GPU Accelerated Realtime Stereo for Augmented Reality

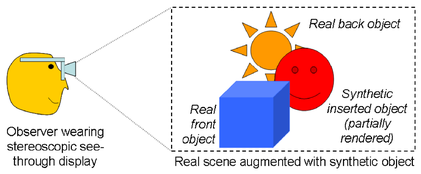
Mikhail Sizintsev [1,2] Sujit Kuthirummal [1] Harpreet S. Sawhney [1]
Supun Samarasekera [1] Rakesh Kumar [1] Ali Chaundhry [1]

[1] *Sarnoff Corporation, Princeton, NJ, USA*

[2] *Department of Computer Science and Engineering, and Centre for Vision Research, York University, Toronto, Ontario, Canada*

Introduction

Augmented Reality requires dense depth maps for correct rendering of virtual objects that respects occlusions with real objects.

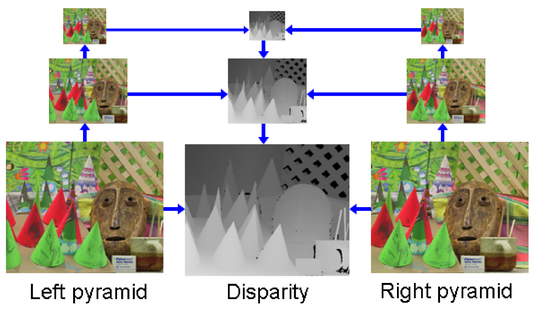


Binocular stereo is the cheapest, robust, and most reliable technique for acquiring real time dense depth maps in unconstrained (non-lab like) scenarios.
Since scene depth has one-to-one correspondence to stereo disparity, we refer to the latter as our major working quantity.

Contributions

• We present a high frame rate stereo system - 32 fps for 640x480 video with 255 disparity values.
• The crispness of the occlusion boundaries and the smoothness of the disparity maps are comparable to those produced by global non-realtime stereo algorithms.
• Our algorithm is very parallelizable and so we exploit the parallel processing capabilities of today's GPUs to get dense depth maps at high frame rates.
• We demonstrate how the proposed algorithm's accuracy can be used for Augmented Reality - to insert virtual characters into videos while respecting occlusions with real scene objects

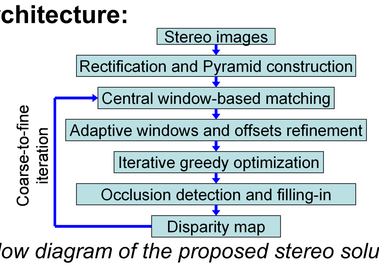
Outline of the Approach



Coarse-to-fine stereo processing: Image pyramids are created from left and right images and disparities are processed in a coarse-to-fine fashion (according to blue arrows)

The stereo solution is built upon the adaptive coarse-to-fine (pyramid based) stereo framework proposed in [1], which has the following advantages::
• Fast execution times since all disparity values are not explicitly evaluated.
• Larger disparities are estimated via smaller search at coarser levels.
• Emulates multi-resolution matching windows -- the same small window size at coarser pyramid levels emulates matching with a larger support at the input image resolution.
• Uses non-centered windows for matching and adaptive upsampling of coarse-level disparity estimates, yielding good performance at occlusion boundaries.

The Proposed Architecture:



Flow diagram of the proposed stereo solution

To this basic algorithm we have added the following:
• An iterative greedy disparity optimization step performed at every pyramid level that minimizes a cost function similar to the objective functions used in MRF stereo. This minimizes propagation of disparity errors from coarser to finer levels.
• Occlusion and bad matches detection are identified and filled it at every pyramid level.

Algorithm and Implementation

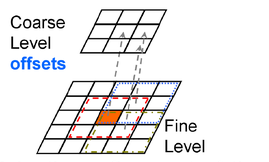
Our algorithm consists of a sequence of parallel computations. Hence, we implemented it on Graphics Processing Units (GPUs) which today have hundreds of parallel cores. We used Compute Unified Device Architecture (CUDA) available on nVidia GPUs [2].

