TOWARDS REAL-TIME AND ACCURATE VOXEL COLORING FRAMEWORK

Oussama Moslah, Arnaud Debeugny, Vincent Guitteny, Serge Couvet
THALES Security Solutions and Services, 1 Rue du General de Gaulle, 95523 Cergy-Pontoise, France

Sylvie Philipp-Foliguet
ETIS - UMR CNRS 8051, ENSEA, 6 Avenue du Ponceau, 95014 Cergy-Pontoise, France

Keywords: Voxel coloring, Visual hull, Marching cubes, Octree, GPGPU.

Abstract: This paper presents algorithms and techniques towards a real-time and accurate Voxel Coloring framework. We combine Visual Hull, Voxel Coloring and Marching Cubes techniques to derive an accurate 3D model from a set of calibrated photographs. First, we adapted the Visual Hull algorithm for the computation of the bounding box from image silhouettes. Then, we improved the accuracy of the Voxel Coloring algorithm using both colorimetric and geometric citerions. The calculation time is reduced using an Octree data structure. Then, the Marching Cubes is used to obtain a polygonal mesh from the voxel reconstruction. Finally, we propose a practical way to speed up the whole process using graphics hardware capababilities.

1 INTRODUCTION

In this paper we address the problem of real-time 3D reconstruction from photographs. Our framework consists of three parts: (1) computation of the bounding box of the object we want to reconstruct using a Visual Hull approach, (2) a voxel reconstruction based on both colorimetric and geometric criterions (3) and a generation of a polygonal mesh using Marching Cubes techniques. The context of this work is the growing interest in automatic reconstruction techniques from photographs. With the increasing capababilities of modern graphics hardware 3D reconstruction techniques can be accelerated to obtain accurate models in real-time.

2 RELATED WORK

The original Voxel Coloring paper described in (Seitz and Dyer, 1997) uses only colorimetric criterions to reconstruct an object consistent with the input images. This algorithm starts by discretizing the 3D space into voxels and projects them on each image. The voxels that are consistent from a colorimetric viewpoint with the images are retained. The complexity of this algorithm is $O(N^3 * n)$ with N^3 is the number of voxels and n the number of images. In order to improve the accuracy of this method we use both colorimetric and geometric criterions to derive 3D models from image silhouettes. The calculation time is improved using an Octree data structure. The Visual Hull algorihm (Franco and Boyer, 2003) operates in a different manner by projecting the image silhouettes into the 3D space. The intersection of the silhouettes cones produces the 3D polygonal model. We adapted this algorithm to compute the bounding box of the 3D object which is needed for the voxel reconstruction. Instead of projecting the image silhouettes into the 3D space we project their 2D bounding boxes. The Marching Cubes technique (Lorensen and Cline, 1987; F. Goetz, 2005) takes as input a 3D point cloud and produces a textured polygonal mesh. Acceleration using graphics hardware has been for a long time restricted to purely graphical processing. With the constant evolution of graphics hardware and the emerging GPGPU (General Purpose GPU) techniques and technologies such Cg (W. R. Mark, 2003) and CUDA (Cuda, 2008) researchers start to re-design their algorithms to benefit from the parallel capababilities of modern GPUs (Trendall and Steward, 2000; Krueger and Westermann, 2003; F. Goetz, 2005).

Moslah O., Debeugny A., Guitteny V., Couvet S. and Philipp-Foliguet S. (2009).
TOWARDS REAL-TIME AND ACCURATE VOXEL COLORING FRAMEWORK.
In *Proceedings of the Fourth International Conference on Computer Vision Theory and Applications*, pages 490-495
DOI: 10.5220/0001794704900495

Figure 1: A sample data set: Our system takes as input a set of calibrated images and silhouettes and produces a textured polygonal model (VGG, 2008).

3 OUR APPROACH

Given a set of calibrated images and silhouettes our system produces a textured polygonal model. Figure 1 illustrates a sample input data set used for evaluating our reconstruction pipeline. The proposed solution described consists of three parts. First we compute the bounding box of the object we want to reconstruct using a Visual Hull approach. Then we reconstruct the 3D object with a colorimetric and geometric consistency based Voxel Coloring scheme. finally, we produce a textured polygonal model using The Marching Cubes technique. The Voxel Coloring algorithm is accelerated using graphics hardware.

3.1 Visual Hull

The Visual Hull algorihm (Franco and Boyer, 2003) computes a 3D coarse representation of an object from it's 2D projections in a set of images. Figure 2 illustrates this algorithm in the simple case of 2 calibrated images. Given (1) two projection matrices P_A and P_B (2) two 2D regions D_A and D_B representing the projection of the same 3D object we compute the 3D cones V_A and V_B and intersect them to compute the coarse 3D model C.

