Implementació de l'Operador SURF en OpenCL

Codirectors: Sergio Escalera
Manel Martínez
Autor: Andrés Pardo

Índex

- Motivació
- Planificació
- Introducció
- Plataforma
- Desenvolupament
- Resultats
- Conclusions i treball futur

Motivació

- Interès en la computació paral·lela
- Interès en el hardware de GPUs
  - Processadors amb més capacitat de computació paral·lela actualment
- Interès en la visió per computador
  - Aplicar el paral·lelisme de GPUs en la visió per computador

Planificació

- Llegir el paper de SURF
- Llegir documentació i exemples d'OpenCL
- Entendre i analitzar el codi de la versió OpenCV
- Aconseguir fer funcionar demos d'OpenCL en hardware propi
- Realitzar versió en C++ com a base del desenvolupament de la versió OpenCL
- Realitzar versió OpenCL

Introducció al problema

- Adaptar un popular algorisme de visió per computador, SURF, a un framework GPGPU de nova creació compatible amb AMD i Nvidia
- L'algorisme original és de codi tancat per tant ens basàrem en la implementació de codi obert de la llibreria OpenCV

Introducció – GPGPU

- Computació paral·lela: dividir el problema en petites peces que seran calculades concurrentment
- General-purpose computing on GPU: aprofitar les característiques de les gràfiques actuals per a realitzar càlculs no relacionats amb la infografia

Introducció – Visió per Computador

- Extreure informació de les imatges necessària per a realitzar alguna tasca
  - Reconeixement
  - Control
  - Modelització
  - Restauració

Introducció – SURF

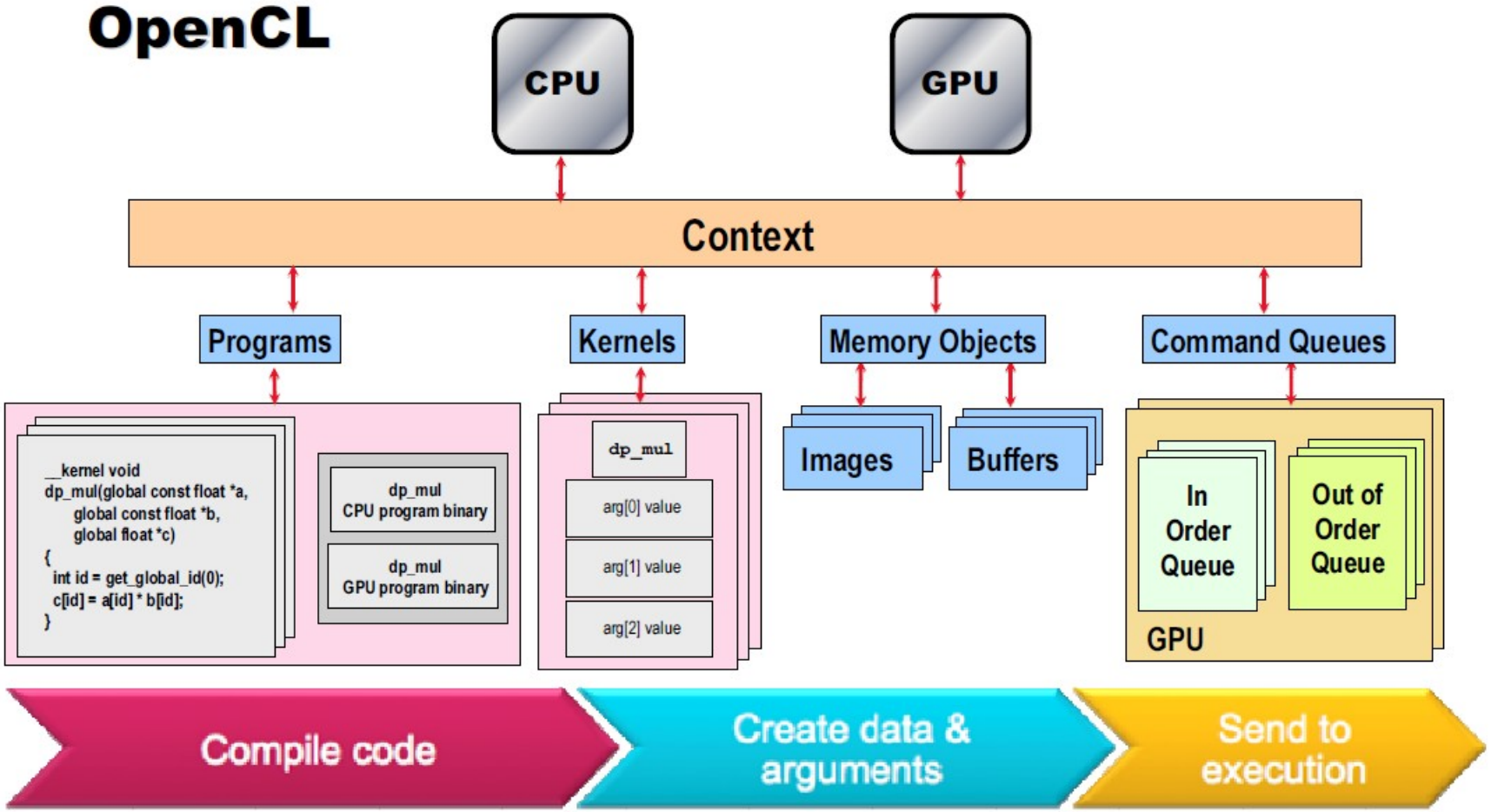
- Algorisme robust de detecció i descripció d'imatges
- Basat parcialment en SIFT
- Popular actualment
- Creat en 2006, Revisat en 2008

