

Università di Bologna - CdL in Informatica

Analisi delle immagini
AA 2008/2009

How to build pattern recognition systems

April, 17 2009

Matteo Roffilli

roffilli@cs.unibo.it

<http://www.cs.unibo.it/~roffilli>

Prerequisites

1. Available time for 1 image/frame/etc
2. Available space for hardware: PC, cluster, mainframe, embedded, etc
3. Available budget for hardware: \$\$
 - a) Available memory: 128 MB, 1 GB, 1 TB
 - b) Available processor
4. Available time for developing
5. Available know-how of the developers

Medical imaging PR system

1. 10 sec / image
2. PC
3. 1000 \$
 - a) 2 GB RAM
 - b) <3 GHz dual/quad processor
4. Medium (1-3 years)
5. High-level

WHAT to do

CAD - Computer Aided Detection

GOAL: to aid the physicians/radiologists in **detecting** pathologies

1. to classify unknown images in 2 classes: diseased or healthy
2. to locate the lesion

REQUIREMENT: to find all lesions without prompting false signals

CADx - Computer Aided Diagnosis

GOAL: to aid the physicians/radiologists in **diagnosing** pathologies

1. to categorize detected pathologies in classes (eg: benign, cancer, type I, type II, etc)

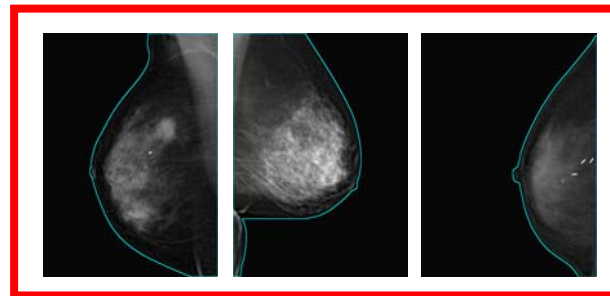
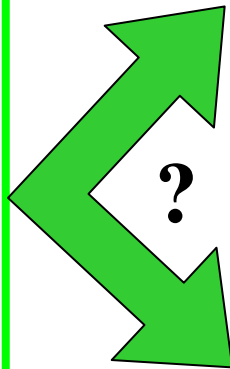
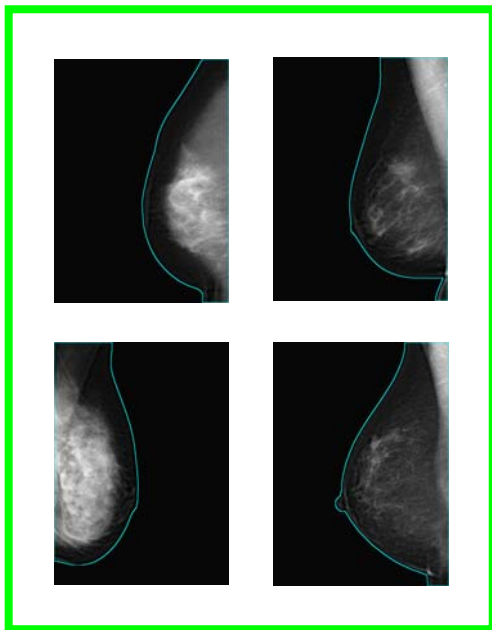
CAD in radiology

CAD GOAL: to aid the radiologist in detecting tumoral masses

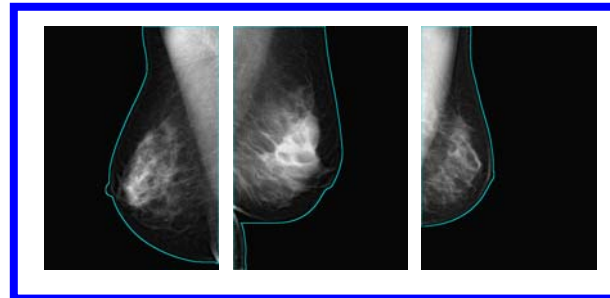
1. to classify unknown images in 2 classes: diseased or healthy
2. to locate the lesions

REQUIREMENT: to find all lesions without prompting false signals

unknown



diseased



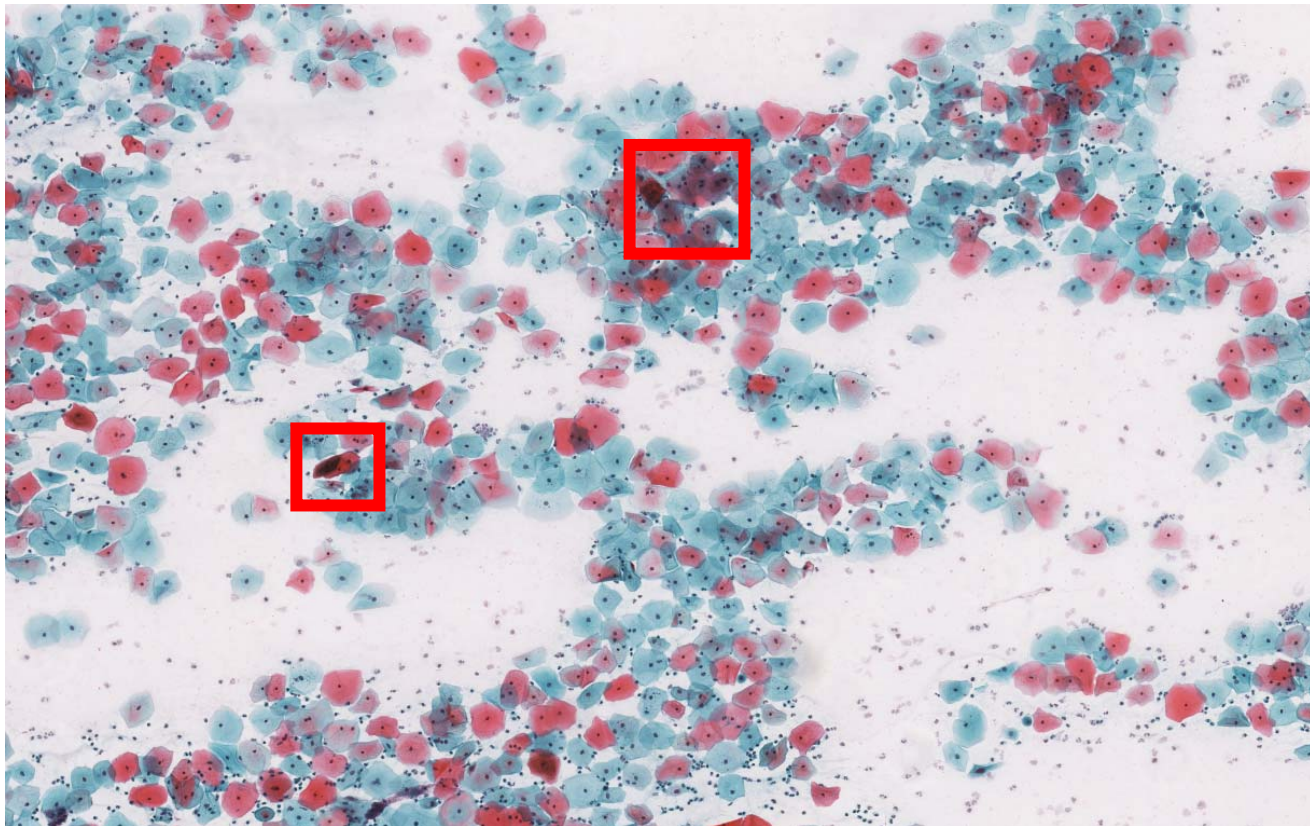
healthy

CAD in cytology

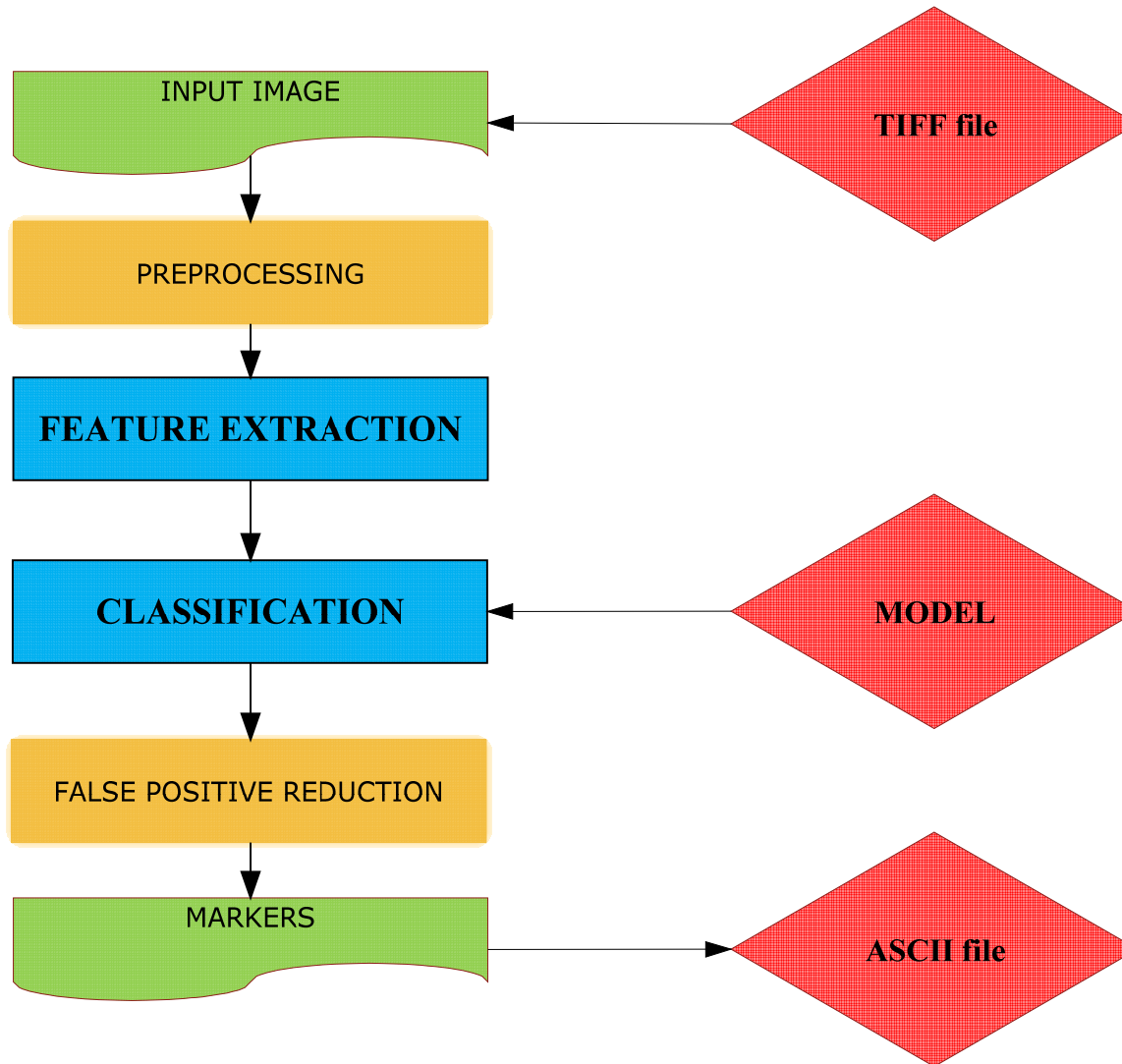
CAD GOAL: to aid the radiologist in detecting tumoral cells