Rectification and Pyramid Construction:
Calibration, done offline, is used to generate rectification warps for the two views. At run time these warps are used to compute rectified stereo views using texture mapping. Gaussian pyramids for the images are constructed by using separable convolution and subsampling.

Adaptive Coarse-to-Fine Block Matching:
ACTF, proposed by [1] is a straightforward extension of correlation based window matching. The differences are that the window used need not be centered at the reference pixel, and the disparities from the coarser level are adaptively chosen. ACTF can be realized as follows (details are in [1]):

a). Compute correlation using centered windows and for every pixel record the maximum correlation with the corresponding disparity. $CC = \frac{\sum_i L_i \times R_i}{\sqrt{\sum_i L_i^2 \sum_i R_i^2}}$

b). For every pixel, find the pixel with the largest correlation score within a neighborhood, and select its corresponding disparity. This is efficiently implemented using two passes of a 1-D max filter.



Adaptive refinement of the disparity at a pixel using estimates from the coarser level – offsets. The best support window (e.g. 3x3) for the center pixel (orange) can be a non-centered window (dashed squares).

Local Greedy Disparity Optimization:

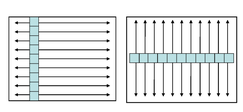
Example of the ACTF failure and its recovery using local greedy optimization

To improve the performance of ACTF and minimize the propagation of errors from coarser to finer levels, at each pyramid level, we refine the estimated disparities using an optimization that minimizes a MRF-stereo like cost function:el.

$$TotalCost(d_i) = CC(d_i) + \lambda\, SC(d_i) \qquad SC(d_i) = \sum_q min(|d_i - D(q)|, \alpha)$$

DC is the data cost defined as *1-CC*. *SC* is the smoothness cost which measures the compatibility of disparity hypothesis *di* with disparities at its immediate neighbors.
We pick the disparity that minimizes TotalCost independently at every pixel in greedy fashion. This can be done in parallel and hence is fast. Typically, only two iterations of this optimization are needed at every level to achieve significant improvement over ACTF.

Occlusion Detection and Extrapolation:
Occlusions and bad matches are detected using Left-Right Check These pixels are filled in with the disparity of their closest background pixel. A heuristic that works well is to traverse from such a pixel outward along the vertical and horizontal directions to find the first non-occluded pixel in each direction. The minimum of the disparities of these 4 pixels is used as value for filling in.



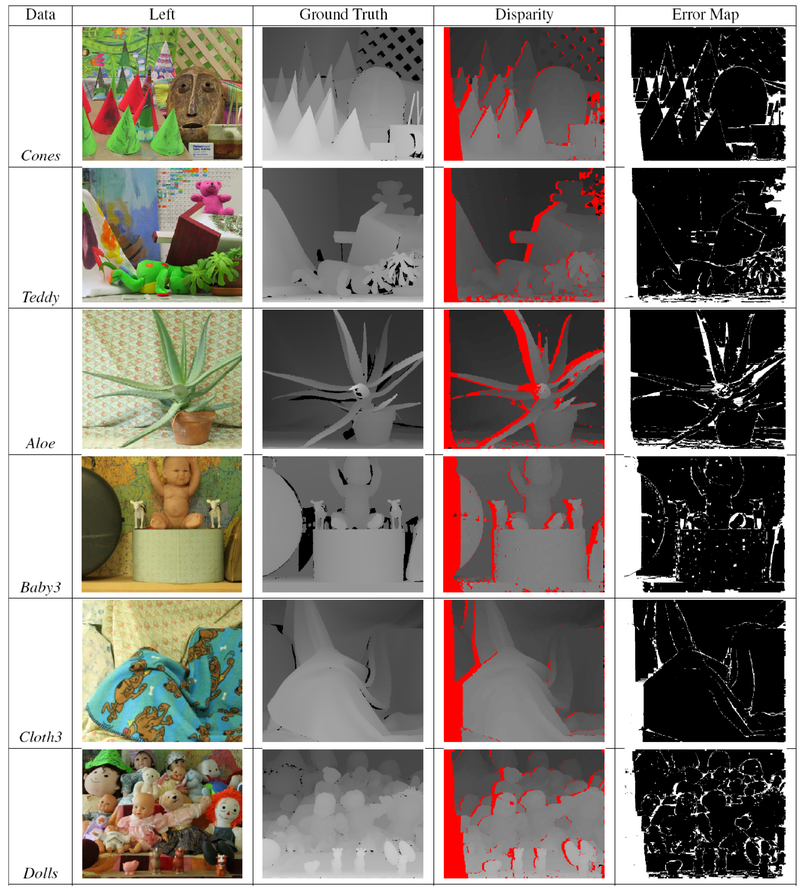
Vertical and horizontal traversal for extrapolation