- Càlcul d'imatge integral
- Càlcul de la matriu de determinants
- Cerca del màxim
- Optimització dels punts trobats
- Càlcul d'Orientacions
- Càlcul del descriptor

Plataforma - OpenCL

- Framework per desenvolupar codi executable en CPUs, GPUs, DSP, etc
  - Compatible amb GPUs Nvidia i AMD
- Llenguatge similar a C99 per escriure kernels + API per definir i controlar la plataforma
- Task o data-based parallelism
  - Farem servir data-based parallelism per aprofitar les arquitectures SIMD de les GPUs

Plataforma - OpenCL



Desenvolupament

- Fer proves amb diferents codis com a kernel
- Escollir bé quines parts seran els kernels per a poder aprofitar al màxim el paral·lelisme de GPUs
- Trobar una mida de work-group adient per cadascun d'ells
  - Molt complicat si tenim com a objectiu més d'una arquitectura

Desenvolupament

- Llistat amb els kernels escollits
  - Imatge Integral en dos parts
  - Càlcul de keypoints
    - Mateixos resultats que amb el d'OpenCV
  - Càlcul d'Orientacions
  - Càlculs del descriptor
- Proves per a diferents arquitectures

Desenvolupament - Exemple

- Exemple càlcul de descriptors C++

  for(int d = 0; d<nPts; d++) {

  Inicialització de variables

  codi }

- Versió OpenCL

  size_t d = get_global_id(0);

  if(d < nPts) {

  Inicialització de variables+codi}

Desenvolupament - Exemple

- Creació del programa amb els 5 kernels dins el context
- S'inclou el kernel a la cua d'execució
- S'executa
- S'encua la lectura dels resultats

Resultats - Equips

Equip Portàtil – Baixa Potència

Pentium Dual-Core T4400 2 nuclis a 2'2GHz

Nvidia 9100M G 256MB de la memòria DDR2 principal bus de 128bits 1 SIMD

2x2GB de DDR2

Windows 7 64bits

Equip Nvidia – Alta Potència

Intel Core i5 750 4 nuclis a 2'66GHz

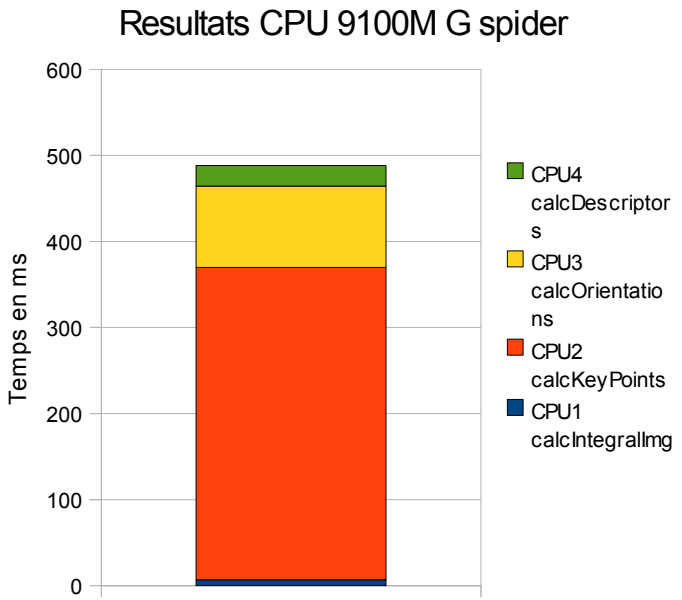
Nvidia GTX 470 1280MB GDDR5 bus de 320bits 14 SIMD

