

Università di Bologna
CdL in Informatica
Analisi delle immagini
AA 2007/2008

Computational Optimizations & Wavelet

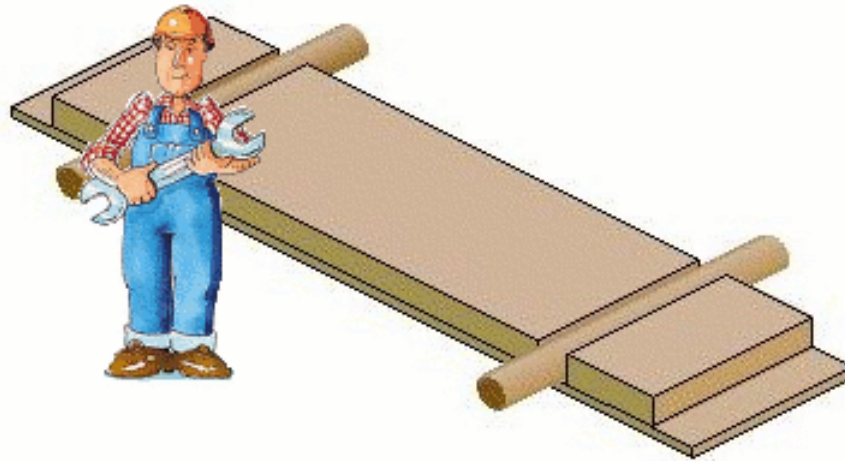
Lesson 2
10 April 2007

Matteo Roffilli

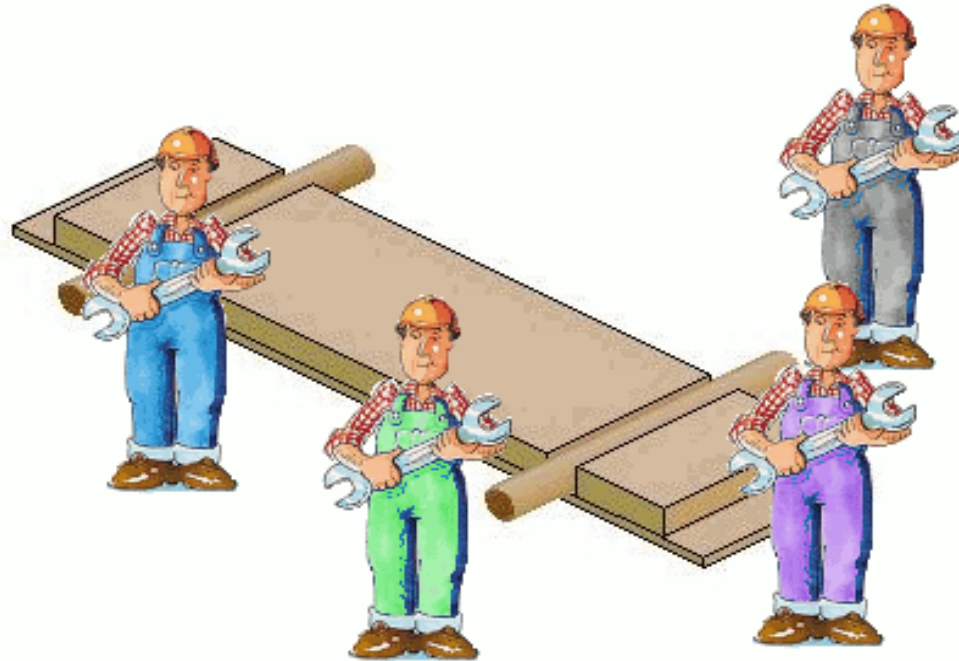
roffilli@csr.unibo.it
<http://www.cs.unibo.it/~roffilli>

Computational Optimizations: Do it faster!

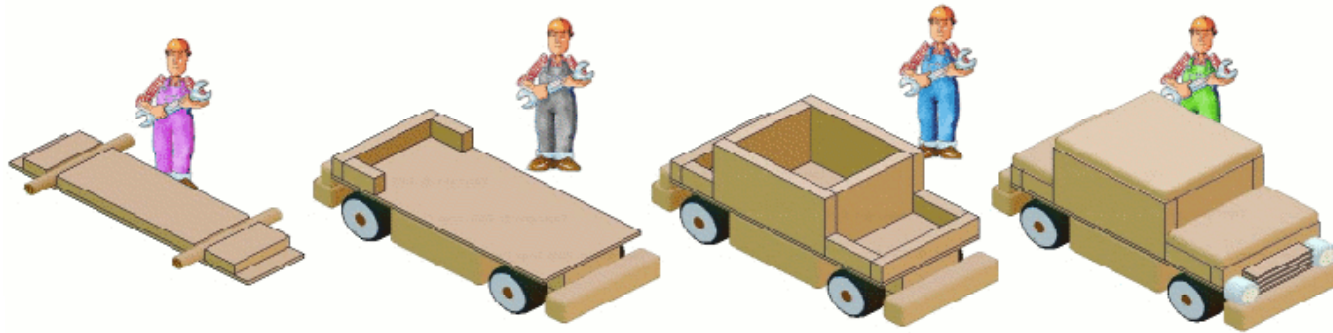
SISD CPU (Standard)



SIMD CPU (MMX,SSE,...)