1. to classify unknown images in 2 classes: diseased or healthy
2. to locate the lesions

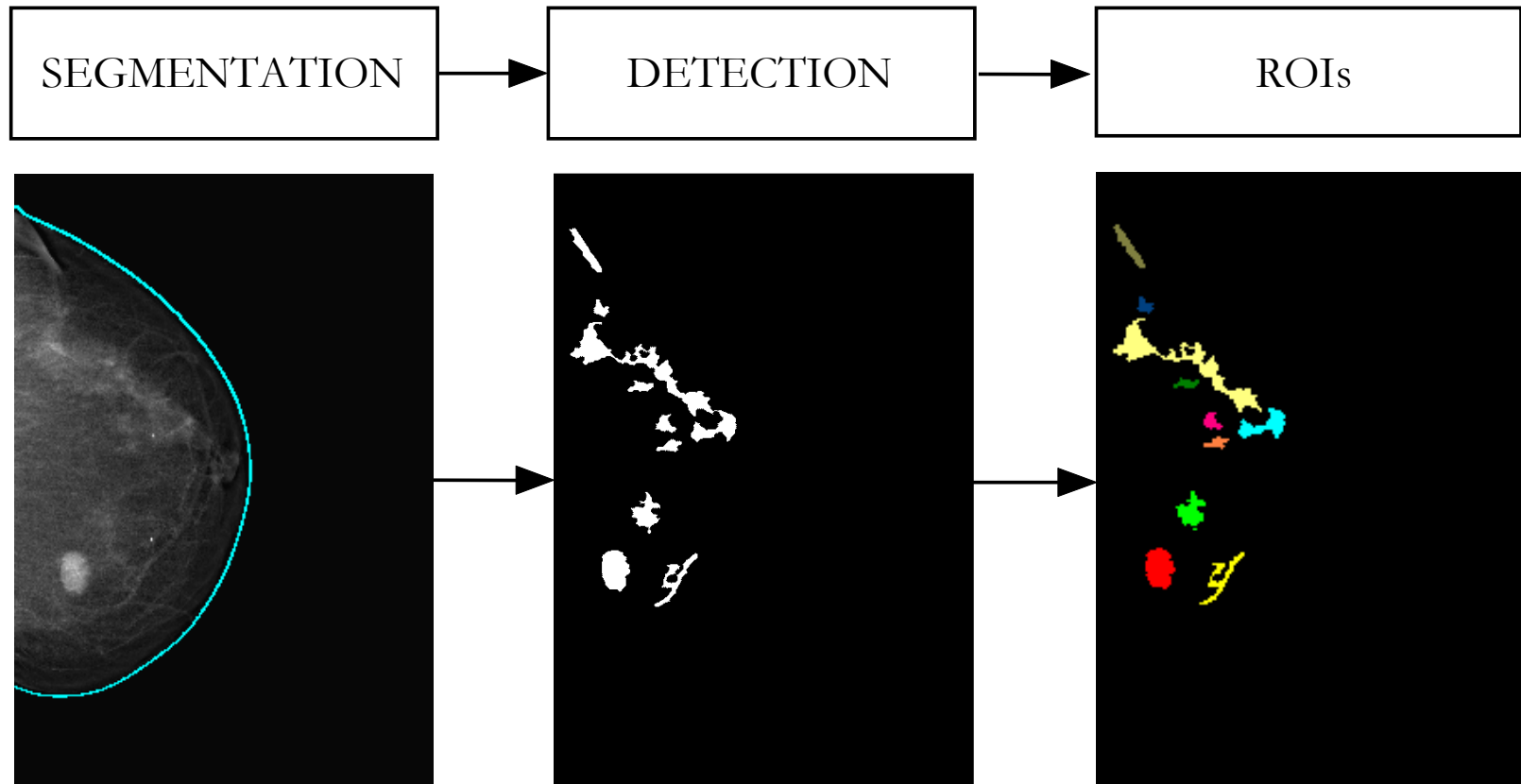
REQUIREMENT: to find all lesions without prompting false signals



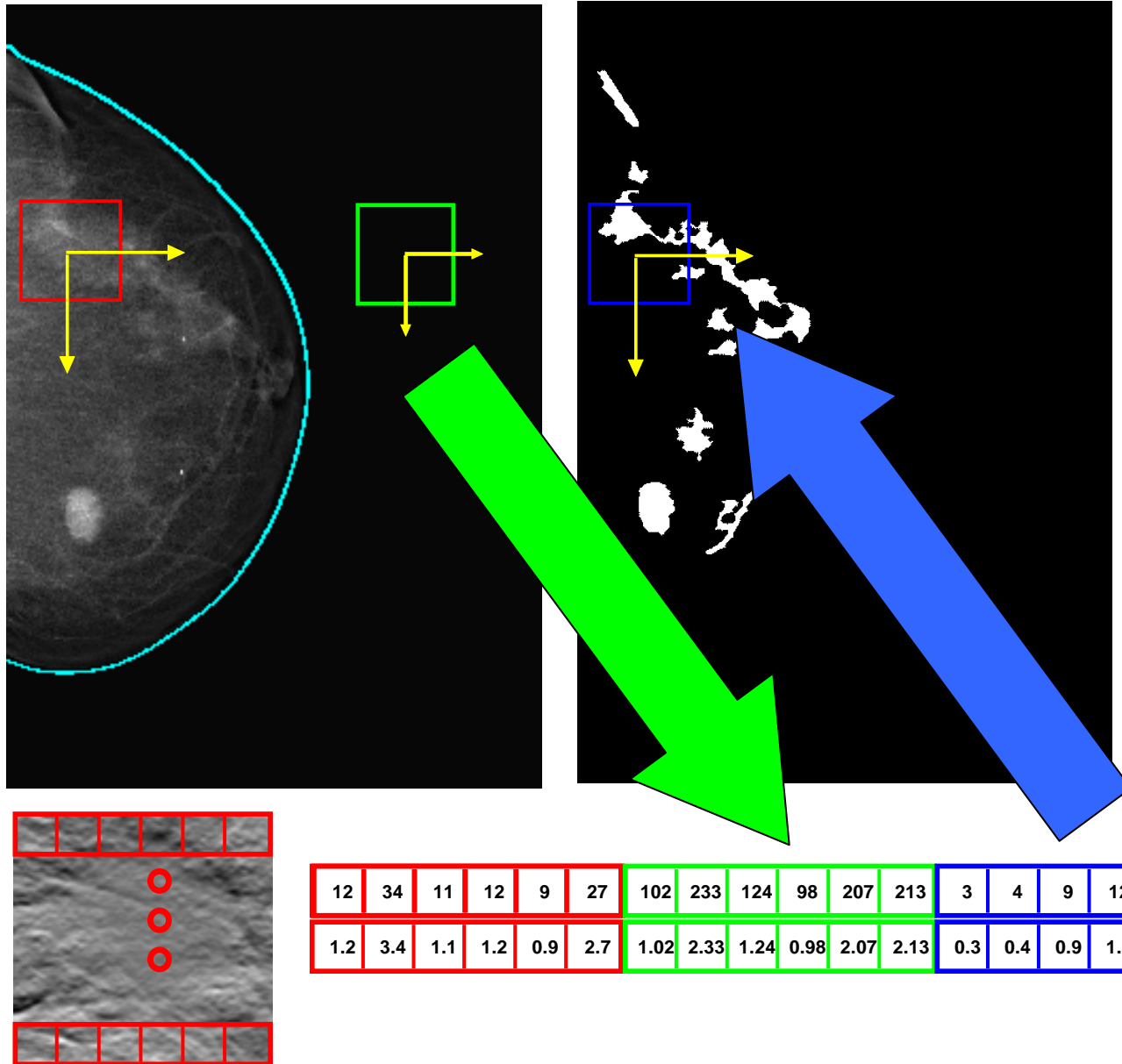
System overview



Classical approach: detection



1. Based on appearance models (external knowledge)
2. Segmenting borders is a difficult task and some difficult masses are lost!
3. About **10-50** Regions Of Interest (ROIs)