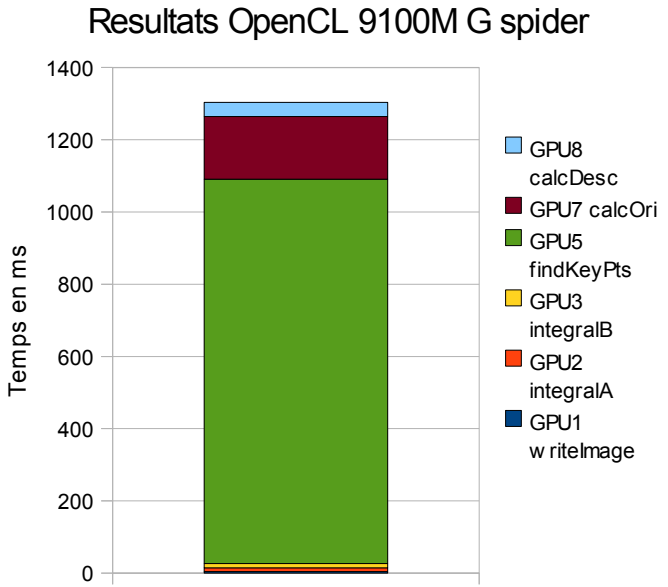
4x2 GB DDR3

Ubuntu 10.04 64bits

Resultats - Nvidia 9100M G

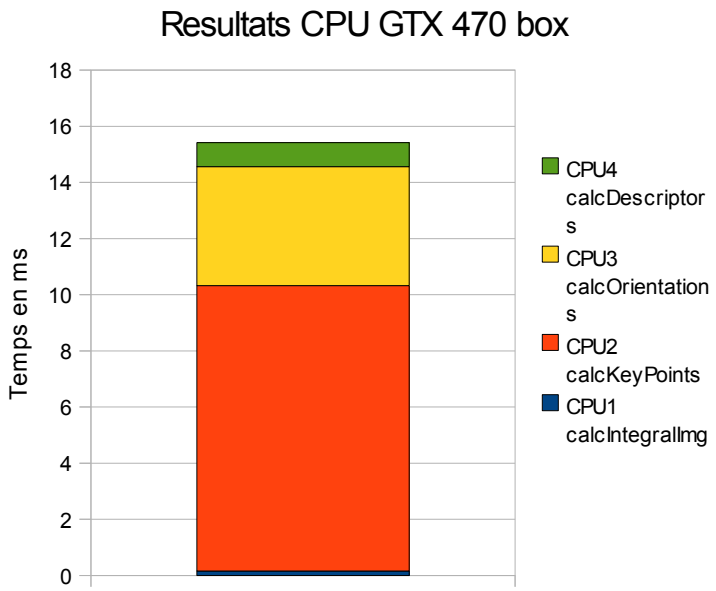
- Entre 2 i 4 vegades més lent en funció de l'entrada

