Performance Analysis of Accelerated Image Registration Using GPGPU

Peter Bui <pbui@cse.nd.edu>
Jay Brockman <jbb@cse.nd.edu>

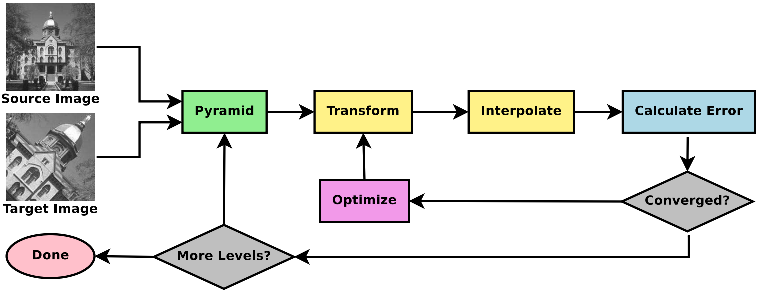
University of Notre Dame, IN, USA

Workshop on General Purpose Processing on Graphics Processing Units, 2009

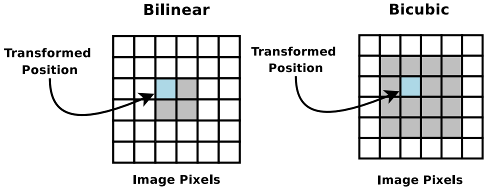
Image Registration (Overview)

- **Objective:** Find transformation coefficients that map source image to target image.
- **Applications:** Remote sensing, computer vision, image-guided surgery, etc.

Image Registration (Algorithm)



- **Pyramid**: Allows for course-to-fine grain optimization.
- **Interpolation**: Bilinear and Bicubic



Previous Work

OpenGL/DirectX

- **Ino, Gomita, Kawasaki, Hagihara (*ISPA*, 2006)**
  2-D/3-D rigid image registration speedup by $5.0\times$ to $9.6\times$.
- **Kubias, Deinzer, Feldmann, Paulus (*PRIA*, 2008)**
  Speedup rigid image registration by $3\times$ to $6\times$ and experimented with different similarity measurements.

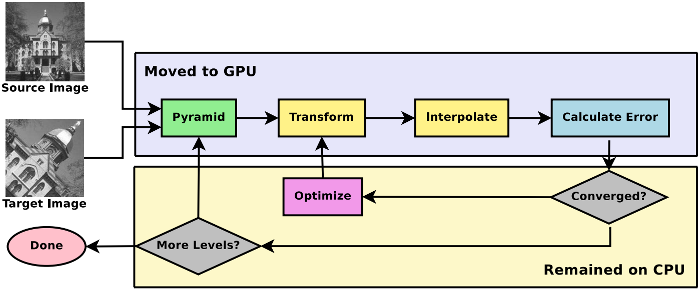
CUDA

- **Sugiura, Deguichi, Kitasaka, Mori, Suenaga (*AMI-ARCS*, 2008)**
  Accelerated rigid image registration used in bronchoscope tracking by a factor of $16\times$.

GPGPU Implementation

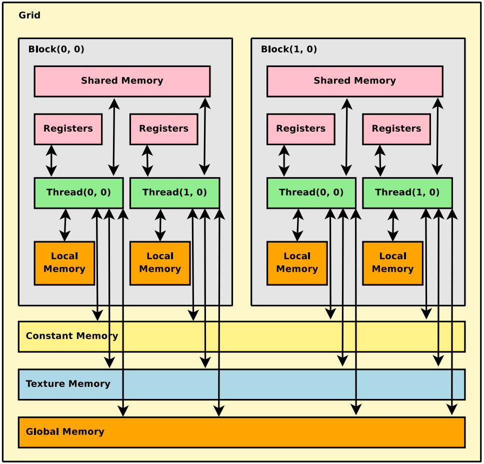
Minimize kernel calls and memory transfers.

- Construct Pyramid image stacks on GPU.
- Perform Transform, Interpolate, Calculate Error in one CUDA kernel.
- Compute partial sum of the mean square errors.
- Keep Optimize on CPU.



GPGPU Implementation (CUDA Organization)

Minimize global memory accesses.



- Store images in texture memory.
- Read transformation matrix from constant memory.
- Build partial sums in from shared memory.

Experimental Setup

System Configuration

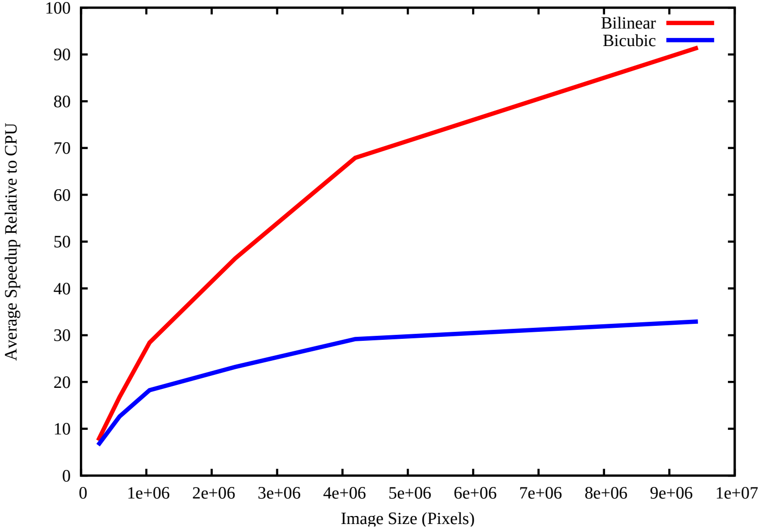
- **Hardware:**
  - Intel Quad-Core Q6700 2.66 GHz CPU, 8.0 GB
  - NVIDIA Tesla C870, 128 Stream Processors, 1.0GB
- **Software:**
  - Ubuntu 8.04 (kernel 2.6.20)
  - GCC 4.1.2
  - NVIDIA CUDA 1.1 SDK