Classical approach: Classification

Data representation Feature extraction



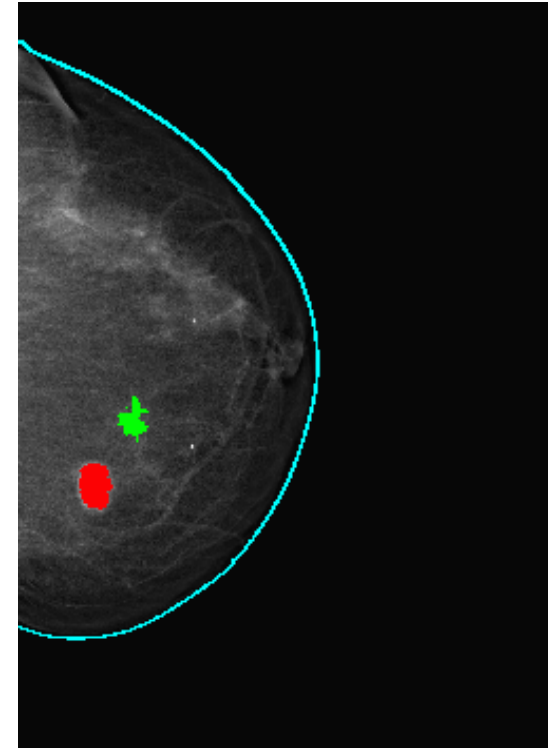
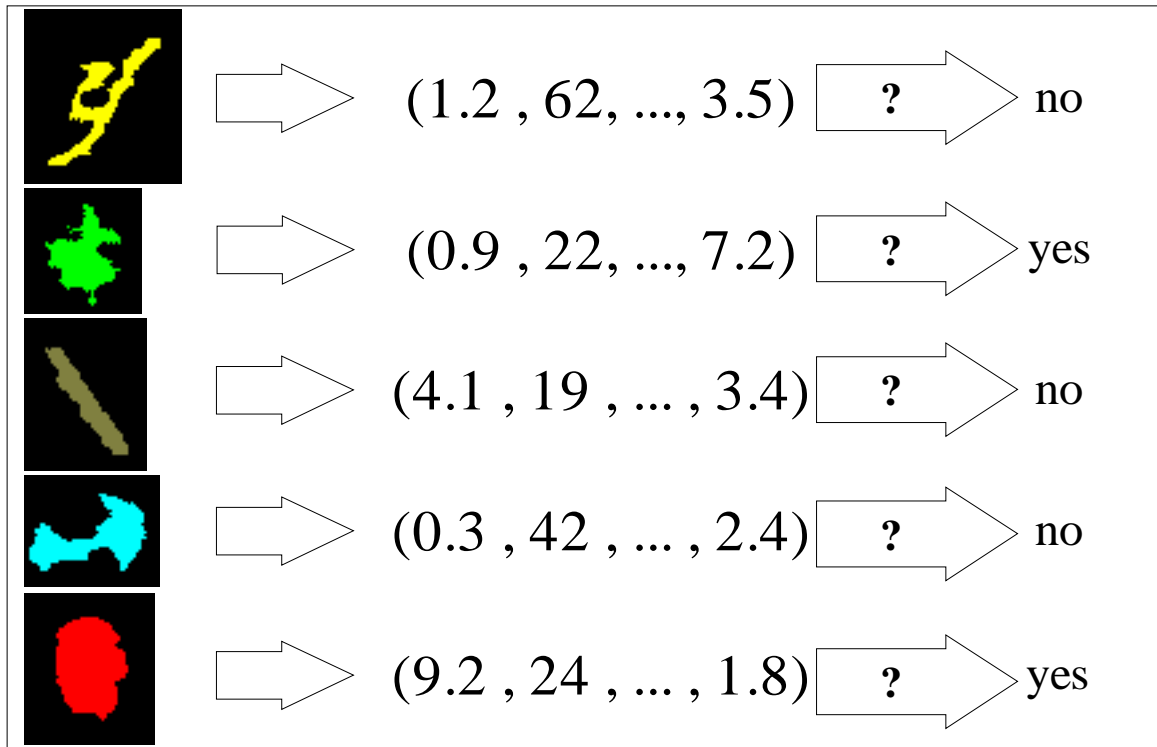
- Area
- Perimeter
- Size
- Intensity
- Shape
- ...

About **10-15** features

Classification

- ANN
- RBF
- Bayesian Networks
- Decision Tree
- Hand-made classifiers

Classical approach: Result



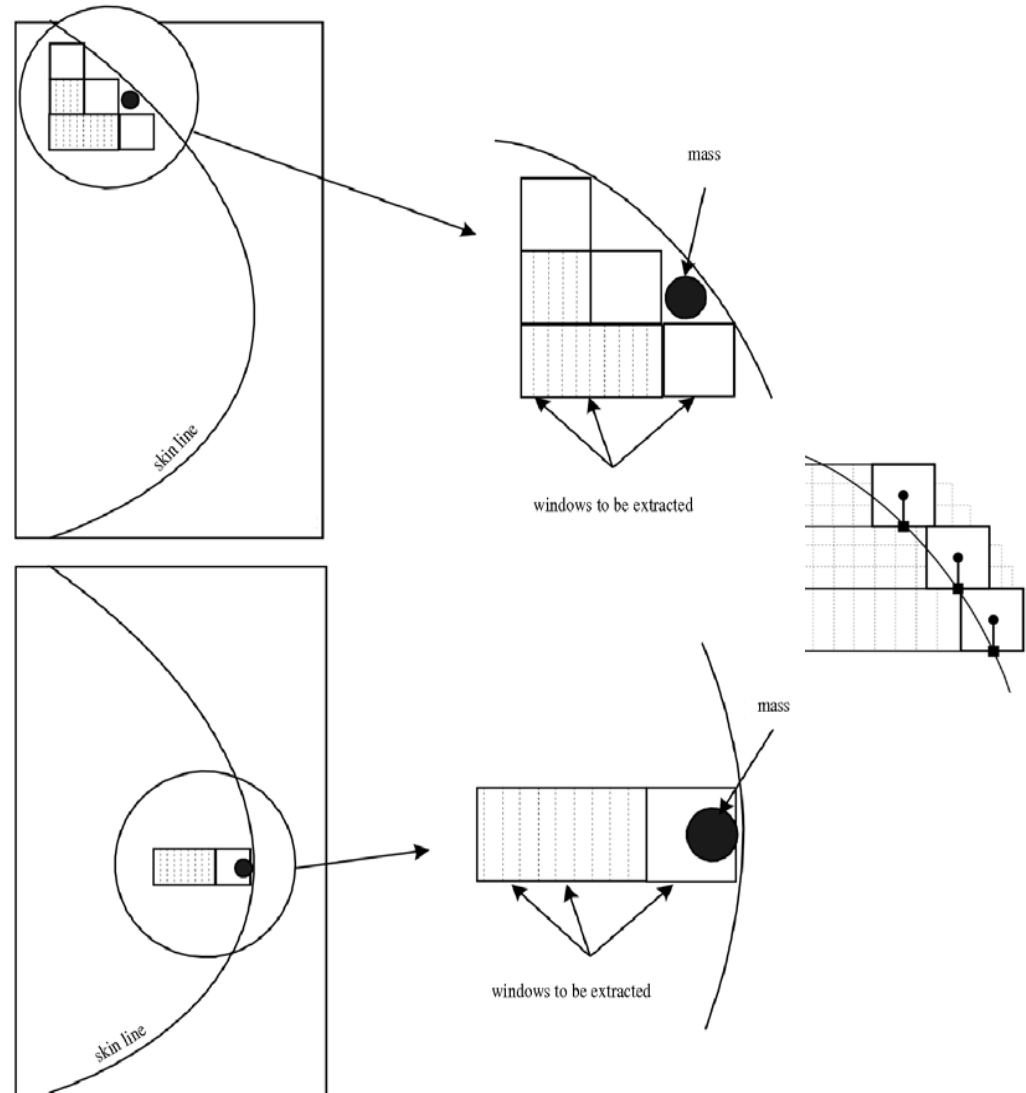
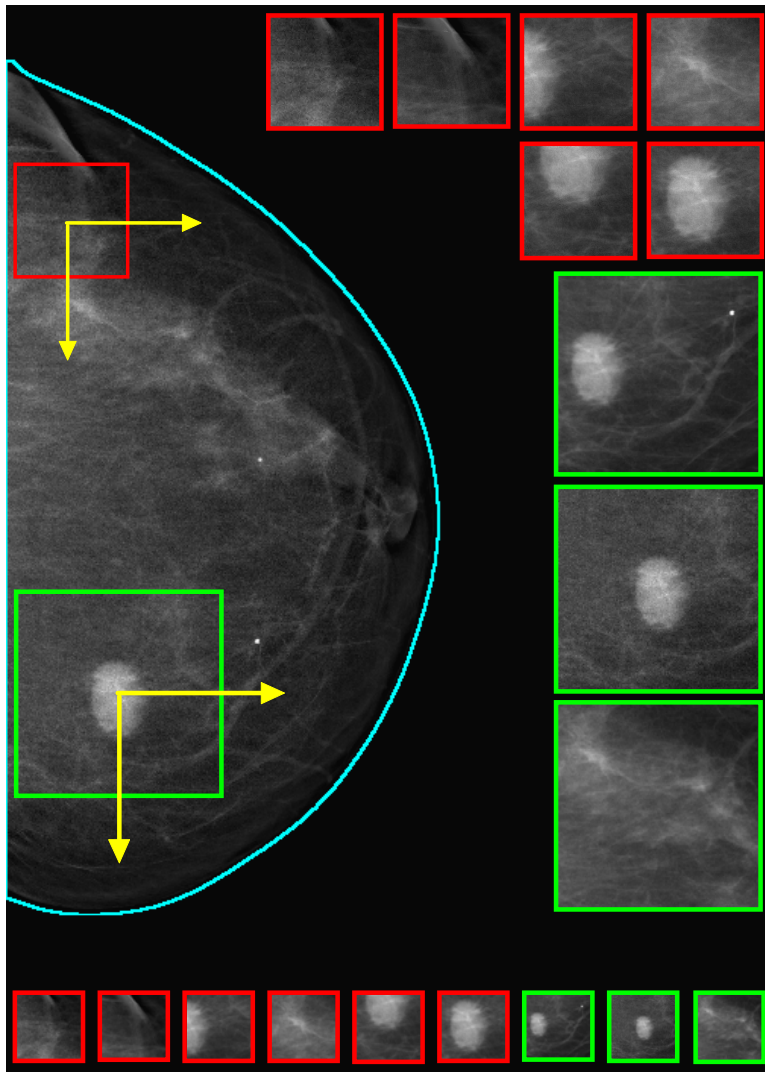
ROIs classified as positive are prompted on the original image

A novel brute force approach

The novel contributions of our work are mainly three:

1. the **detection** step is performed without the use of external knowledge
2. the **feature extraction** step is avoided
3. SVM and RVM are used as **classifiers** for the classification step