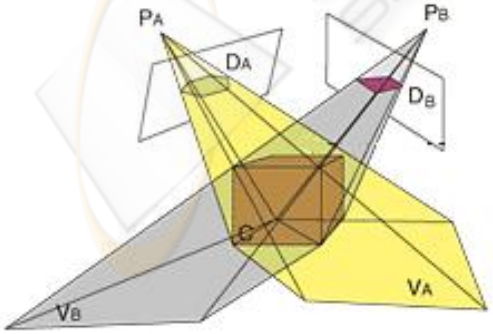


Figure 2: Illustration of the Visual Hull principle.

We adapted this algorithm to compute the bounding box of the 3D object which is needed for the voxel reconstruction. Instead of projecting the image silhouettes into the 3D space we project their 2D bounding boxes.

3.2 Voxel Coloring

The Voxel Coloring algorithm described in (Seitz and Dyer, 1997) uses a colorimetric criterion to decide if a voxel is consistent or not. Thus voxels can be colorized even if their projection is totally outside of the object silhouette. In order to improve the accuracy of the Voxel Coloring algorithm we add a geometric criterion (Kuzu and Rodehorst, 2001). Figure 3 illustrates the principle of the use of silhouettes: voxels are identified respectively as gray, black or white depending if their projection into images falls in the boundary, outside or inside of the silhouettes.

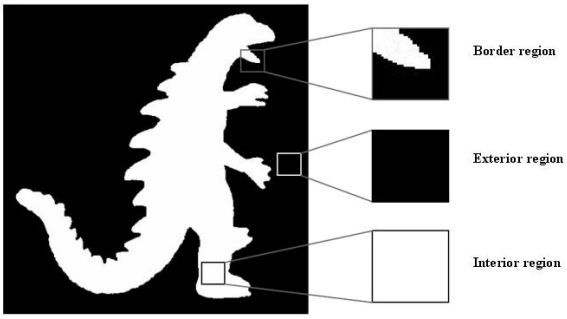


Figure 3: Illustration of the geometric consistency check.

To reduce the complexity of the algorithm we use an Octree data structure (A. W. Fitgibbon, 1998). We recursively subdivide the volume into 8 subvolumes. The subdivision of a volume is made only if it's projection into images is on the boundary of the silhouette. Thus the object is reconstructed in an economic way. Figure 4 shows an illustration of a volume subdivision and it's associated data structure.

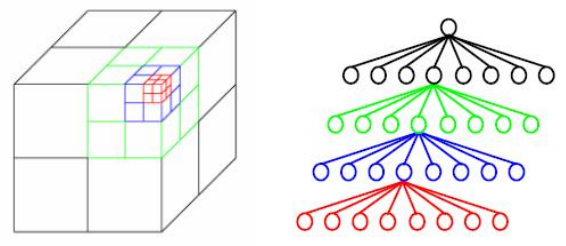


Figure 4: Recursive subdivision of a volume and the associated octree.

We also use a 3D connexity check to improve the surface of the voxel reconstruction. We proceed as follows to update the voxel classification: (1) we check the 6 neighbours of each gray voxel and if no black voxel is found we identify it as white, (2) we also check the neighbourhood of each white voxel and if at least one black voxel is found we identify it as gray. Figure 5 illustrates the principle of this algorithm.

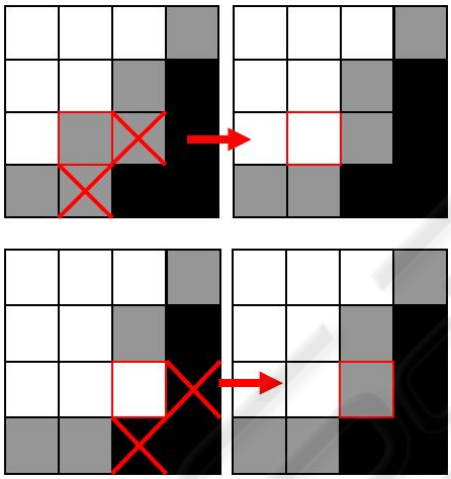


Figure 5: Illustration of the voxels connexity check.

3.3 Marching Cubes

The Marching cubes algorithm (Lorensen and Cline, 1987) is used to obtain a polygonal model from a scattered set of voxels. The algorithm starts by taking eight neighbor locations to construct a cube, then determine the polygons that passes through this cube. The individual polygons are then fused into the model surface. We use an index of precalculated array of 256 possible polygon configurations ($2^8 = 256$) within the cube. This array of 256 cube configurations is obtained by reflections and symmetrical rotations of the basic cases illustrated by Figure 6.