Realtime Performance

	Image Size	Disparities	Speed (fps)	GPU
Our method (ACTF only)	640 × 480	256	113	GeForce GTX 285 (240 cores)
Our method (no filling-in)	640 × 480	256	45	GeForce GTX 285 (240 cores)
Our method (full)	640 × 480	256	32	GeForce GTX 285 (240 cores)
Census [5]	640 × 480	50	75	GeForce GTX 280 (240 cores)
Our method (ACTF only)	640 × 480	256	48	Quadro 3700M (128 cores)
Our method (no filling-in)	640 × 480	256	21	Quadro 3700M (128 cores)
Our method (full)	640 × 480	256	18	Quadro 3700M (128 cores)
Semi-Global Matching [8]	640 × 480	128	4.2	GeForce 8800 Ultra (128 cores)
BitVote [7]	450 × 375	64	12	GeForce 8800 GTX (128 cores)
Our method (no filling-in)	320 × 240	128	88	GeForce GTX 285 (240 cores)
emi-Global Matching [8]	320 × 240	64	13	GeForce 8800 Ultra (128 cores)
BitVote [7]	320 × 240	16	87	GeForce 8800 GTX (128 cores)
Census [5]	320 × 240	120	110	GeForce GTX 280 (240 cores)
Our method (no filling-in)	800 × 600	512	29	GeForce GTX 285 (240 cores)
Our method (full)	800 × 600	512	20	GeForce GTX 285 (240 cores)
Census [5]	800 × 600	120	22	GeForce GTX 280 (240 cores)

Comparison of the execution speeds of the proposed algorithm with other real time algorithms for different image resolutions and disparity search ranges.

Stereo Experiments



Results on the Middlebury subset [3]. In disparity maps, red pixels are detected occlusions. In error maps, pixels with disparity error greater than 1 are white.

Algorithm	Teddy			Cones		
	nocc	all	disc	nocc	all	disc
DP [4]	7.23	14.4	17.6	6.41	13.7	16.5
BP [6]	8.72	13.2	17.2	4.61	11.6	12.4
BitVote [7]	9.90	15.0	19.5	6.66	12.3	13.4
Proposed	7.85	15.8	19.0	4.86	14.1	13.0

Quantitative comparison of some of the realtime algorithms on Teddy and Cones. The percentage of pixels whose estimated disparities differ from ground truth by more than 1 in non-occluded regions, entire image and near discontinuities.

Augmented Reality Experiments

References

[1] M. Sizintsev and R. P. Wildes "Coarse-to-fine stereo vision with accurate 3D boundaries", 28(3):352–366, 2010
[2] nVidia CUDA. http://www.nvidia.com/object/cuda_home.html
[3] Middlebury College Stereo Vision Page, http://www.middlebury.edu/stereo/
[4] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister. High-quality real-time stereo using adaptive cost aggregation and dynamic programming.In 3DPVT, pages 798–805, 2006.
[5] M. Weber, M. Humenberger, and W. Kubinger. A very fast census based stereo matching implementation on a graphics processing unit. In ICCVW, 2009.
[6] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister. Realtime global stereo matching using hierarchical belief propagation. In BMVC, pages 989–998, 2006
[7] K. Zhang, J. Lu, G. Lafruit, R. Lauwereins, and L. Van Gool. Realtime accurate stereo with bitwise fast voting on CUDA. In ICCVW, 2009
[8] I. Ernst and H. Hirschmuller. Mutual information based semi-global stereo matching on the GPU, In ISVC 1:228-239, 2008