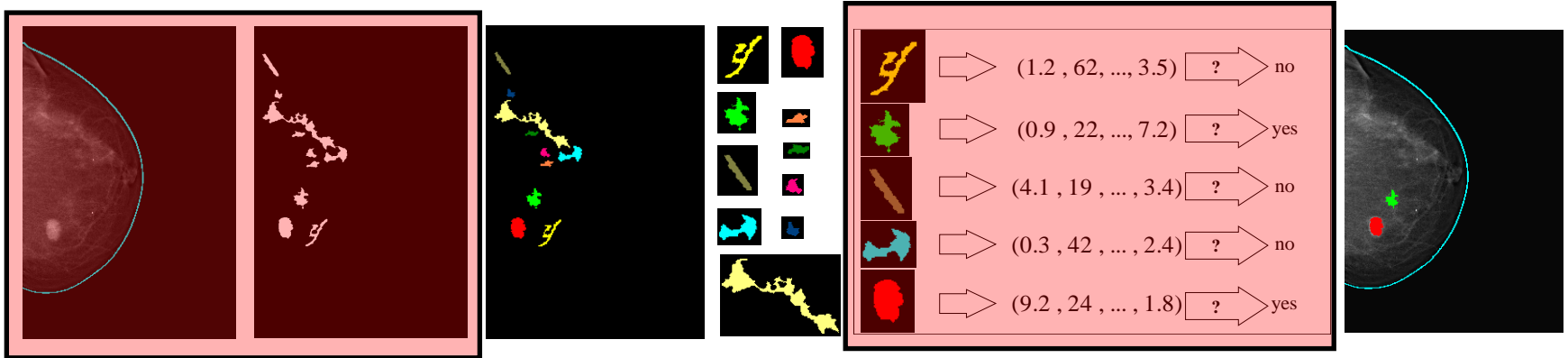
Detection



This detection produces about **100.000** ROIs

HOW to do

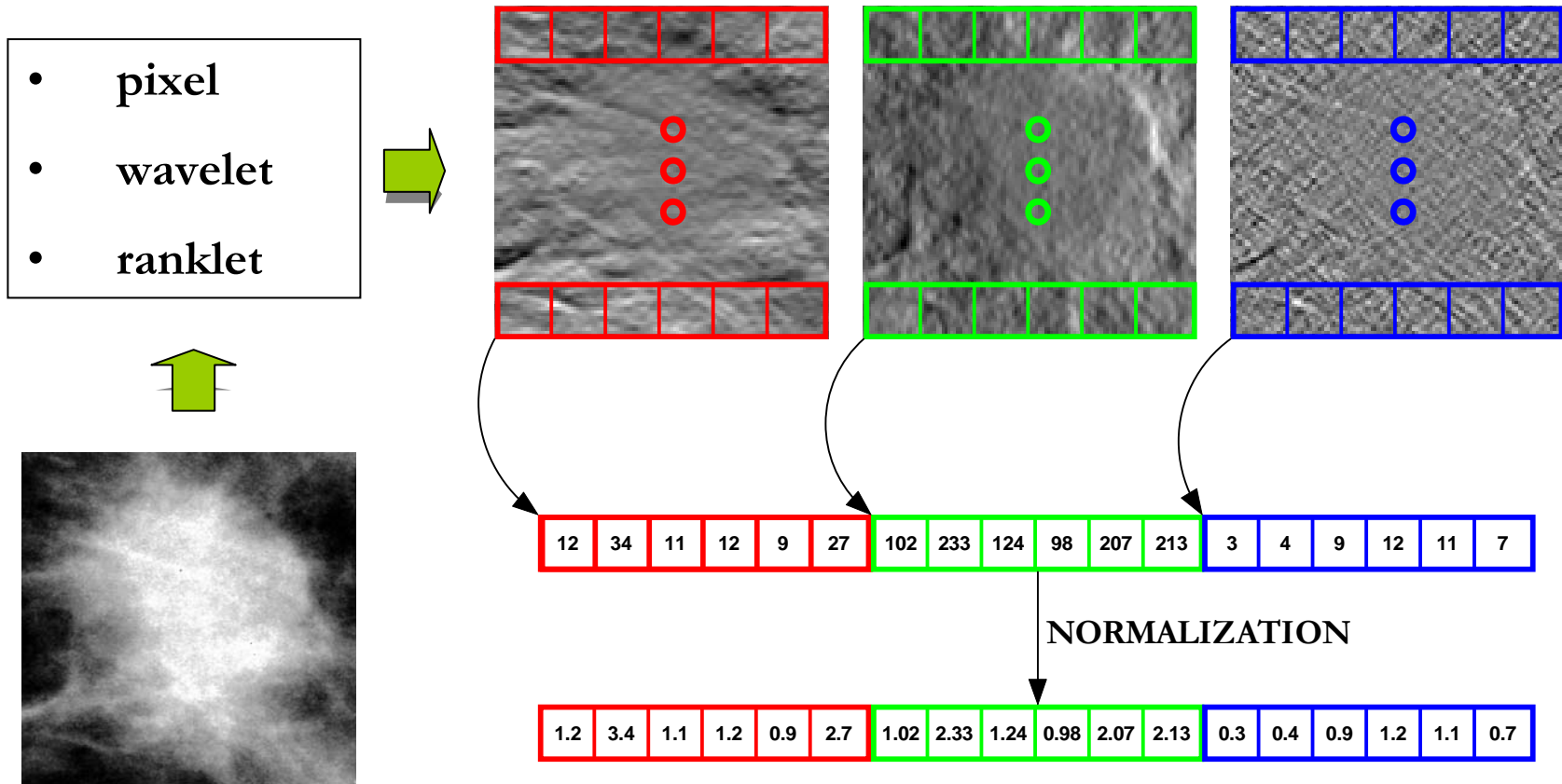
Computational overview



- Red blocks produce about 80% of computational cost.
- How can we improve the performance?

Efficient exploitation of memory/CPU bus!!!

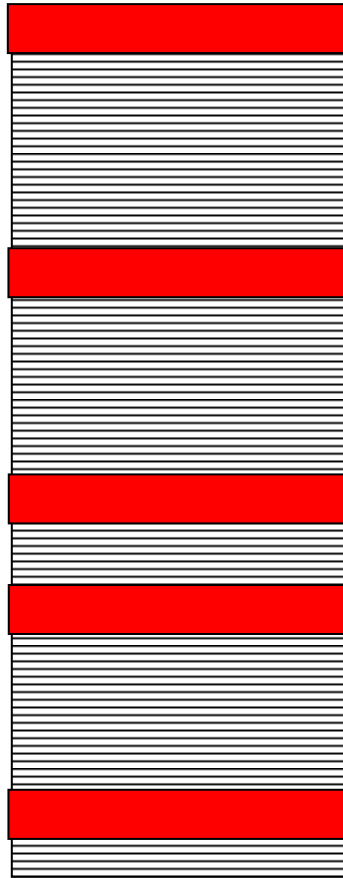
Data representation



This vector identifies a point in a ***n*-dimensional space** ($n \sim 4000$)
Each element of the vector is a **feature**