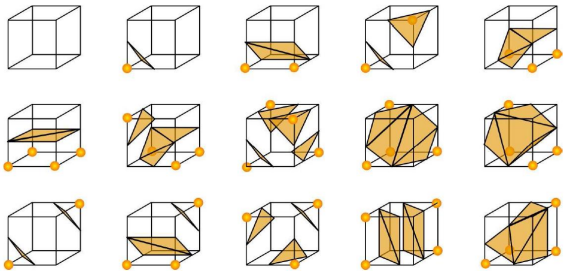


Figure 6: 15 basic configurations of polygons.

3.4 Acceleration using Graphics Hardware

The GPU architecture is specialized for parallel computing tasks. The graphics hardware consists of a set of processors grouped together in a common multiprocessors block. Figure 7 illustrates the hierarchy of the parallel architecture of actual GPUs.

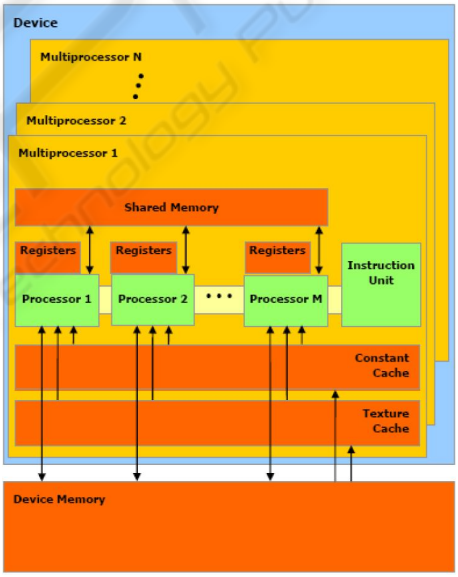
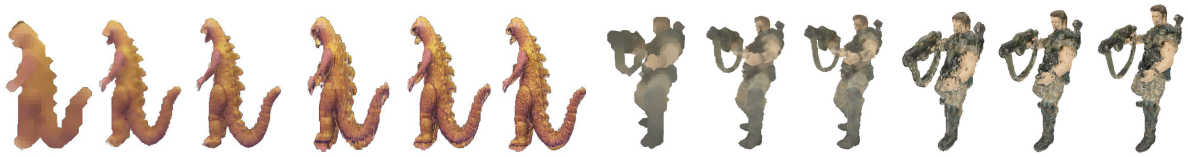


Figure 7: Graphics hardware parallel architecture.

For example the device used in this work is the NVidia GeForce 8800GTS card. This card mainly consists of 12 multiprocessors and a 512 MB device memory. Each multiprocessor is composed itself by 16 processors, a shared memory and an instruction unit. This card can thus make 192 calculations in parallel. Unlike the device memory the shared memory consists of 16 Kb and is accessible only by processors belonging to the same multiprocessor unit. However this memory is very usefull and can be accessible much more faster than the device memory. To execute hundreds of processes working in various programs, multi-processors use new architecture called



(a) (b) (c) (d)

Figure 8: Reconstruction results for three different level of details ($64 = 2^6$, $128 = 2^7$, $256 = 2^8$) (a,c) Calculation using the CPU (b,d) Calculation using the GPU.

SIMT (Single Instruction Multiple Thread).

The Voxel Coloring process is accelerated using the parallel capabilities of modern graphics hardware. Unlike the original Voxel Coloring algorithm (Seitz and Dyer, 1997) we use an Octree data structure (Szelinski, 1993). First, each volume is subdivided into 8 subvolumes. This gives rise to 27 points (we remove redundant 3D points). The projection is made using the well known pinhole camera model (Zissermann and Hartley, 2003) that describes how a 3D point M with coordinates (X, Y, Z) in the world coordinate space projects into an image point m with coordinates (u,v) in pixels using the classic perspective transformation:

$$m \cong K \left[R^T \mid -R^T t \right] M = PM \tag{1}$$

Where R and t respectively represent the camera orientation and position, K the camera matrix or matrix of intrinsic parameters and P the projection matrix. In order to optimize the use of parallel architecture of the graphics hardware we assign a different process for image point coordinate computation using the simple formula:

$$m[p+27n] = \sum_{k=0}^{3} P[4n+12i+k] * M[4p+k] \tag{2}$$

Where m, P and M respectively represent the concatenated matrices of 2D image points, projection matrices and 3D points. Then the result is normalized and we obtain 27 points in homogenous coordinates (u,v,1) for each image using the formula :

$$m[p+27n] = \frac{m[p+27n]}{m[p+54]} \tag{3}$$

Finally, a subvolume can be recursively subdivided if the projection into images falls in the boundary of the silhouette. Thus we get better level of details without the need of subdividing all the voxels.