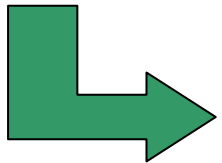
STREAM CPU (GPU)



Optimization target

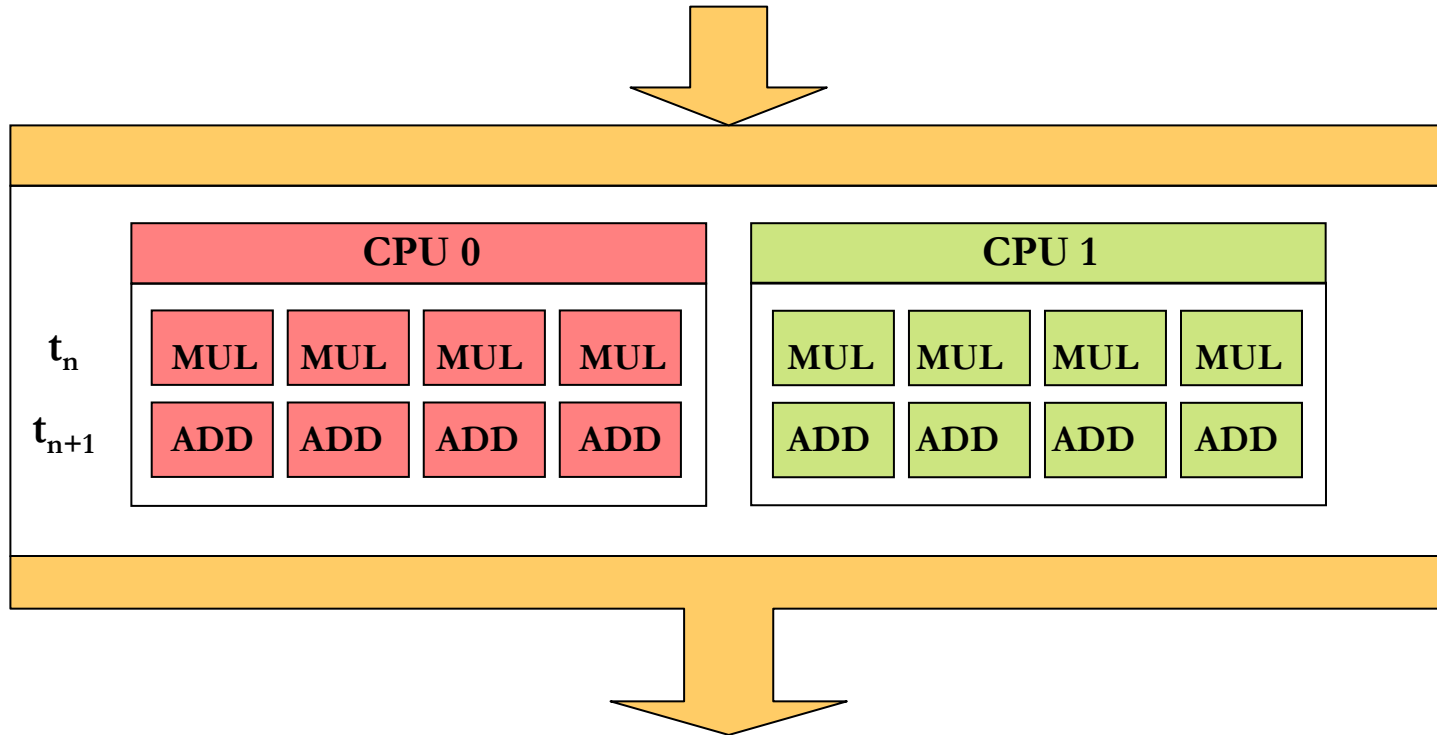
By using Support Vector Machine or Relevance Vector Machine a new image \mathbf{x} is assigned to class +1 or -1 according to the following function:

