

Real-Time View Synthesis Using Commodity Graphics Hardware

Ruigang Yang, Greg Welch, Gary Bishop, Herman Towles *
Department of Computer Science, University of North Carolina at Chapel Hill

1 Our Real-Time View Synthesis Method

We present a novel use of commodity graphics hardware that effectively combines a plane-sweeping algorithm [Collins 1996] and view synthesis in a single step for real-time, on-line 3D view synthesis. Unlike typical stereo algorithms that use image-based metrics to estimate *depths*, we focus on using image-based metrics to directly estimate *images*. Using real-time imagery from a few calibrated cameras, our method can generate new images from nearby viewpoints, without any prior geometric information or requiring any user interaction, in real time and on line.

For a desired new view C_n , we discretize the 3D space into a number of candidate focal planes $\{D_i\}$ parallel to the image plane C_n of the desired view. These candidate planes discretize each desired view ray into a finite set of sample points. We step through the set of candidate focal planes, looking for the sample point along each view ray that offers the maximum color consistency among input images. Each final image pixel is determined by the color at this point of maximum consistency.

To accomplish this, for each candidate plane D_i we project (texture) the input images onto that plane. We then render the resulting textured plane onto the image plane of C_n to get an image (I_i) of D_i . We combine these two operations into a single homography (planar-to-planar) transformation. In the first row of Figure 1, we show a number of images from different planes. Note that each of these images contains the projections from all input images, and the area corresponding to the intersection of objects and the correct candidate focal plane remains sharp. For each pixel location (u,v) in I_i , we compute the mean and Sum of Squared Difference (SSD) score. The final color of (u,v) is the color with minimum SSD score in $\{I_i\}$.

We have discovered a novel use of graphics hardware to carry out the *entire* computation on the graphics board. Modern graphic cards, such as NVIDIA's GeForce series, provide a programmable means for per-pixel fragment coloring through the use of *register combiners* [Kilgard 2000]. We exploit this programmability, together with the texture mapping functions, to carry out the entire computation on the graphics board.

2 Hardware Acceleration

In our hardware-accelerated renderer, we step through the candidate focal planes from near to far. At each step (i), there are two stages of operations, *scoring* and *selection*. In the *scoring* stage, we set up the transformation according to the new view point. We then project the reference images onto the plane D_i . The textured D_i is rendered into the image plane (the frame buffer). In this stage, the *Pixel Shader* is configured to compute the SSD score and the frame buffer acts as an accumulation buffer to keep the mean color (in the RGB channel) and the SSD score (in the alpha channel) for D_i . In the second row of Figure 1, we show the SSD score images (the alpha channel of the frame buffer) at different steps.

In the next *selection* stage, we need to select the mean color with the smallest SSD score. The content of the frame buffer is copied to a temporary texture (T_{work}), while another texture (T_{frame}) holds the mean color and minimum SSD score from the previous step. These two textures are rendered again into the frame buffer through

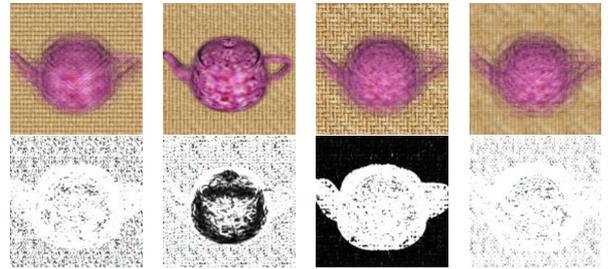


Figure 1: Mean color (above) and SSD scores encoded in the alpha channel (below) for different candidate focal planes in the first *scoring* stage. The scene consists of a teapot and a textured planar back wall.

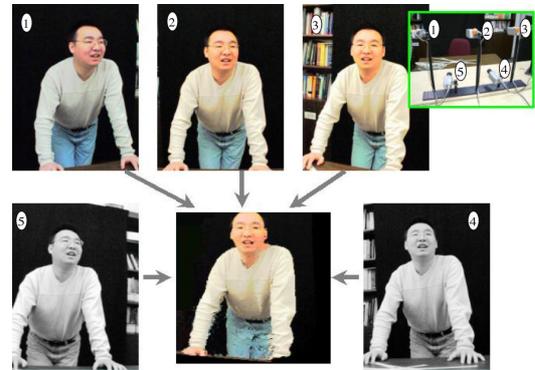


Figure 2: Example setup for 15 frame-per-second on-line reconstruction using five cameras.

orthogonal projection. We reconfigure the *Pixel Shader* to compare the alpha values on a per pixel basis, the output color is selected from the one with the minimum alpha (SSD) value. Finally the updated frame buffer's content is copied to T_{frame} for use in the next step.

One major advantage of our method is that once the input images are transferred to the texture memory, all the computations are performed on the graphics board. There is no expensive copy between the host memory and the graphics board, and the host CPU is essentially idle except for executing a few OpenGL commands.

3 Preliminary Results

We have implemented and tested our method with an NVIDIA GeForce3 graphics card. A typical reconstruction (256x256 output resolution with 50 candidate planes) takes less than 70 ms. Figure 2 shows our setup and a sample result. More compelling, interactive results can be found in the accompanying videotape. To better illustrate the results we include some live red-blue stereo sequences on the video.

References

- COLLINS, R. T. 1996. A Space-Sweep Approach to True Multi-image Matching. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 358–363.
- KILGARD, M. J. 2000. A Practical and Robust Bump-mapping Technique for Today's GPUs. In *Game Developers Conference 2000*.

*E-mail: {ryang, welch, gb, towles}@cs.unc.edu