Test Images

Image	Dimensions (Pixels)
lenna	512×512
ndbuntu	768×768
halo	1024×1024
jump	1536×1536
victory	2048×2048
crabnebula	3072×3072

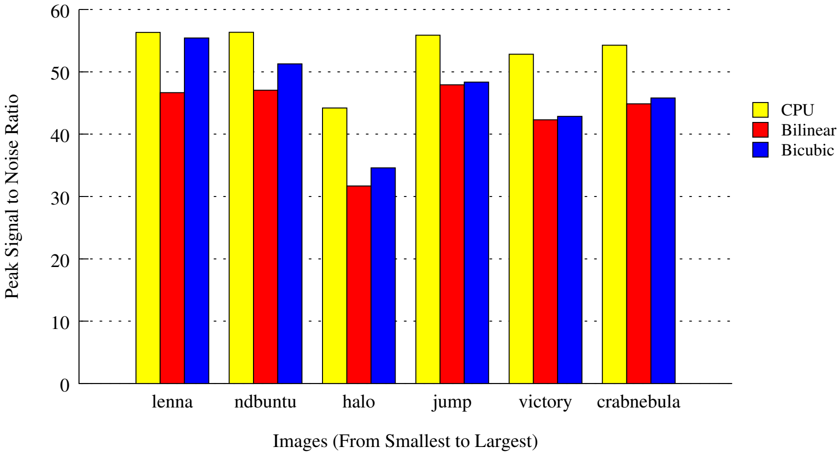
Experimental Results (Performance)

Average Speedup Over CPU

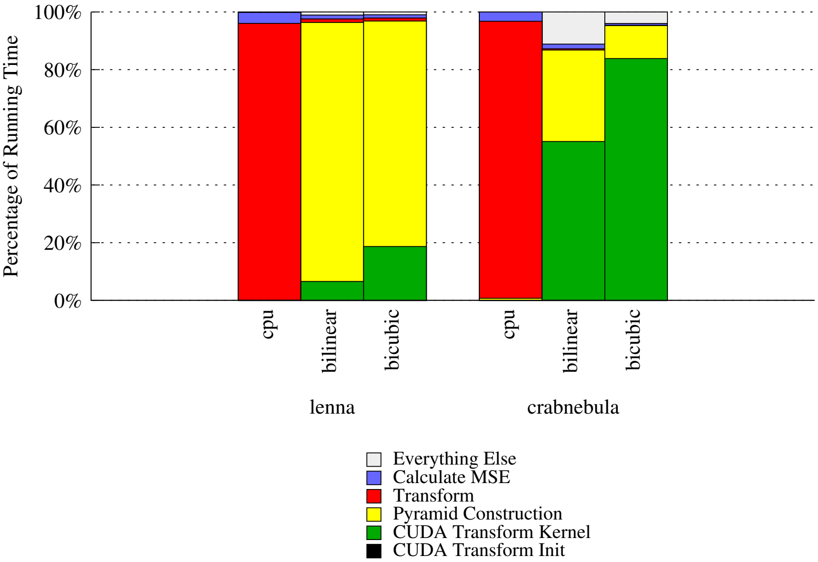


Experimental Results (Accuracy)

Average Peak Signal To Noise Ratio (Higher is Better)



Profiling


100%
80%
60%
40%
20%
0%
Percentage of Running Time
cpu
bilinear
bicubic
cpu
bilinear
bicubic
lenna
crabnebula
Everything Else
Calculate MSE
Transform
Pyramid Construction
CUDA Transform Kernel
CUDA Transform Init

Conclusion

Performance

- 7.5$\times$ to 91.5$\times$ speedup for the bilinear version.
- 6.5$\times$ to 33.0$\times$ speedup for the bicubic version.
- Speedup limited by CUDA device initialization time

Accuracy

- CUDA kernels yield PSNRs in the range of $35 - 55$.
- Overall, bicubic interpolation more accurate than bilinear.
- Accuracy affected by GPU floating point implementation.

Future Work

- Amortize CUDA device initialization time.
- Explore faster interpolation methods.
- Consider alternative optimization algorithms.

Experimental Results (Summary)

Image	**Version**	**Run-time**	**Speedup**	**PSNR**
lenna	CPU	5.19	1.00	56.32
	Bilinear	0.69	7.48	51.60
	Bicubic	0.80	6.50	55.43
ndbuntu	CPU	12.49	1.00	56.35
	Bilinear	0.74	16.80	47.05
	Bicubic	0.98	12.64	51.27
halo	CPU	23.73	1.00	44.19
	Bilinear	0.83	28.40	31.70
	Bicubic	1.30	18.25	34.60
jump	CPU	48.09	1.00	55.86
	Bilinear	1.04	46.41	47.92
	Bicubic	2.07	23.22	48.34
victory	CPU	92.50	1.00	52.83
	Bilinear	1.36	67.89	42.31
	Bicubic	3.16	29.17	42.85
crabnebula	CPU	205.31	1.00	54.26
	Bilinear	2.24	91.47	44.86
	Bicubic	6.24	32.92	45.80