Memory access

Not efficient



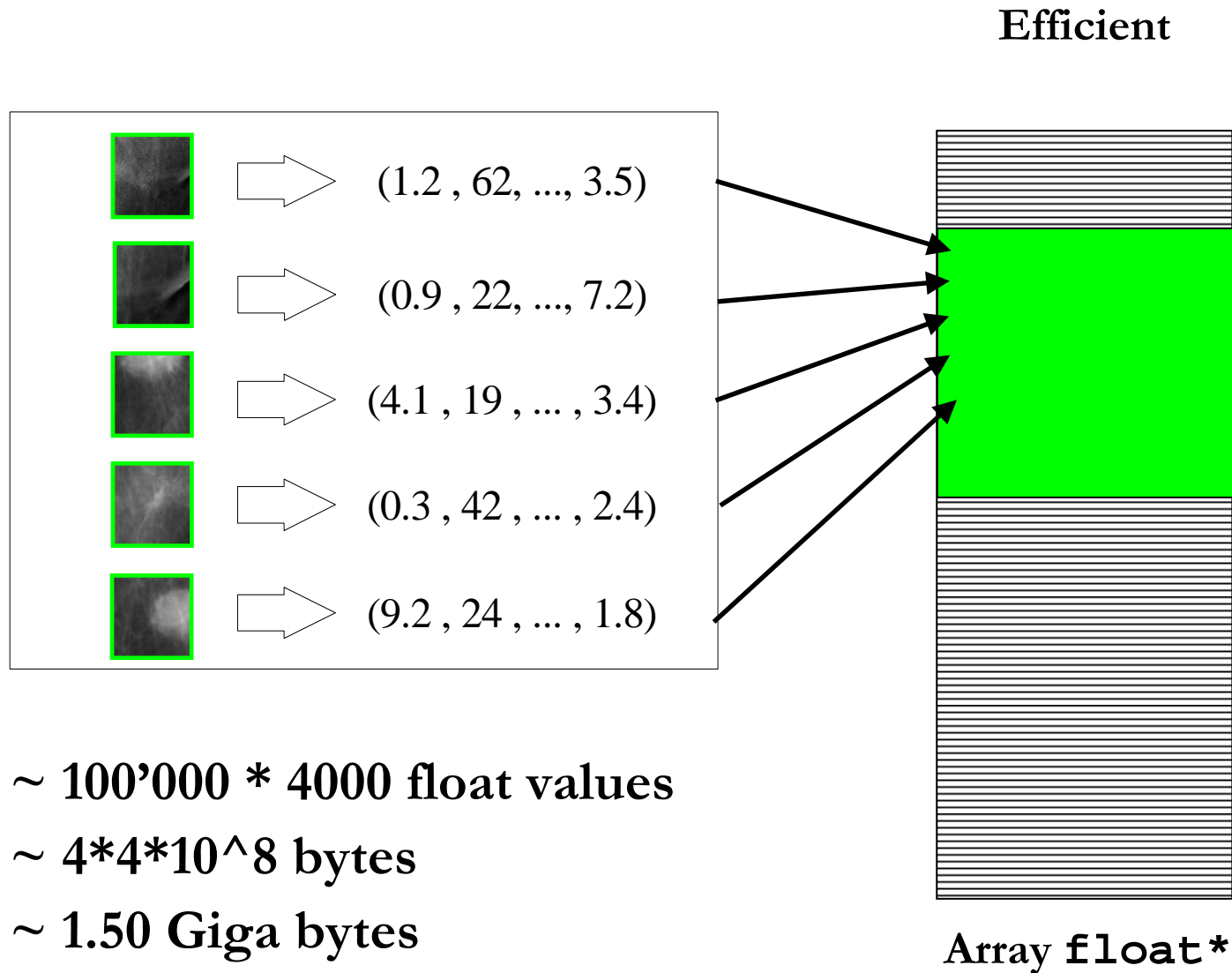
Matrix `float**`

Efficient



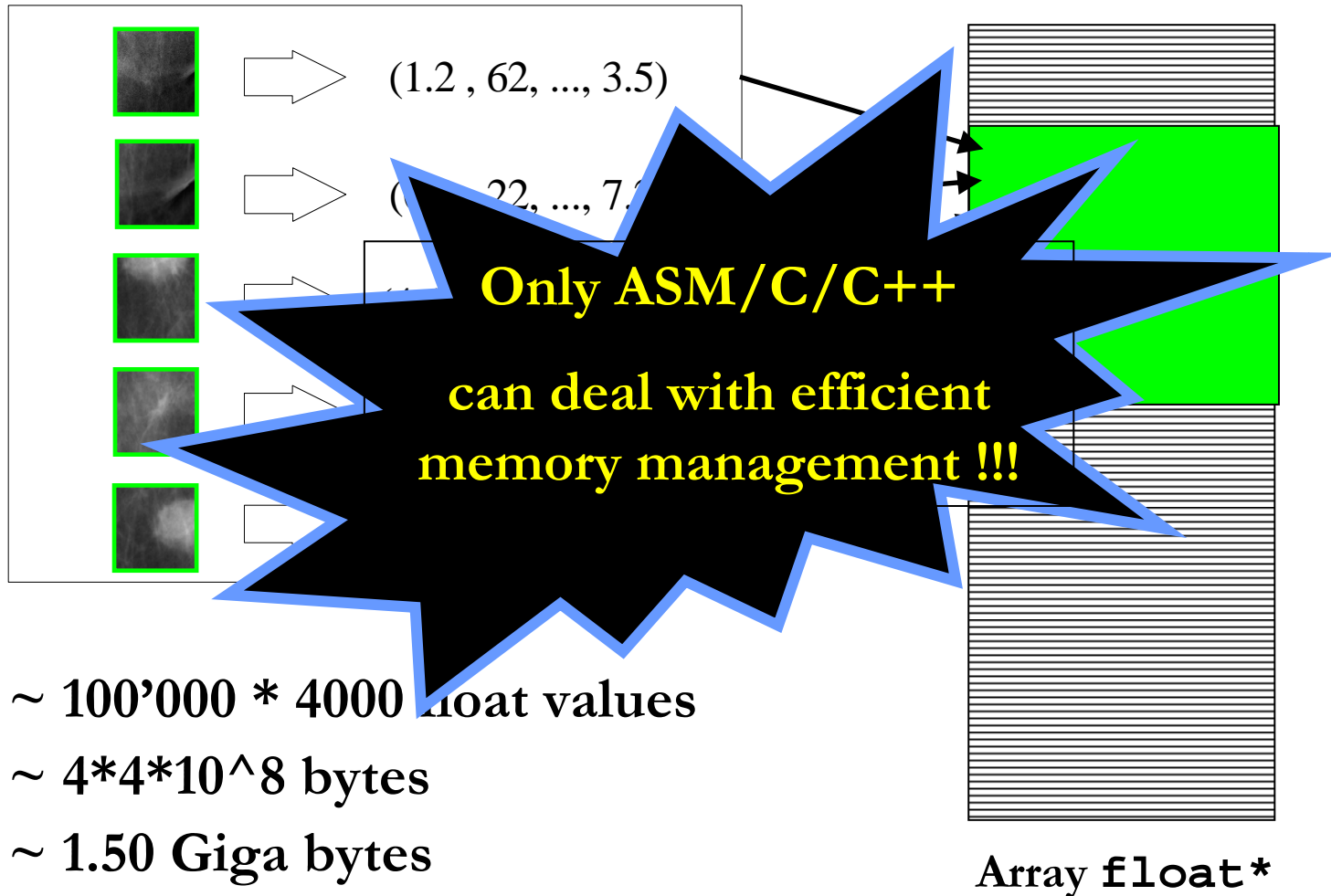
Array `float*`

Data representation storage



Data representation storage

Efficient

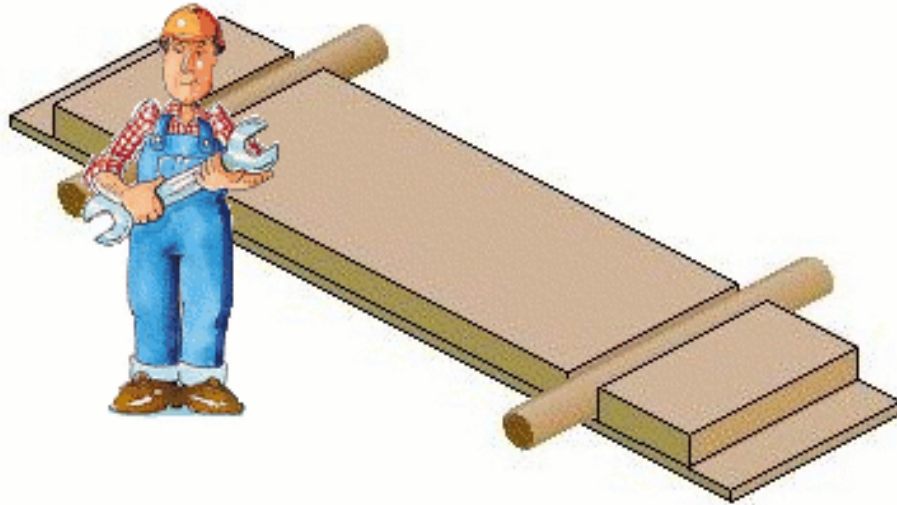


Computational Optimizations: Do it faster!

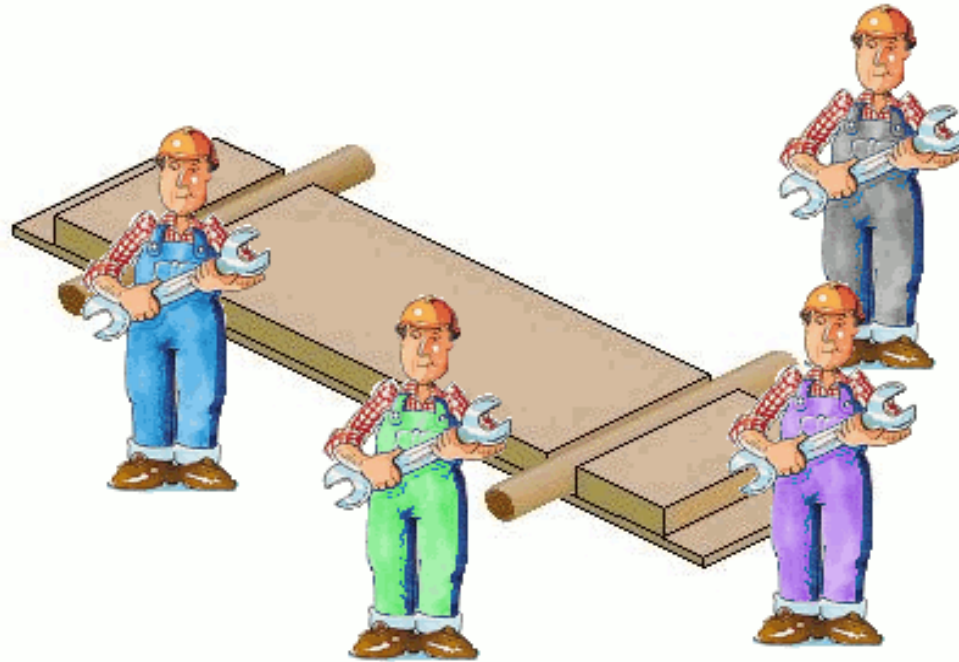
Overview

1. Hardware taxonomy
2. C programming language
3. The matrix-way of thinking about programs
4. BLAS
5. ATLAS
6. SIMD
7. GPU
8. IBM CELL BE

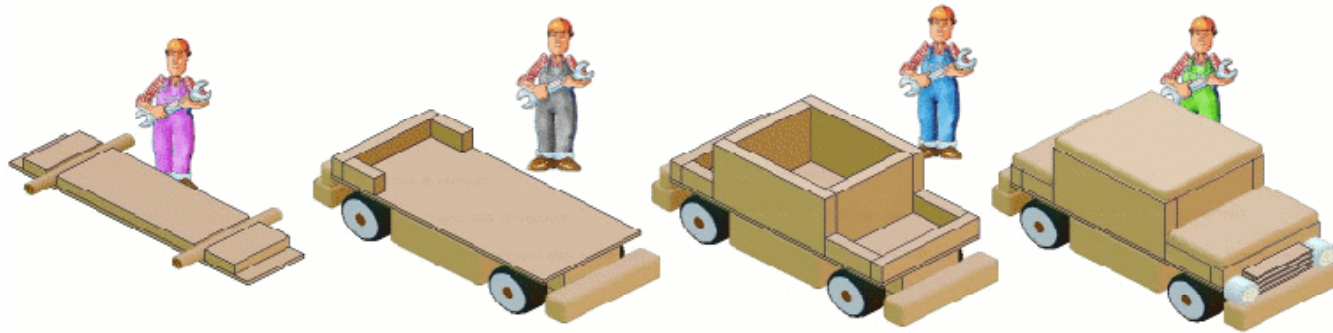
SISD CPU (Standard)



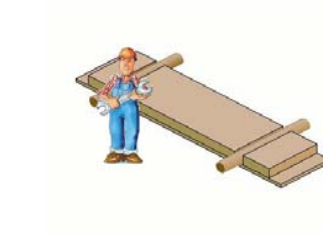
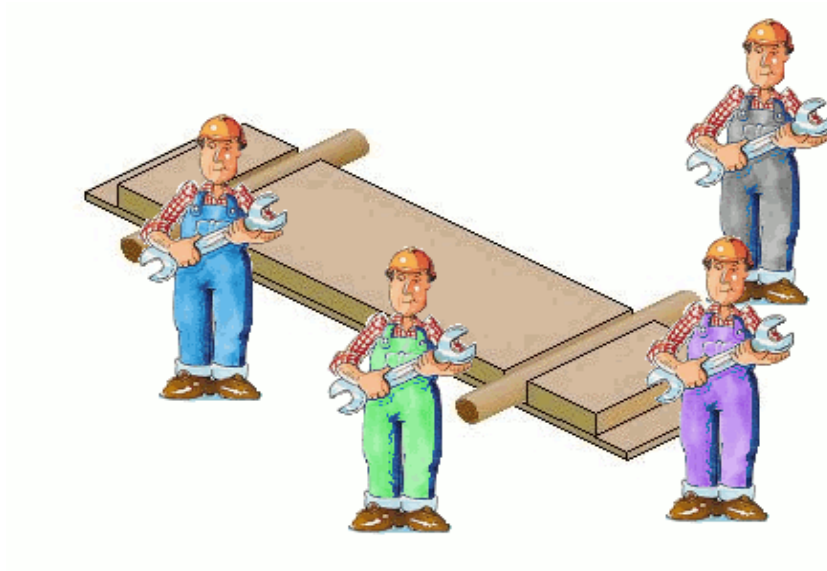
SIMD CPU (MMX,SSE,...)



