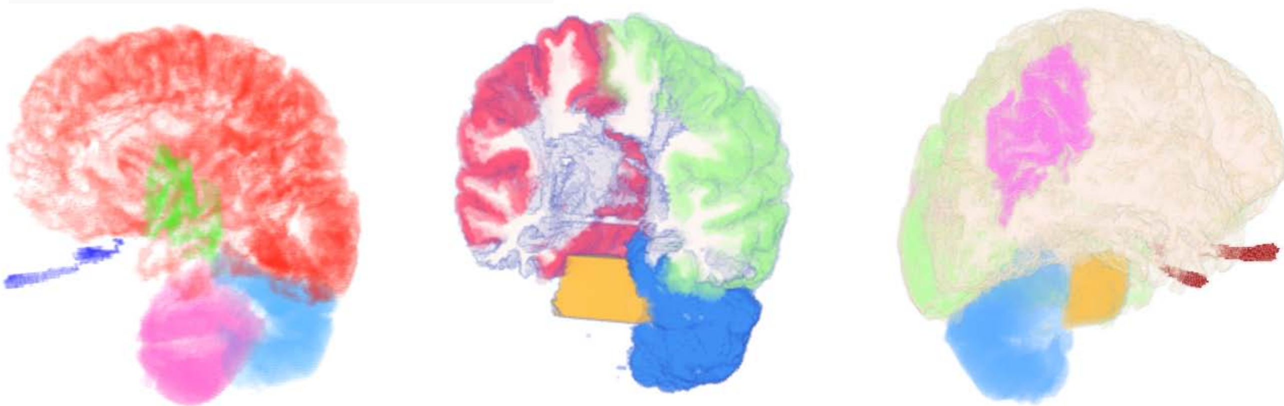
Data set	N	Selected classes	Z	ATI	NVIDIA	AMD Quad
Foot	3	2	2	0.028	0.008	0.236
	4	3	5	0.066	0.020	0.495
Brain	9	2	15	1.498	0.358	7.114
	9	4	26	2.636	0.649	12.393
	9	6	33	3.403	0.848	16.047
	9	9	36	3.818	0.959	18.446
Thorax	8	2	13	8.750	2.011	56.374
	8	4	22	14.860	3.532	95.657
	8	6	27	18.340	4.467	118.978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

x130 C++
x35 OpenMP/GCD
400x400x400x8x36x30 …

4. Results

Label mapping results (qualitative)

$$I = \{\{c_1\},\{c_2\},\{c_4\},\{c_5\},\{c_6\}\},$$
$$I = \{\{c_1\},\{c_2,c_3\},\{c_5\},\{c_6\}\},$$
$$I = \{\{c_1\},\{c_2\},\{c_3\},\{c_4\}\},$$



Different shading techniques

4. Results

Application video sample

5. Conclusion & Future work

- We proposed a **semi-automatic framework for general multiclass volume labeling on demand based on the ECOC framework.**
- The system is decomposed into a two-level **(classification + visualization) GPU-based labeling** algorithm with Adaboost based classifier.
- **Parallelized testing steps** with different programming languages and hardware architectures.
- Empirical results on different data sets shows very good **speed ups of this novel, automatic, and general-purpose multi-decision framework**.
- **Future work**: “Use of different base classifier/segmentation strategies”
- **Future work**: “Feature space representation analysis”
- **Future work**: “Parallelized learning stage” → On-line learning

Thank you for your attention!

Outline

1. Context and motivation
2. Related work
3. Framework
    3.1 Multi-class learning
    3.2 Testing and label mapping
    3.3 GPGPU Implementation
4. Results
5. Conclusion & Future work