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



Fast and easily parallelizable

Dot product by SWAR and SMP



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
```

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

Navigation: [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 [Search](#) [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](#) [Tracker](#) [Mailing Lists](#) [Forums](#) [Code](#) [Services](#) [Download](#) [Tasks](#) [Project Web Site](#)

[Donate](#) [Stats](#) [RSS](#)

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

- Enter Here to Research Featured Solutions -

Ads by Google

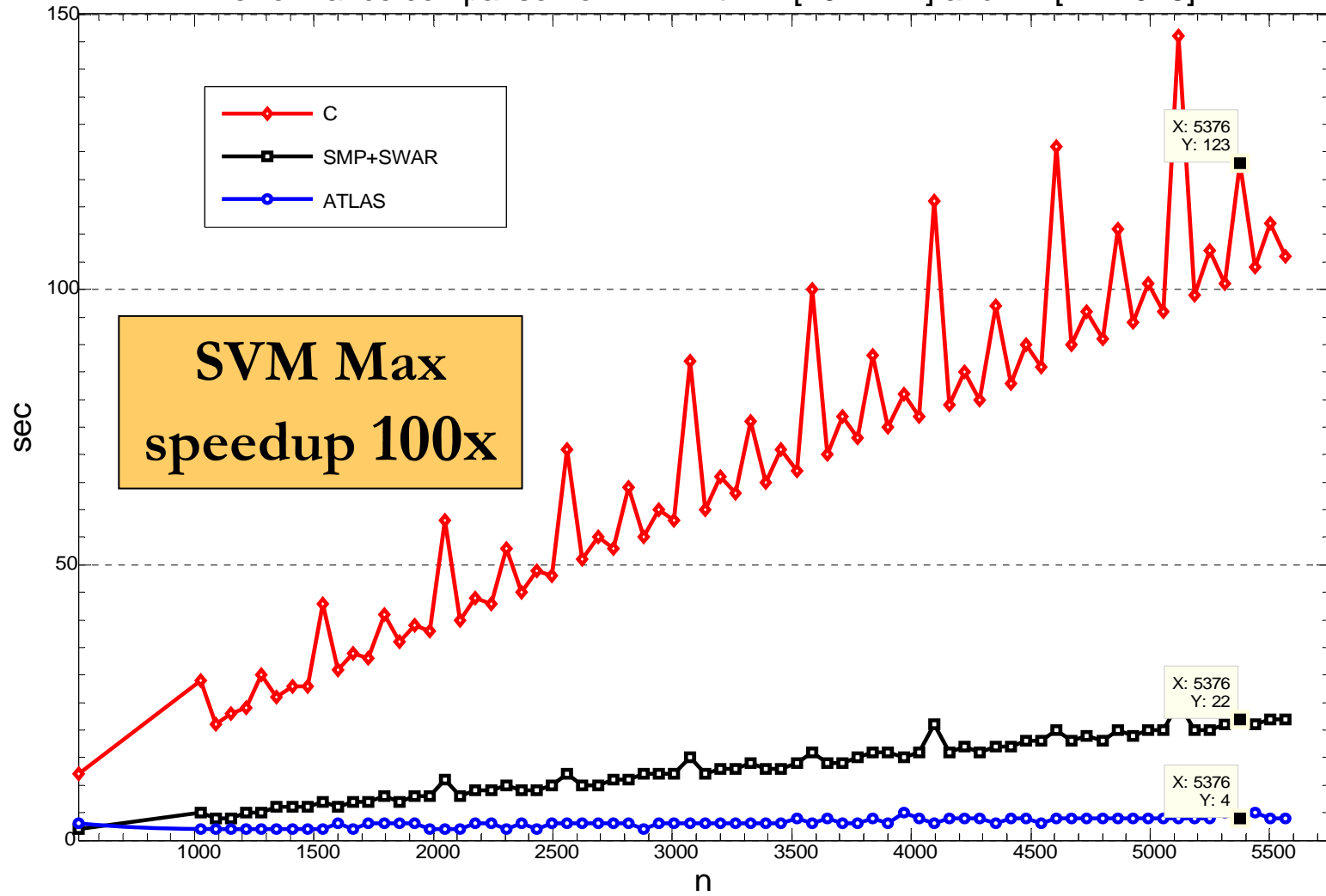
MATLAB Numerical Analysis
Perform numerical analysis with interactive tools in MATLAB.
www.mathworks.com

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

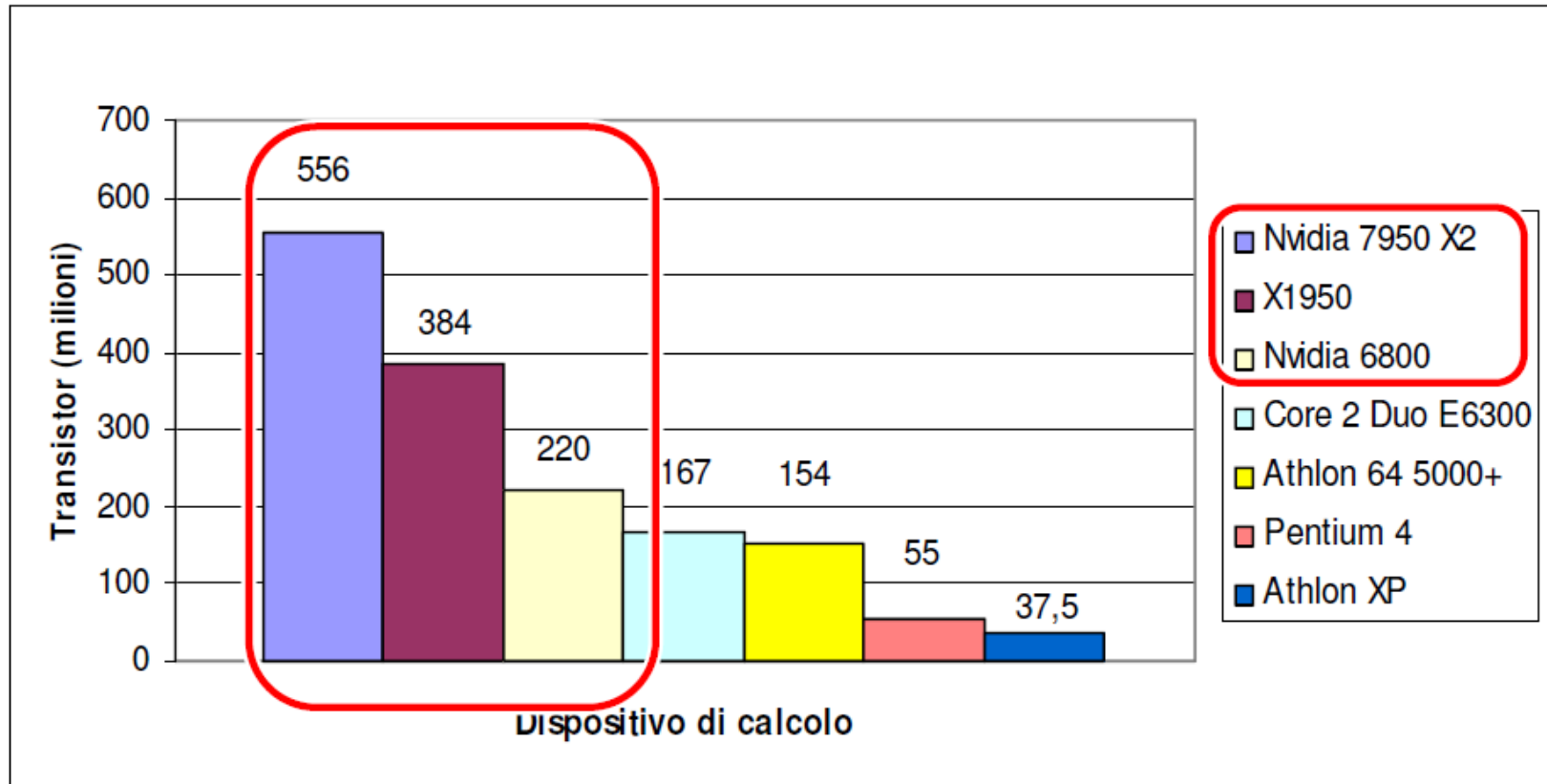
Linux Downloads
Free Guide to Open Source Software in Government
www.mysql.com

Performance comparison

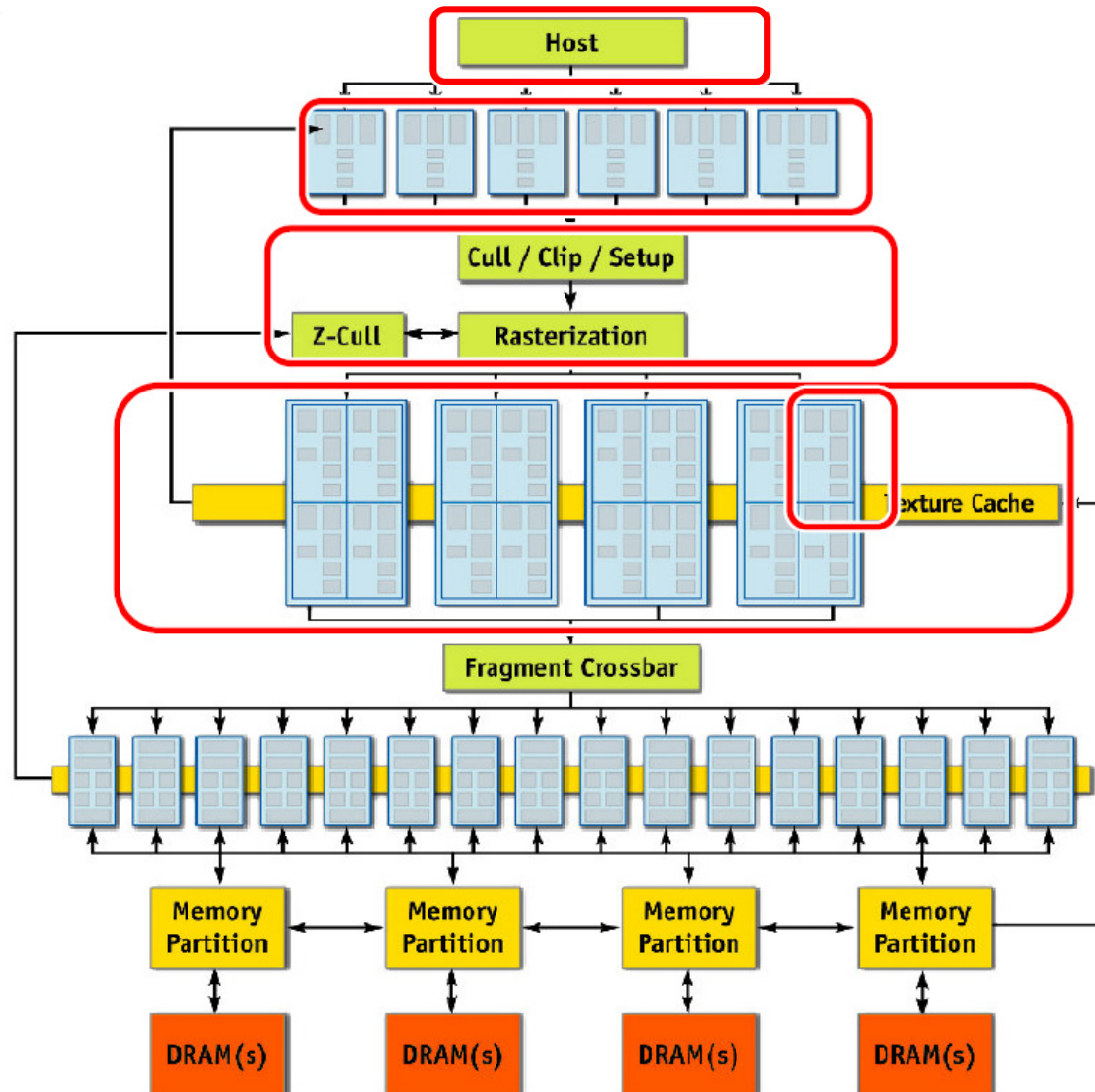
Performance comparison on $A \times B$ with $A=[1024 \times n]$ and $B=[n \times 2048]$



The future computing devices



GPU architecture



GPU programming



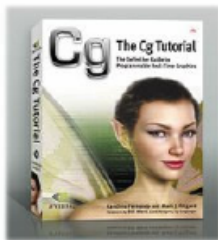
GCC 4.1.1
OpenGL

