Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware

Ruigang Yang and Marc Pollefeys*
Department of Computer Science, University of North Carolina at Chapel Hill

1 Introduction

We demonstrate a stereo algorithm that is implemented using only the OpenGL APIs [Yang and Pollefeys 2003]. It allows a standard Graphic Processor Unit (GPU), which can be found in every commodity PC with an accelerated graphics card, to perform many tens of millions of disparity evaluations per second and frees up the main processor for other tasks including high-level interpretation of the stereo results.

At the heart of our method is a multi-resolution approach to achieve good results close to depth discontinuities as well as on low texture areas. We combine the sum-of-square-differences (SSD) dissimilarity measures for windows of different sizes. This is, in fact, equivalent to using a large *weighted* correlation kernel with a pyramid shape. By utilizing the mipmap functionality on the graphics hardware, we can compute this dissimilarity measure very efficiently.

Our implementation extends the work of [Yang et al. 2002]. When running on an NVIDIA GeForce4 graphics card, it can achieves 50-70M disparity evaluations per second including all the overhead to download images and read-back the disparity map, which is equivalent to the fastest commercial CPU implementations available. An important advantage of our approach is that rectification is not necessary so that correspondences can also be obtained for images that contain the epipoles. Another advantage is that this approach can easily be extended to multi-baseline stereo.

2 Multi-Resolution Stereo on Commodity Graphics Card

Due to the lack of space, we briefly outline our method here. Interested readers are encourage to consult the full paper that is included in the proceedings of CVPR 2003 [Yang and Pollefeys 2003].

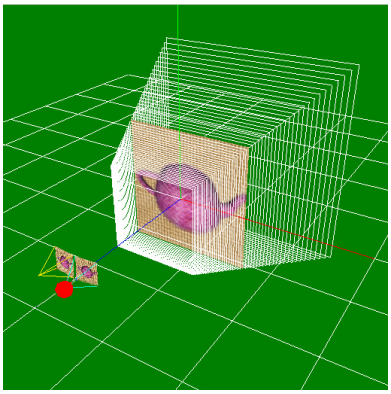

Figure 1: A illustration of our plane-sweep approach. the red dot represents the reference view. Spaces are discretized into a number of parallel planes.

*E-mail:{ryang, marc}@cs.unc.edu. This work was supported in part by the NSF grant ISS-0237533, and by a generous 2002-2003 Link Foundation fellowship.

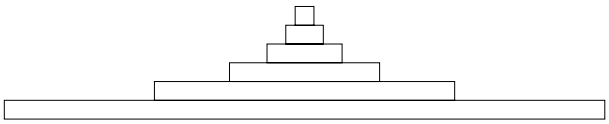

Figure 2: Shape of kernel for 6-level SSD.

To efficiently compute dense correspondence maps between two images using graphics hardware we use a plane-sweep approach [Collins 1996]. Given a plane in space, it is possible to project both images onto it using projective texture mapping. If the plane is located at the same depth as the recorded scene, pixels from both images should be consistent. This can be verified by evaluating the square difference (SD) between the pixel intensities. This image difference operation can be carried out using the programable pixel shader available in today's graphics card. To estimate a dense set of correspondences, a plane hypothesis is set up for every possible disparity (depth) value. Input images are warped on every plane through texture mapping, and a matching cost (SD) is computed for every pixel. Then we can simply select the best match along the line of sight of every pixel in one of the two images.

In the two-view stereo case, it is necessary to use larger support region to aggregate the SD scores to provide a more robust estimate. GPUs have build in box-filters (in hardware) to efficiently generate all the mipmap levels needed for texturing. Therefore, it is very efficient to sum values over $2^n \times 2^n$ windows. When a mipmap is built, we effectively obtain a sum of squared difference (SSD) images with a power-of-two support size at each level.

We use a multi-resolution approach to sum up SSD images at different levels. This can easily be done by using multiple texturing units. This approach, in fact, corresponds to using a large window, but with larger weights for pixels closer to the center. An example of a kernel is shown in Figure 2. The peaked region in the middle allows good localization while the broad support region improve robustness.

