Fast and efficient dense variational stereo on GPU

http://www.enpc.fr/certis

J. Mairal - R. Keriven
A. Chariot
CERTIS, ENPC
Marne-La-Vallée, France

Problem Statement: Achieve fast variational stereo reconstruction.

Context:
- Variational stereo algorithms are usually slow but provide smooth and accurate results.
- Graphics cards can be considered nowadays as very efficient parallel SIMD machines.

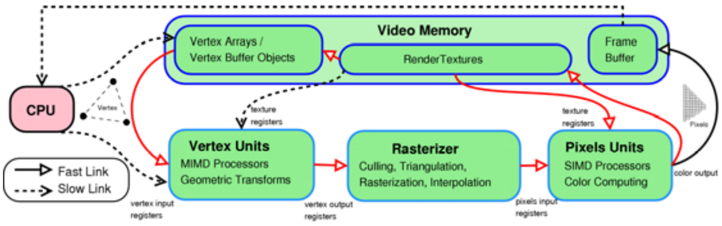
Achievements:
- Parallelization and discretization of an algorithm on a GPU.
- speedup of 10-15 times between a recent GPU and a CPU (3 GHz).
- Extension to three cameras.

Future work:
- Work with more than one GPU.
- Achieve fast multi-view reconstruction on GPU.
- taking discontinuities into account.

GPU Programming

Constraints:
- Concurrent Read, Exclusive Write.
- Iterative algorithms must run *entirely* on the GPU to be efficient.
- Data structures are Textures.

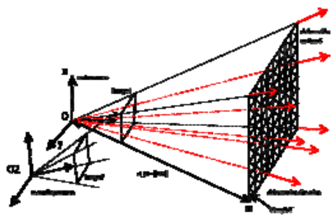
Tools:
- OpenGL, Cg
- Vertex Buffer Object, Frame Buffer Object (or RenderTextures)



GPU Discretization

A simple two-cameras model which runs entirely on the GPU.

- The mesh is the input of each iteration and is stored in a Vertex Buffer Object.
- Projections are performed in Vertex Shaders.
- Interpolation features of the rasterizer are used.
- A gradient descent is performed by pixel shaders.



A robust energy based on normalized cross-correlation.

- The images are back-projected onto the surface S which is composed of triangles T

$$E(S) = \sum_T E_T = \sum_T (1 - \rho_T(I_1 \circ \Pi_1, I_2 \circ \Pi_2)) + \Gamma(S)$$

- ρ_T denotes the normalized cross-correlation function on the triangle T
- $\Gamma(S)$ denotes a regularization function which is adapted for a GPU implementation.

Energy minimization by means of a descent gradient.

- Several levels of detail to prevent from converging toward a local minima.
- for each vertex M of the mesh,

$$\frac{\partial E(S)}{\partial d_M} = \sum_{T \in V(M)} \frac{\partial E_T}{\partial d_M}$$

is computed, where $V(M)$ is the set of the 6 triangles to which M belongs.

d_M is the distance between M and O.

- for each triangle $T = M_1 M_2 M_3$, $\left(\frac{\partial E_T}{\partial d_{M_1}}, \frac{\partial E_T}{\partial d_{M_2}}, \frac{\partial E_T}{\partial d_{M_3}}\right)$ are computed

simultaneously thanks to the parallel computing capabilities of the GPU.

- Some operations are performed at the vertex level. A pixel of one triangle receives interpolated values from the vertices of the triangle it belongs to.

Direct regularization by smoothing d_M.

- Each vertex tends to the mean value of the $d_{M'}$, where M' are its neighbors. The obtained results here were better than the use of mean curvature motion.

An extension to three cameras

An algorithm almost as fast as the two-cameras version, which still runs on the GPU.

- One camera is denoted as the « reference » or « center » camera.
- We consider the « left » and the « right » camera.
- The triangles are classified in a first step in three categories, using the Stencil Buffer of the graphics cards :
  - occluded triangles.
  - triangles associated with the « left » camera: S_L
  - triangles associated with the « right » camera: S_R
- For each « non-occluded » triangle, a simple normal test to select the best camera.
- Then, the two-camera algorithm is run on S_L, then on S_R.
- The complexities of the two-cameras and the three-cameras version are the same.
- The overhead between them is about 30% on the GPU.

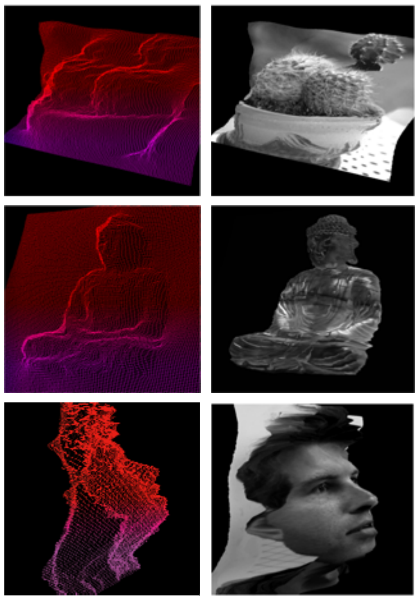
Results on data sets with occluded areas are better.

Two cameras with occluded areas : Reconstruction problems occur.

With three cameras and a camera selection for eath triangle, the reconstruction is correct.

This dataset comes from `http://www. cs.ust.hk/~quan/WebPami/pami.html`

Results



Smooth and accurate results.

- These examples were obtained with two input images.
- The two first datasets are courtesy of Kyros Kutulakos (University of Toronto).

Fast results

- These examples were computed in less than 250ms with a NVIDIA Geforce 7800 GTX.
- Not as fast as plane-sweep based methods, but more accurate !

GPU/CPU speedup : 10 to 15

- Geforce 7800 vs 3GHz CPU, Iterations per second for one particular level of detail.

Image	Mesh	GPU	CPU	Speedup
64^2	5^2	1.60 kHz	555 Hz	2.9
128^2	9^2	1.33 kHz	116 Hz	11.5
256^2	17^2	464 Hz	28.6 Hz	16.2
512^2	33^2	102 Hz	7.5 Hz	14.1
512^2	65^2	89.4 Hz	7.3 Hz	12.2
512^2	129^2	67.9 Hz	7.2 Hz	9.0

[1] R. Keriven, O. Faugeras : "*Variational principles, surface evolution, PDEs, level set methods and the stereo problem*". IEEE Transactions on Image Processing. 1998.
[2] J. Mairal, R. Keriven : "*A GPU implementation of variational stereo*". Internal Report CERTIS. 2005
[3] C. Zach, A. Klaus, M. Hadwiger, K. Karner : "*Accurate Dense Stereo Recontruction using Graphics Hardware*". EUROGRAPHICS 2003.
[4] I. Geys, T.P. Koninckx, L. Van Gool : "*Fast Interpolated Cameras by combining a GPU based Plane Sweep with a Max-Flow Regularization Algorithm*", 3DPVT'04. 2004
[5] J. Woetzel, R. Koch. "*Multi-camera Real-time Depth Estimation with Discontinuity Handling on PC Graphics Hardware*". ICPR 2004.
[6] R. Yang, M. Pollefeys "*A Versatile Stereo Implementation on Commodity Graphics Hardware*" Real-Time Imaging Vol 11, 2005.
[7] M. Gong, Y-H. Yang "*Near Real-time Reliable Stereo Matching Using Programmable Graphics Hardware.*", CVPR'05, 2005.
[8] `http://www.gpgpu.org`