STREAM CPU (GPU)



HYBRIB CPU (CELL BE)



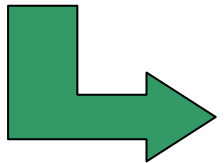
The C supremacy

1. Efficient code
2. Efficient compiler
3. Efficient linker
4. ANSI C is portable: Linux, Windows, etc
5. ANSI C will be always available (*e.g.* VB6 is dead!)
6. **ALL** O.S. are written in C
7. C can produce .SO/.DLL to be called by others
8. C is simple for algorithm implementation
9. By using C you can access SIMD instructions

An example of optimization target

By using Support Vector Machine or Relevance Vector Machine a new image \mathbf{x} is assigned to class +1 or -1 according to the following function:

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{SV} (y_i \alpha_i^0 K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}) + b^0) \right)$$



Fast and easily parallelizable

The standard iterative view

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{SV} (y_i \alpha_i^0 K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}) + b^0) \right)$$

```
for (i = 0; i < sv_num; i++)  
{  
    buffer = p_scalar(&sv[(i * features_num)], example, features_num);  
    dist += buffer * alfa[i];  
}  
  
dist -= b;  
  
return dist;
```

The matrix view

```

for (i = 0; i < sv_num; i++)
{
    buffer = p_scalar(&sv[(i * features_num)], example, features_num);

    dist += buffer * alfa[i];
}

dist -= b;

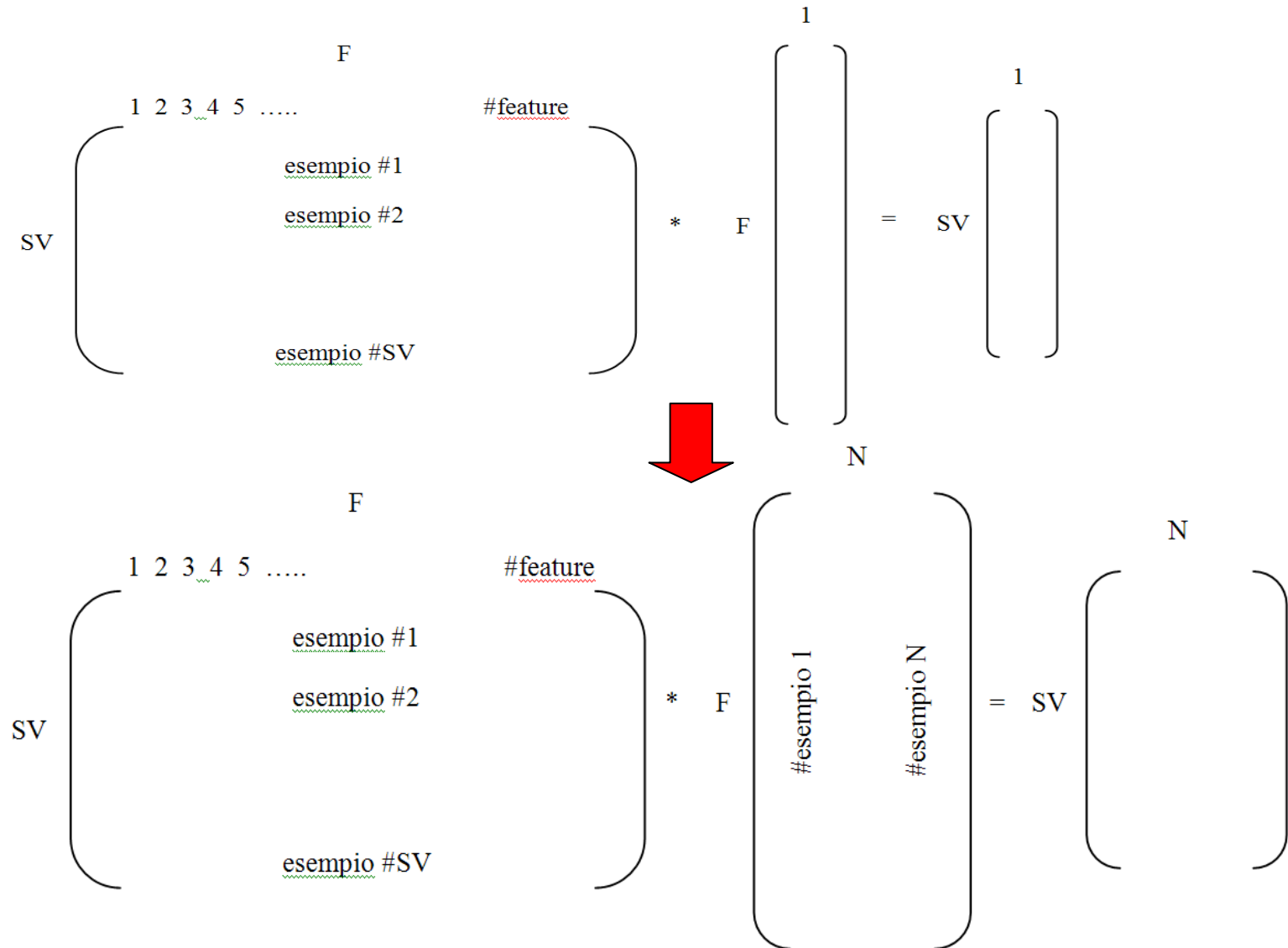
return dist;

```

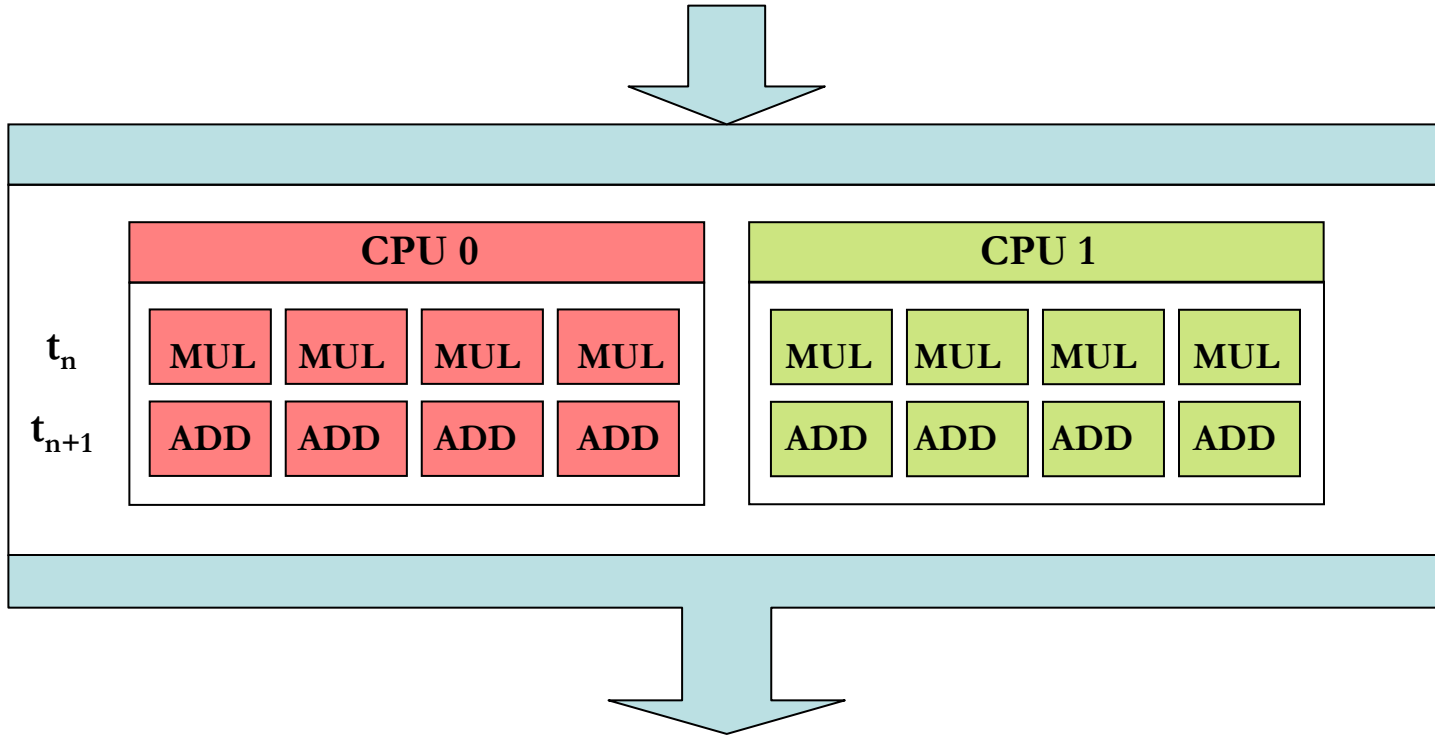
$$\begin{array}{c} \text{SV} \end{array} \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & \dots \\ \text{esempio \#1} \\ \text{esempio \#2} \\ \vdots \\ \text{esempio \#SV} \end{pmatrix} \begin{array}{c} F \\ \text{\#feature} \end{array} * \begin{array}{c} F \\ \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \end{array} = \begin{array}{c} \text{SV} \\ \begin{pmatrix} 1 \\ \vdots \\ \text{dist} \end{pmatrix} \end{array}$$

$$\begin{array}{c} 1 \end{array} \begin{array}{c} \text{SV} \\ \begin{pmatrix} \text{alfa1} & \text{alfa2} & \dots & \text{alfaSV} \end{pmatrix} \end{array} * \begin{array}{c} \text{SV} \\ \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \end{array} = \begin{array}{c} 1 \\ \begin{pmatrix} \text{dist} \end{pmatrix} \end{array}$$

The matrix view - global



Optimizing Dot product



1 CPU no SWAR

$\sim 3000 \times 1500 \times 10^5 \times 2$ ops

~ 1000 Giga ops =

~ 5 minutes on 3 GHz CPU

Number of SV ~ 1500

Number of features ~ 3000

Number of samples ~ 100000

2 CPUs with SWAR

$\sim 3000 \times 1500 \times 10^5 \times 2$ ops / 2 / 4

~ 125 Giga ops =

~ 0.5 minutes on 3 GHz CPUs

Assembler built-in

v4sf a,b,c,d,e,f,g,h,m,n;

[...]

a=__builtin_ia32_loadups(va);
b=__builtin_ia32_loadups(vb);

data loading

[...]

a=__builtin_ia32_mulps(a,b);

*Superscalar
multiplication*

[...]

__builtin_ia32_storess (&b2, a);

data store

Assembler hand-made

p_scalar:

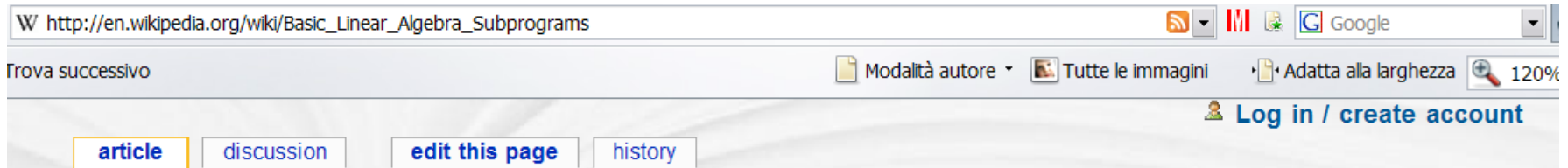
```
movaps (%eax), %xmm2
movaps 32(%eax), %xmm3
movaps (%edx), %xmm6
movaps 32(%edx), %xmm7
movaps 16(%eax), %xmm1
movaps 16(%edx), %xmm5
movaps 48(%edx), %xmm0
movaps 48(%eax), %xmm4
```

```
mulps %xmm6, %xmm2
mulps %xmm5, %xmm1
mulps %xmm4, %xmm0
```

```
mulps %xmm7, %xmm3
addps %xmm1, %xmm2
addps %xmm3, %xmm0
subl $16, %ecx
addps -88(%ebp), %xmm2
addps -104(%ebp), %xmm0
addl $64, %eax
addl $64, %edx
testl %ecx, %ecx
```

```
movaps %xmm2, -88(%ebp)
movaps %xmm0, -104(%ebp)
jg .L12
```

BLAS



Basic Linear Algebra Subprograms

From Wikipedia, the free encyclopedia

Basic Linear Algebra Subprograms (BLAS) is a de facto [application programming interface](#) standard for publishing libraries to perform basic [linear algebra](#) operations such as [vector](#) and [matrix multiplication](#). They were first published in 1979, and are used to build larger packages such as [LAPACK](#). Heavily used in [high-performance computing](#), highly optimized implementations of the BLAS interface have been developed by hardware vendors such as by [Intel](#) as well as by other authors (e.g. [ATLAS](#) is a portable self-optimizing BLAS). The [LINPACK](#) benchmark relies heavily on [DGEMM](#), a BLAS subroutine, for its performance.