3 Results

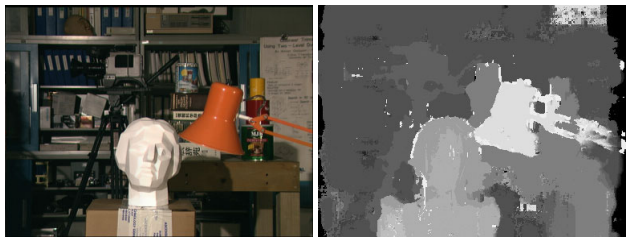

Figure 3: Calculated disparity maps from the Tsukuba set.

We have tested our implementation on a variety of image pairs. In Figure 3 and 4, we show our results from a few data sets that have been widely used in the computer vision literature, they are computed with 4-level SSD.

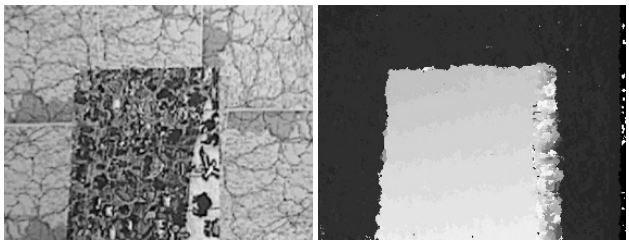

Figure 4: Calculated disparity maps from another stereo pair.

Output Size	Search Range	Times (ms)	Times (Hz)	Img. Update (ms)	Read (ms)	Disp. Calc. (M/sec)
512^2	20	71.4	14	(VGA)	6.0	58.9
	50	182	5.50	5.8×2		65.6
	100	366	2.73			68.3
256^2	20	20.0	50	(QVGA)	1.5	53.1
	50	49.9	20	1.6×2		60.0
	100	99.0	10.1			63.2

Table 1: Performance on an NVIDIA GeForce4 card when summing four (4) mipmap levels. The two input images are 640×480.

We also measured the performance on an NVIDIA GeForce4 card. As in Table 1 and Figure 5, we can see that our method exhibits real-time performance—50-70 million disparity calculations/second, as well as very good linearity with respect to the image size.

We also implemented a real-time system that captures and processes live data online. Our current prototype performs a few additional steps in software, such as radial distortion correction and segmentation.[1] As a proof of concept, these yet-to-be-optimized parts are not fully pipelined with the reconstruction. These overheads slow down the overall reconstruction rate to 6-8 frames per second at 256×256 resolution with 100 depth planes. In Figure 6, we show some disparity maps from our real-time system.

References

COLLINS, R. 1996. A Space-Sweep Approach to True Multi-Image Matching. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 358–363.

YANG, R., AND POLLEFEYS, M. 2003. Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware. In *CVPR 2003*.

YANG, R., WELCH, G., AND BISOP, G. 2002. Real-Time Consensus-Based Scene Reconstruction Using Commodity Graphics Hardware. In *Proceedings of Pacific Graphics 2002*, 225–234.

[1]The cameras are facing a white wall with little texture. So we segment the images to fill the background with different colors.

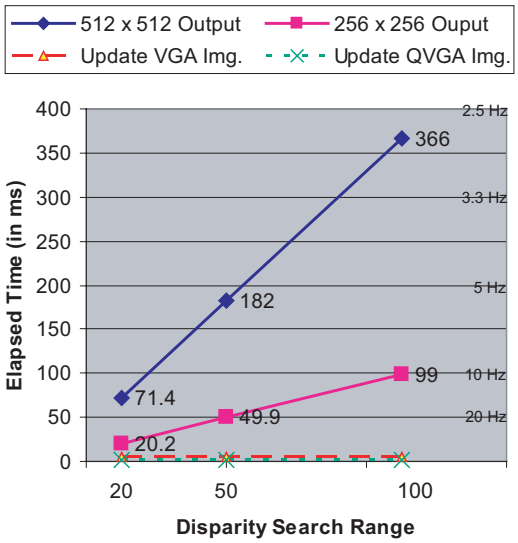

Figure 5: Performance on a NVIDIA GeForce4 Card. The data are from Table 1.

Figure 6: More results from our real-time online stereo system. The first column shows the input images; The second column shows the disparity map.