```
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

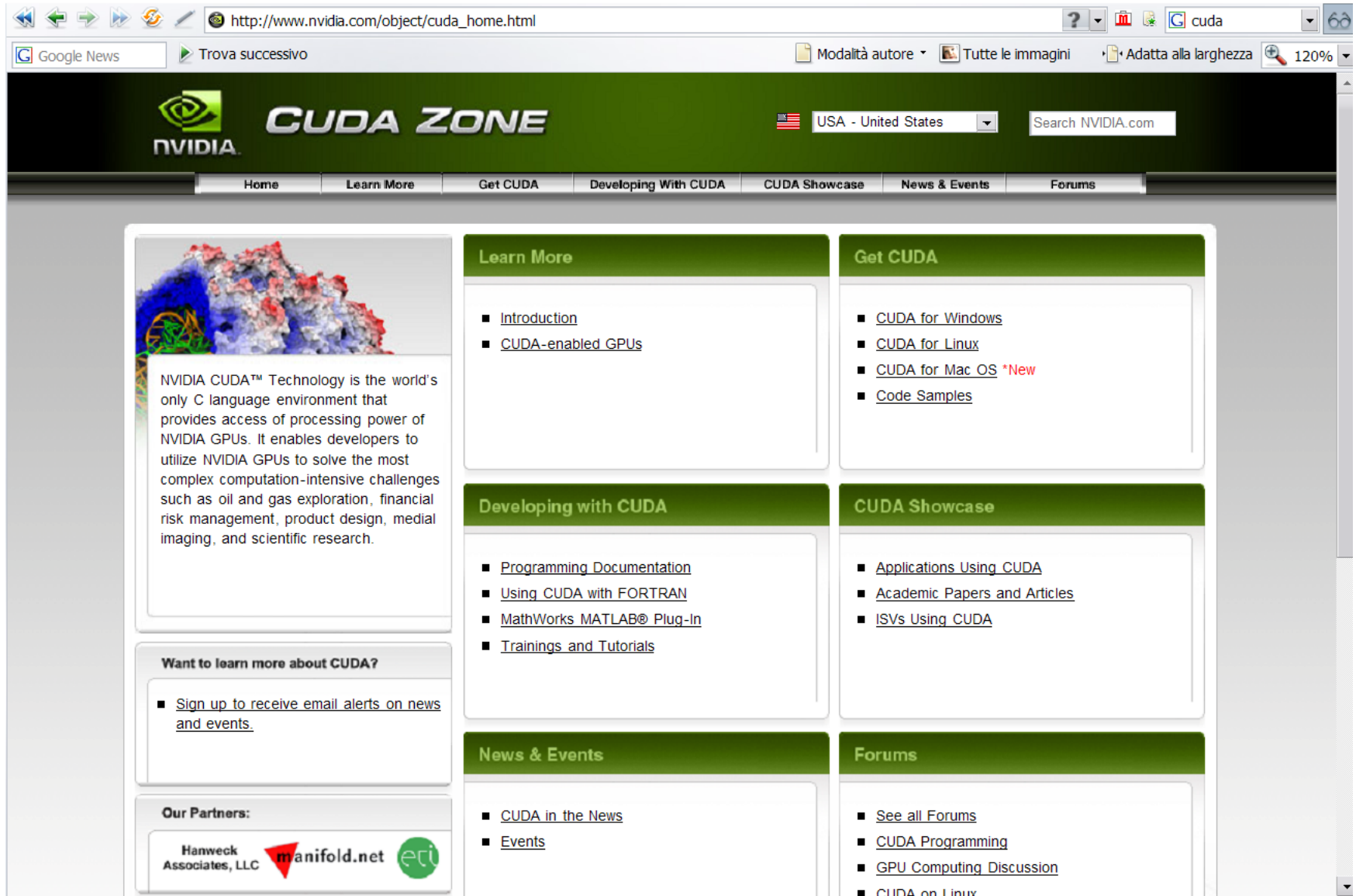


```
void matrixm(
    uniform samplerRECT image,
    float2 texcoord : TEXCOORD0,
    out float4 color0 : COLOR0,
    out float4 color1 : COLOR1)
```