The BLAS functionality is divided into three levels: 1, 2 and 3.

Level 1

[edit]

This level contains *vector operations* of the form

$$\mathbf{y} \leftarrow \alpha \mathbf{x} + \mathbf{y}$$

as well as scalar [dot products](#) and [vector norms](#), among other things.

Level 2

[edit]

This level contains *matrix-vector operations* of the form

$$\mathbf{y} \leftarrow \alpha A \mathbf{x} + \beta \mathbf{y}$$

as well as solving $T\mathbf{x} = \mathbf{y}$ for \mathbf{x} with T being triangular, among other things.

Level 3

[edit]

This level contains *matrix-matrix operations* of the form

$$C \leftarrow \alpha AB + \beta C$$

as well as solving $B \leftarrow \alpha T^{-1}B$ for triangular matrices T , among other things. This level contains the widely used [General Matrix Multiply](#) operation.

ATLAS

ATLAS (**A**utomatically**T**uned**L**inear**A**lgebra **S**oftware)

API BLAS (**B**asic **L**inear**A**lgebra **S**ubroutine)

<http://math-atlas.sourceforge.net/>

```
(void)catlas_sset(ra*rb,1,c,1);
```

Memory set

[...]

```
(void)cblas_sgemm(CblasRowMajor, CblasNoTrans, CblasTrans,  
[...],coef_lin, coef_const, [...]);
```

Matrix Multiply

[...]

```
(void)cblas_scopy(ra*rb,c,1,tmp,1);
```

Memory copy

```
for(i=0;i<rb;i++)
```

Scalar multiply

```
(void)cblas_sscal(ra,alfa[i],c+i,rb);
```

```
for(i=0;i<ra;i++)
```

Dot multiply

```
result[i]=cblas_sdot(rb,c+i*rb,1,tmp+i*rb,1)-threshold;
```

ATLAS download page

Browser address bar: http://sourceforge.net/project/showfiles.php?group_id=23725

Google News | Trova successivo | Modalità autore | Tutte le immagini | Adatta alla larghezza | 120%

SOURCEFORGE.NET [Log in](#) | [Create account](#) | [Help](#)

Home | **Browse Software** | Marketplace NEW | Community | Create Project | Jobs

Software [Advanced](#)

Algorithm?
Need an Algorithm? ScienceOps has answers.
www.ScienceOps.com

C++ code
C, C+, C# Developer Resources. News, Tips, Tools, and More.
www.DevSource.com

Real-Time Java that Works
Register for free white papers, product demos & evaluation software
www.JaveLocity.com

Ads by Google



SF.net » Projects » Automatically Tuned Linear Algebra Soft. » **Files**

Automatically Tuned Linear Algebra Soft.

[Project Web Site](#)

[Advanced](#)

- Enter Here to Research Featured Solutions -

Ads by Google

MATLAB Numerical Analysis
Perform numerical analysis with interactive tools in MATLAB.
www.mathworks.com

20X Faster C/C++ Builds
Enterprise-Class Tools to Automate C/C++ Builds. Free Whitepapers.
Electric-Cloud.com





Linux Downloads
Free Guide to Open Source Software in Government
www.mysql.com

About Automatically Tuned Linear Algebra Soft.

ATLAS (Automatically Tuned Linear Algebra Software) provides highly optimized Linear Algebra kernels for arbitrary cache-based architectures. ATLAS provides ANSI C and Fortran77 interfaces for the entire BLAS API, and a small portion of the LAPACK AP

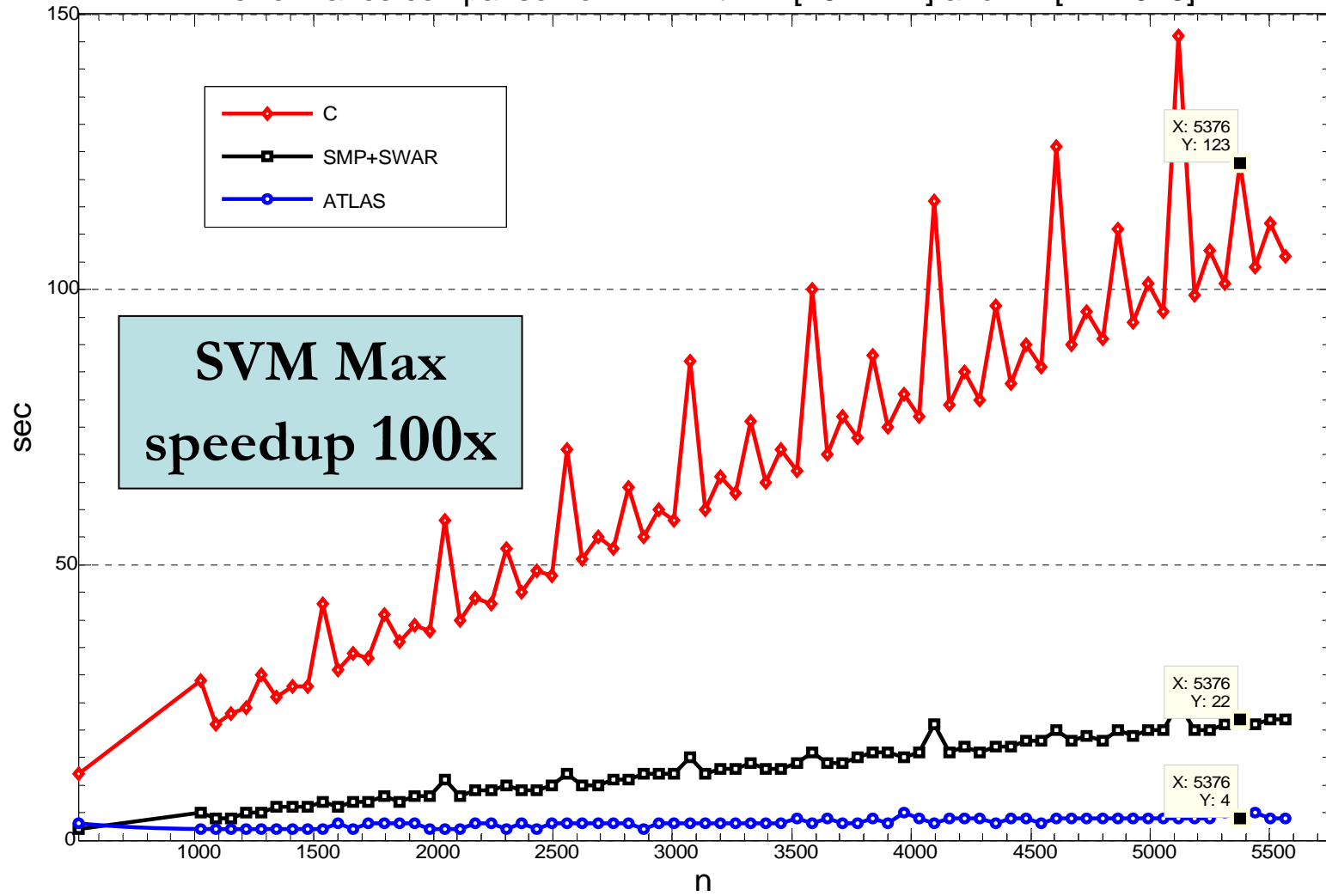
[Imagine](#)

Latest File Releases

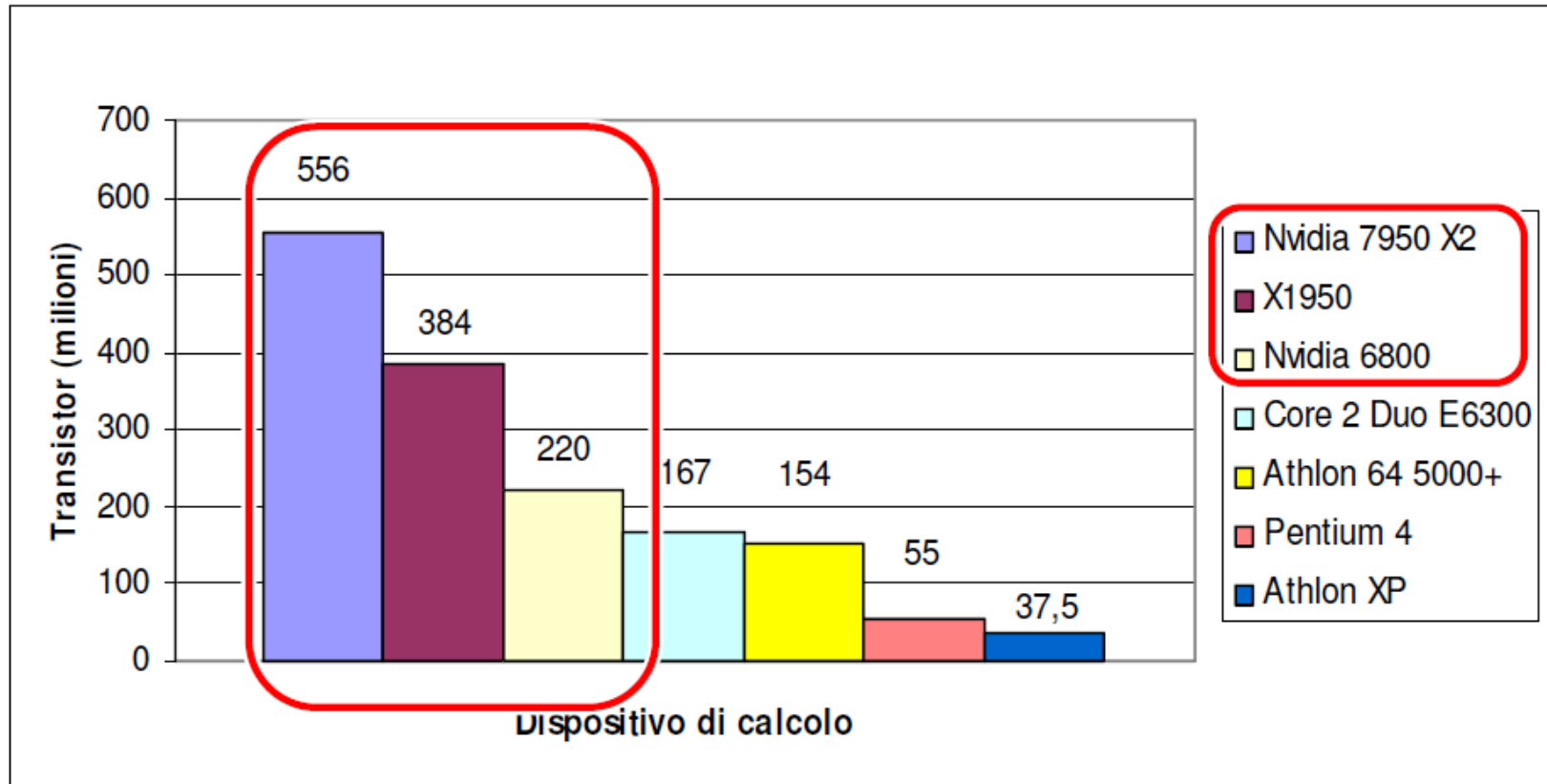
| Package | Release | Date | Notes / Monitor | Downloads |
|------------------------|---------|-------------------|---|--------------------------|
| Stable | 3.8.1 | February 21, 2008 |  -  | Download |
| Tester | 1.1.3 | January 8, 2007 |  -  | Download |