4 RESULTS

Figure 8 illustrates the final results obtained with our Voxel Coloring framework for different level of details. The level of detail correspond to the depth of the Octree data structure. It is the maximum number of recursive subdivision of a voxel. Figure 9 shows the reconstruction results of the Dinosaur and the Soldier for the maximum level of detail. Evaluation of the results and the computing times of our framework are presented in Table 1.

5 DISCUSSION

Comparison to Previous Work. Most previous work on reconstruction techniques from image silhouettes using Voxel Coloring algorithm either use geometric or colorimetric approach. In our work we combine those two criterions to get an accurate reconstruction. We also use an Octree data structure to improve the computing times and propose a way to accelerate the algorithm using parallel processing capabilities of modern GPUs.

Robustness and Limitations. The robustness of our approach strongly depends on the quality of silhouettes. Actually only the Voxel Coloring algorithm is accelerated using the GPU. Thus our framework is a mixed CPU/GPU implementation.

Future Work. We are working on the implementation of the Marching Cubes and the Visual Hull algorithms using the GPU and we expect to get real-time computing times for the whole process. We are also working on photorealistic rendering of the reconstructed model using view-dependent texture mapping techniques (P. E. Debevec, 1996).

6 CONCLUSIONS

This paper introduced methods and techniques for real-time recovering of accurate textured 3D models

(a) (b)

Figure 9: Results using the highest level of detail ($256 = 2^8$). (a) the Dinosaur reconstructed (b) the Soldier reconstructed.

Table 1: Evaluation results of our Voxel Coloring framework using the Nvidia card 8800GTS 512MB on a Dualcore Intel Pentium 4 3.2Ghz with 2GB RAM.

		Dinosaur			Soldier		
		64	128	256	64	128	256
Visual Hull (seconds)		0.442			3.703		
Voxel Coloring (seconds)	CPU	92.4	518	5449	56.47	200	1614
	GPU	1.50	3.25	16.44	0.79	1.98	8.33
Marching Cubes (seconds)		2.37	12.49	97.16	3.13	13.56	90.34
Mesh (number of triangles)		18736	73672	308956	24020	93652	392468

from image silhouettes. The main contributions of this work is the way we compute the bounding box using a Visual Hull approach, the combined colorimetric and geometric criterions used inside the Voxel Coloring algorithm and the way the computations are accelerated using the parallel capabilities of modern GPUs.

7 ACKNOWLEDGEMENTS

We wish to acknowledge the Cap Digital Business Cluster Terra Numerica project for sponsoring the research reported in this paper.

REFERENCES

A. W. Fitgibbon, G. Cross, A. Z. (1998). Automatic 3d model construction for turn-tables sequences. *Lectures notes in Computer Sciences*.

Cuda (2008). Cuda: Compute unified device architecture, www.nvidia.com/cuda. NVidia.

F. Goetz, T. Junklewitz, G. D. (2005). Real-time marching cubes on the vertex shader. In *Eurographics*.

Franco, J. and Boyer, E. (2003). Exact polyhedral visual hull. In *British Machine Vision Conference (BMVC'03)*, volume I, pages 329–338.

Krueger, J. and Westermann, R. (2003). Acceleration techniques for gpu-based voluume rendering. In *IEEE Visualization'03*.

Kuzu, Y. and Rodehorst, V. (2001). Volumetric modeling using shape from silhouette, photogrammetry and cartography.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH'97*.

P. E. Debevec, C. J. Taylor, J. M. (1996). Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH'96*.

Seitz, S. M. and Dyer, C. R. (1997). Photorealistic scene reconstruction by voxel coloring. In *Computer Vision and Pattern Recognition Conf.*, pages 1067–1073.

Szelinski, R. (1993). Rapid octree construction from images sequences. In *CVGIP*, pages 23–32.

Trendall, C. and Steward, A. J. (2000). General calculation using graphics hardware, with application to interac-

tive caustics. In *Eurographics Workshop on Rendering*, pages 287–298. Springer.

VGG (2008). Visual geometry group dataset. www.robots.ox.ac.uk/˜vgg/data/data-mview.html.

W. R. Mark, R. S. Glanville, K. A. M. K. (2003). Cg: A system for programming graphics hardware in a c-like language. In *Proceedings of SIGGRAPH*.

Zissermann, A. and Hartley, R. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University press, 2nd edition.