NVIDIA
7800 GT
256Mb di RAM

GPU programming with CUDA



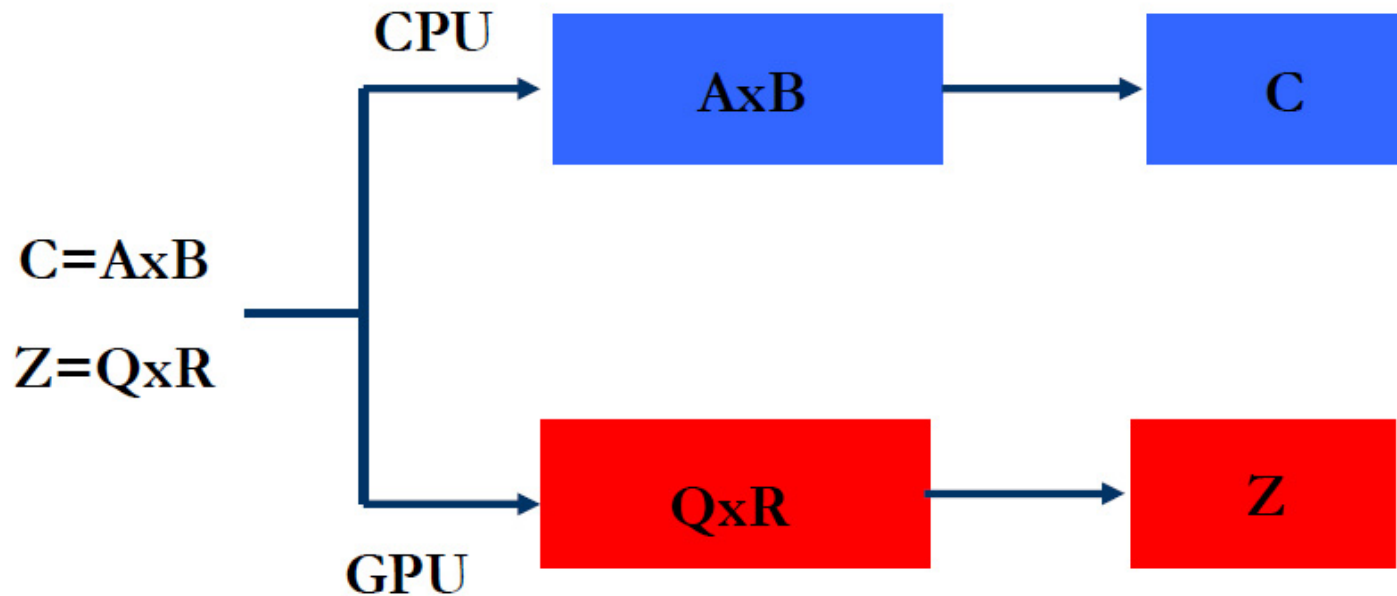
The screenshot shows the NVIDIA CUDA Zone website in a web browser. The browser's address bar displays `http://www.nvidia.com/object/cuda_home.html`. The page features a green header with the NVIDIA logo and the text "CUDA ZONE". A navigation menu includes links for Home, Learn More, Get CUDA, Developing With CUDA, CUDA Showcase, News & Events, and Forums. The main content area is divided into several sections:

- Learn More:**
 - [Introduction](#)
 - [CUDA-enabled GPUs](#)
- Get CUDA:**
 - [CUDA for Windows](#)
 - [CUDA for Linux](#)
 - [CUDA for Mac OS](#) *New
 - [Code Samples](#)
- Developing with CUDA:**
 - [Programming Documentation](#)
 - [Using CUDA with FORTRAN](#)
 - [MathWorks MATLAB® Plug-In](#)
 - [Trainings and Tutorials](#)
- CUDA Showcase:**
 - [Applications Using CUDA](#)
 - [Academic Papers and Articles](#)
 - [ISVs Using CUDA](#)
- News & Events:**
 - [CUDA in the News](#)
 - [Events](#)
- Forums:**
 - [See all Forums](#)
 - [CUDA Programming](#)
 - [GPU Computing Discussion](#)
 - [CUDA on Linux](#)

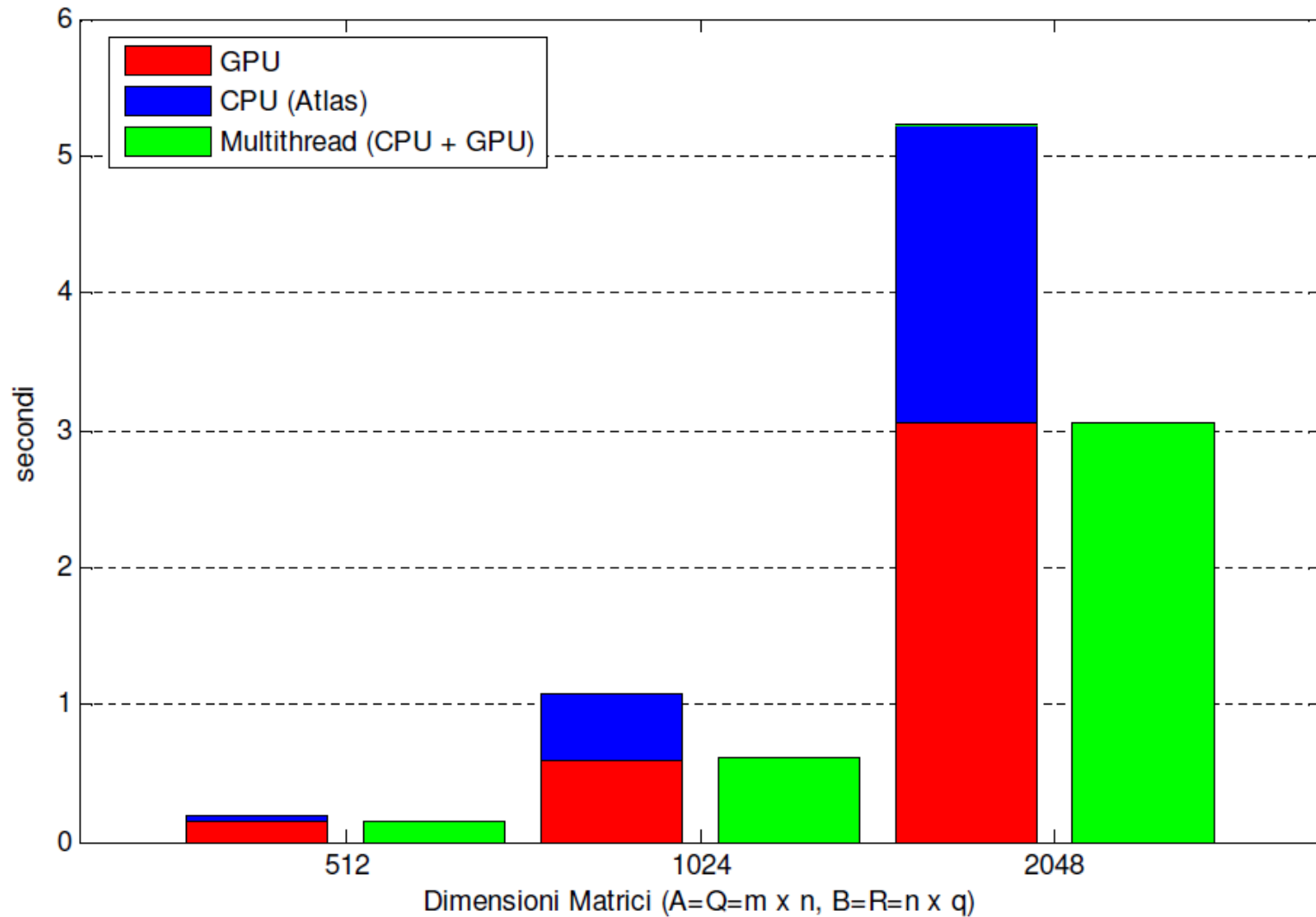
On the left side of the page, there is a large image of a molecular structure. Below it, a text box reads: "NVIDIA CUDA™ Technology is the world's only C language environment that provides access of processing power of NVIDIA GPUs. It enables developers to utilize NVIDIA GPUs to solve the most complex computation-intensive challenges such as oil and gas exploration, financial risk management, product design, medial imaging, and scientific research." Below this, a section titled "Want to learn more about CUDA?" contains a link: "■ [Sign up to receive email alerts on news and events.](#)". At the bottom left, the "Our Partners:" section lists "Hanweck Associates, LLC", "manifold.net", and "eti".