Performance comparison

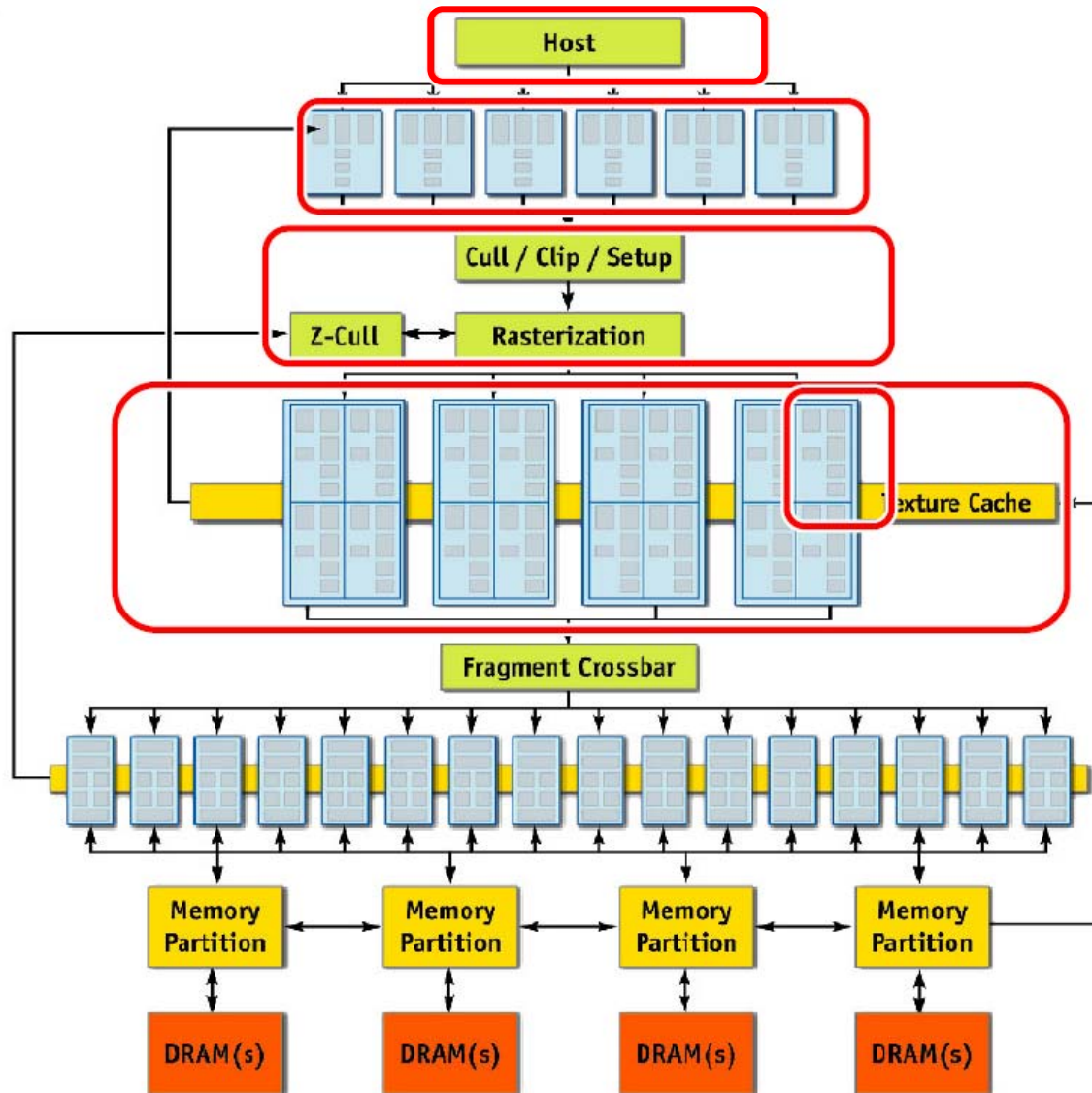
Performance comparison on $A \times B$ with $A=[1024 \times n]$ and $B=[n \times 2048]$



The future computing devices



GPU architecture



GP-GPU “raw” programming



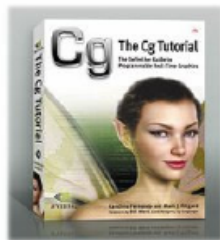
GCC 4.1.1
OpenGL

```
glGenFramebuffersEXT(1, &fb_id);  
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT,  
                      fb_id);  
glEnable (GL_TEXTURE_RECTANGLE_ARB);
```



Pentium 4 a 3.00GHz
1024 Kb di cache

NVIDIA
Cg Compiler
Release 1.4



```
void matrixm(  
    uniform samplerRECT image,  
    float2 texcoord : TEXCOORD0,  
    out float4 color0 : COLOR0,  
    out float4 color1 : COLOR1)
```



NVIDIA
7800 GT
256Mb di RAM

GP-GPU CUDA programming

http://www.nvidia.com/object/cuda_what_is.html

Trova nella pagina Trova successivo Modalità autore Tutte le immagini Adatta alla larghezza 120%

NVIDIA **CUDA ZONE** USA - United States Search NVIDIA

DOWNLOADS WHAT IS CUDA CUDA U DEVELOPING WITH CUDA FORUMS NEWS AND EVENTS

What is CUDA

- CUDA Applications
- Learn More
- Universities Teaching CUDA
- Sign up for CUDA Alerts

What is CUDA

NVIDIA® CUDA™ is a general purpose parallel computing architecture that leverages the parallel compute engine in NVIDIA graphics processing units (GPUs) to solve many complex computational problems in a fraction of the time required on a CPU. It includes the CUDA Instruction Set Architecture (ISA) and the parallel compute engine in the GPU. To program to the CUDATM architecture, developers can, today, use C, one of the most widely used high-level programming languages, which can then be run at great performance on a CUDATM enabled processor. Other languages will be supported in the future, including FORTRAN and C++.

With over 100 million CUDA-enabled GPUs sold to date, thousands of software developers are already using the free CUDA software development tools to solve problems in a variety of professional and home applications – from video and audio processing and physics simulations, to oil and gas exploration, product design, medical imaging, and scientific research. With over 100 million CUDA-capable GPUs already deployed, thousands of software programmers are already using the free CUDA software tools to accelerate applications


TECHNOLOGY FEATURES

- Standard C language for parallel application development on the GPU
- Standard numerical libraries for FFT (Fast Fourier Transform) and BLAS (Basic Linear Algebra Subroutines)
- Dedicated CUDA driver for computing with fast data transfer path between GPU and CPU
- CUDA driver interoperates with OpenGL and DirectX graphics drivers
- Support for Linux 32/64-bit and Windows XP 32/64-bit operating systems

WHITE PAPERS AND ARTICLES

Accelerating MATLAB with CUDA using MEX Files

IBM Cell BE



WIKIPEDIA
The Free Encyclopedia

navigation

- [Main page](#)
- [Contents](#)
- [Featured content](#)
- [Current events](#)
- [Random article](#)

search

interaction

- [About Wikipedia](#)
- [Community portal](#)
- [Recent changes](#)
- [Contact Wikipedia](#)
- [Donate to Wikipedia](#)
- [Help](#)

W http://en.wikipedia.org/wiki/Cell_(microprocessor)

Trova nella pagina Trova successivo

Modalità autore Tutte le immagini Adatta alla larghezza 150%

[Log in / create account](#)

[article](#) [discussion](#) [edit this page](#) [history](#)

Cell (microprocessor)

From Wikipedia, the free encyclopedia

Cell is a [microprocessor](#) architecture jointly developed by [Sony Computer Entertainment](#), [Toshiba](#), and [IBM](#), an alliance known as "STI". The architectural design and first implementation were carried out at the STI Design Center in [Austin, Texas](#) over a four-year period beginning March 2001 on a budget reported by IBM as approaching US\$400 million.^[1] Cell is shorthand for **Cell Broadband Engine Architecture**, commonly abbreviated *CBEA* in full or *Cell BE* in part. Cell combines a general-purpose [Power Architecture core](#) of modest performance with streamlined [coprocessing](#) elements^[2] which greatly accelerate [multimedia](#) and [vector processing](#) applications, as well as many other forms of dedicated computation.^[2]

The first major commercial application of Cell was in Sony's [PlayStation 3 game console](#). [Mercury Computer Systems](#) has a dual Cell server, a dual Cell [blade](#) configuration, a rugged computer, and a PCI Express accelerator board available in different stages of production. Toshiba has announced plans to incorporate Cell in [high definition](#) television sets. Exotic features such as the [XDR](#) memory subsystem and coherent [Element Interconnect Bus](#) (EIB) interconnect^[3] appear to position Cell for future applications in the [supercomputing](#) space to exploit the Cell processor's prowess in [floating point](#) kernels. IBM has announced plans to incorporate Cell processors as

IBM Cell BE HW