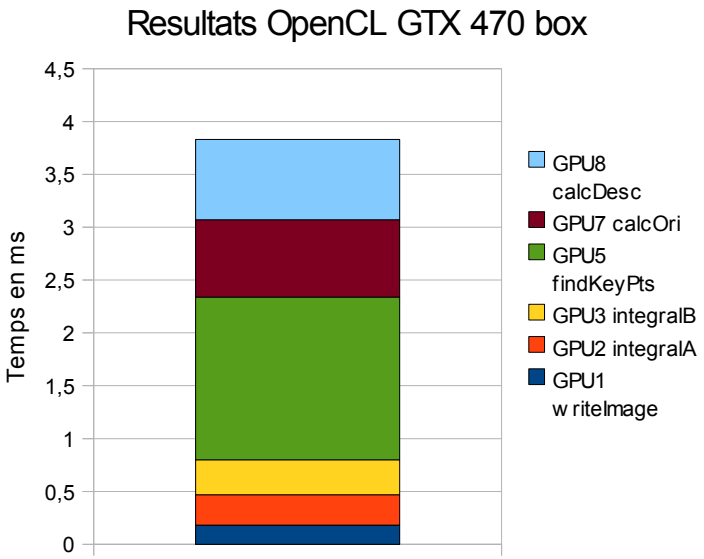




Resultats - Nvidia GTX 470

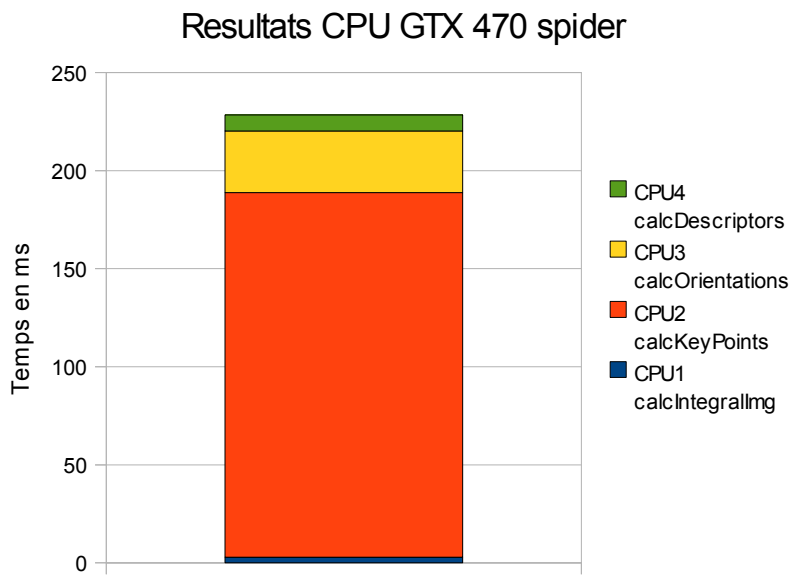
- Unes 4 vegades més ràpid per imatges petites

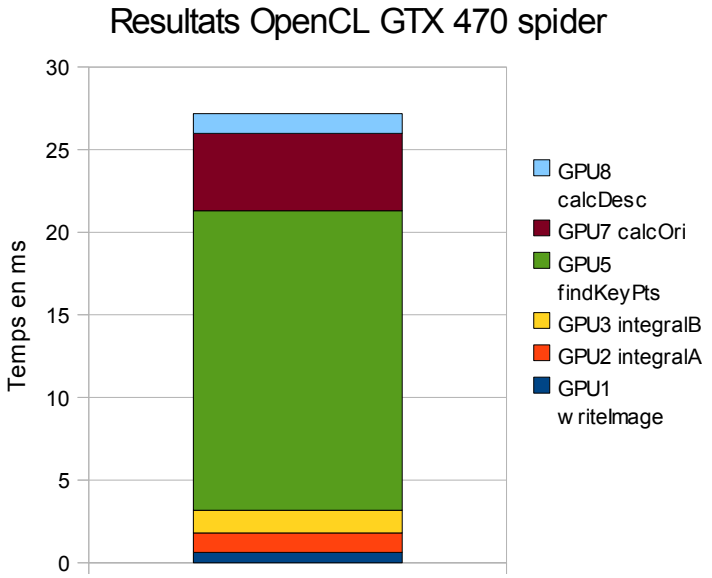




Resultats - Nvidia GTX 470

- 8-9 vegades més ràpid per imatges grans





Objectius vs Resultats

- Keypoint detector funcionant en qualsevol arquitectura OpenCL
- Més velocitat que en CPU sense perdre robustesa

- Keypoint detector i orientació + descriptor funcionant correctament en Nvidia, Keypoint detector en AMD
- Entre 4 i 9 vegades més ràpid per GPUs actuals de gama mitja-alta

Conclusions

- SURF OpenCL té sentit en gràfiques Nvidia de gama mitja o alta dels últims 2 o 3 anys.
- Els productes integrats no són prou potents actualment
- És molt difícil programar per aquestes arquitectures sense optimitzar per una en concret si s'esperen resultats realment bons

Treball Futur

- Aconseguir una implementació compatible completament amb AMD
- Provar les noves plataformes portàtils APU, CPU+GPU, d'AMD
- Adaptar l'algorisme per arquitectures Intel