CPU-GPU mixed programming

1. The idea is to use jointly both the CPU and the GPU (as a math coprocessor)
2. GPU's floating number precision is the same of CPU's one
3. As example: matrix-matrix multiplication as below:



CPU-GPU results



Wavelet: an optimization approach

Haar wavelet

⇒ Haar wavelet transformation

It codifies differences in the intensity value of 2 adjacent pixels

It enhances structural variations of 3 main directions (O,V,D)



⇒ Overcomplete Haar wavelet

As the classical one with increased resolution

Classical vs. overcomplete

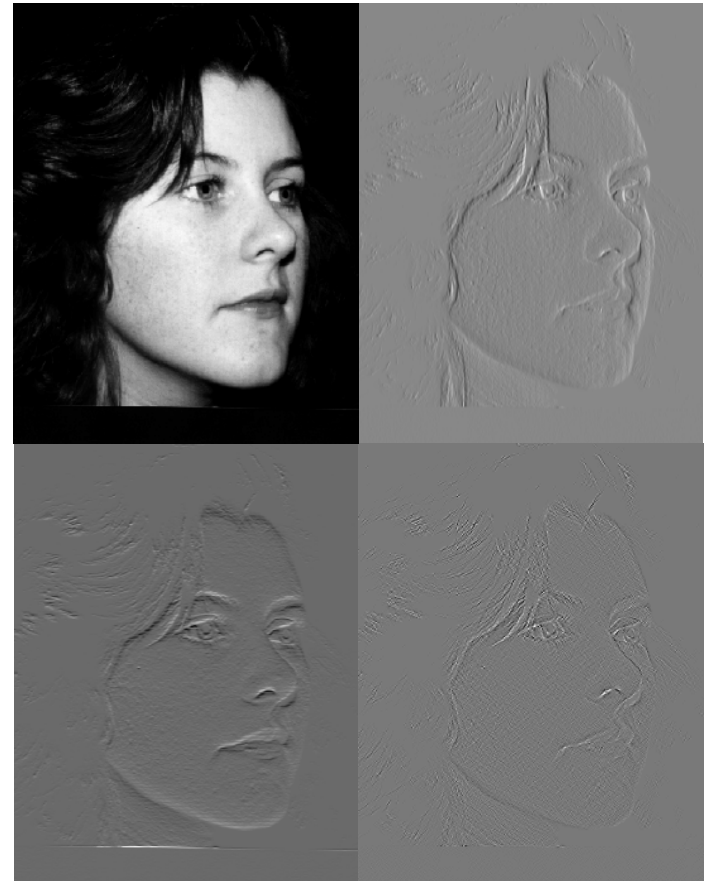
Original
image



Classical
Haar



Overcomplete
Haar



Algorithmic analysis

Classical Haar wavelet

- At each step $N \times N$ pixels $\rightarrow N \times N$ pixels
- Thus, we must allocate statically N^2 pixels
- At each step $\frac{3}{4} * N^2$ pixels are fixed
- So, time cost decreases linearly
- Terminating condition is well defined: 1 pixel left of user parameter
- Original image is INTEGER, resulting image is FLOAT
- If original image is not squared we must pad it

Algorithmic analysis

Overcomplete Haar wavelet

- At each step $N \times N$ pixels $\rightarrow > N \times N$ pixels
- Thus, we must allocate memory dynamically
- At each step $> \frac{3}{4} * N^2$ pixels are fixed
- So, time cost decreases **proportionally** to the level
- Terminating condition is well defined: 1 pixel left of user parameter
- Original image is INTEGER, resulting image is FLOAT
- If original image is not squared we must pad it

Padding

```
305 int pow2_wn(
306     MASS_BITMAP_FLOAT4* src,
307     MASS_BITMAP_FLOAT4* out
308 )
309 {
310     int k, h, max;
311     unsigned int i, j, n;
312     unsigned int width, height;
313     width=src->width; height=src->height;
314     /* Trova la più piccola potenza di due minore di max */
315     max=width>=height ? width : height;
316
317     for(i=0; ((1<<i)<max); i++);
318     n=1<<i;
319
320     /* Dimensione della maschera */
321     out->width=n;
322     out->height=n;
323     /* Allocazione */
324     if((out->image=(MASS_FLOAT4*)malloc(n*n*sizeof(MASS_FLOAT4)))==NULL)
325         ERROR_RETURN("haar_over.c: Error allocating memory for out", 0);
326     /* Estrazione dei pixel */
327     k=0; h=0;
328     for(i=0; i<height; i++)
329     {
330         for(j=0; j<width; j++)
331         {
332             out->image[k]=src->image[(i*src->width)+j];
333             k++; h++;
334         }
335         /* Completamento delle righe alla potenza di 2 */
336         for(j=width; j<n; j++)
337         {
338             out->image[k]=0;
339             k++;
340         }
341     }
342     /* Completa l'altezza alla potenza di 2 */
343     for(i=height; i<n; ++i)
344         for(j=0; j<n; ++j)
345             out->image[j+i*n]=0;
346     return 1;
347 }
```


2D Haar

```
void DWT2D_HAAR(  
    MASS_FLOAT4* m,    /* Matrice da trasformare */  
    unsigned int n,    /* Dimensione della matrice */  
    unsigned int ln     /* Livello di decomposizione */  
)
```

```
1031  /* ciclo livelli */  
1032  for(k=0;k<ln;k++)  
1033  {  
1034      n2=n/2;  
1035  
1036      /* ciclo righe */  
1037      for(i=0;i<n;++i)  
1038      {  
1039          for(u=0;u<n2;u++)  
1040          {  
1041              offset=(i*width+2*u);  
1042  
1043              /* ciclo wavelet */  
1044              t[u+n2]=0.5F*(m[offset]-m[offset+1]);  
1045  
1046              /* ciclo scaling */  
1047              t[u]=t[u+n2]+m[offset+1];  
1048          }  
1049  
1050          /* copy processed vector */  
1051          memcpy((void*)(m+(i*width)),(void*)t,(size_t)(n*sizeof(MASS_FLOAT4)));  
1052      }  
1053  
1054      /* ciclo colonne */  
1055      for(j=0;j<n;++j)  
1056      {  
1057          for(u=0;u<n2;u++)  
1058          {  
1059              offset=(2*u*width+j);  
1060  
1061              /* ciclo wavelet */  
1062              t[u+n2]=0.5F*(m[offset]-m[offset+width]);  
1063  
1064              /* ciclo scaling */  
1065              t[u]=t[u+n2]+m[offset+width];  
1066          }  
1067  
1068          /* copy processed vector */  
1069          for(i=0;i<n;++i) m[i*width+j]=t[i];  
1070      }  
1071      /* go down to next level */  
1072      n/=2;  
1073  }
```

Extract coefficients

```
859 int get_feature_from_haar_normal(  
860     MASS_FLOAT4* trasformata,    /** Trasformata Wavelet dell'immagine */  
861     int dim,                     /** Dimensione della trasformata */  
862     int dim_w,                   /** Larghezza dell'immagine originale */  
863     int dim_h,                   /** Altezza dell'immagine originale */  
864     int last_wavelet_level,      /** Livello della trasformata */  
865     MASS_WAVELET_COEFF* feature, /** Struttura contenente i coefficienti */  
866     int* wave_active)  
867 {  
868     int k, x_or, y_or, x_ve, y_ve, x_di, y_di, i, width, height, row, col;  
869     k = 0;  
870     for(i=1; i<=last_wavelet_level; i++)  
871     {  
872         /** Estrae i coefficienti dei livelli selezionati */  
873         if(wave_active[i-1])  
874         {  
875             /** Calcolo del pixel di partenza per l'estrazione */  
876             x_or=x_di=(dim/(int)pow((MASS_FLOAT4)2, (MASS_FLOAT4)i));  
877             x_ve=0;  
878             y_ve=y_di=(dim/(int)pow((MASS_FLOAT4)2, (MASS_FLOAT4)i));  
879             y_or=0;  
880  
881             /** Calcolo della dimensione del quadrante */  
882             width = (dim_w/(int)pow((MASS_FLOAT4)2, (MASS_FLOAT4)i));  
883             height = (dim_h/(int)pow((MASS_FLOAT4)2, (MASS_FLOAT4)i));  
884  
885             /**  
886              * Estrazione dei coefficienti dalla trasformata  
887              * e inserimento nella struttura */  
888             for (row=0; row<height; row++)  
889                 for (col=0; col<width; col++)  
890                 {  
891                     feature->oriz[k+feature->size] = trasformata[((y_or+row)*dim)+x_or+col];  
892                     feature->vert[k+feature->size] = trasformata[((y_ve+row)*dim)+x_ve+col];  
893                     feature->diag[k+feature->size] = trasformata[((y_di+row)*dim)+x_di+col];  
894                     k++;  
895                 }  
896         }  
897     }  
898     /** Mantiene aggiornato il numero di coefficienti inseriti per quadrante */  
899     feature->size += k;  
900     return feature->size;  
901 }
```

Wavelet on GPU

Parallel implementation of the 2D Discrete Wavelet
Transform on Graphics Processing Units:
Filter-Bank versus Lifting.

Christian Tenllado, *Member, IEEE*, Javier Setoain, Manuel Prieto, *Member, IEEE*,
Luis Pinuel, *Member, IEEE*, and Francisco Tirado, *Senior Member, IEEE*