Hierarchical Upsampling for Fast Image-Based Depth Estimation

Blake Foster and Rui Wang*
University of Massachusetts Amherst

Figure 1: *Reconstruction from 2 input images. The initial point cloud is obtained from running Bundler [Snavely et al. 2006], resulting in depth estimates for ∼0.1% of the pixels in each image within 30 seconds. Our upsampling algorithm then produces a dense reconstruction containing more than 1 million points in less than a second. The three images on the right show the dense point cloud after reconstruction.*

1 Introduction

While many stereo vision algorithms can quickly and robustly estimate sparse geometry from sets of photos, *dense reconstruction*, where depth estimate is required at per-pixel or sub-pixel level, remains a time-consuming and memory-intensive process. In this work, we propose a fast hierarchical upsampling method for dense image-based depth estimation. The main idea is to start from sparse depth estimates that can be quickly computed using any existing multiview stereopsis tool, then iteratively upsample the depth values to obtain a dense reconstruction consisting of millions of points. Using a GPU-based implementation, the upsampling algorithm can perform up to 15 images per second. The results can be directly used for 3D modeling applications; in addition, they can be used to digitally manipulate the depth-of-field effects in the input images in order to simulate refocusing.

2 Our Approach

Given a set of high-resolution images, we first run multiview stereopsis on downsampled images (at ∼300 pixels in either dimension) to quickly obtain a sparse point cloud. One possibility is to run the Bundler software [Snavely et al. 2006], which computes structure-from-motion to recover camera parameters, while simultaneously outputting a sparse point cloud representing the scene geometry.

Next, we combine the high-resolution color images with the sparse point cloud to obtain a dense reconstruction. This is achieved using an algorithm inspired by joint bilateral upsampling [Kopf et al. 2007]. To begin, we select a set of reference views R from the input images. For each R, we compute a sparse depth map D_s by projecting the sparse 3D points to the view. We then use joint bilateral upsampling to estimate a depth value for every pixel in R. Specifically, the depth value of an unknown pixel p is computed as a linear sum of its valid neighbor pixels (i.e. pixels with known depth values), weighted by both the spatial and color similarities. The intuition is that pixels with similar colors in a local neighborhood are likely to have similar depth values. The upsampling equation is:

$$D_u(p) = \frac{1}{k_p} \sum_{q \in \Omega} D_s(q) f(\|p-q\|)\, g(\|R(p) - D_s^{rgb}(q)\|) \quad (1)$$

where p is a target pixel, f is the spatial kernel, g is the range kernel, Ω is the spatial support, q is a pixel in Ω, $\frac{1}{k_p}$ is the normalization factor, D_u and D_s are the upsampled and the sparse depth maps respectively, R is the color image, and D_s^{rgb} is the color of q stored in the sparse depth map.

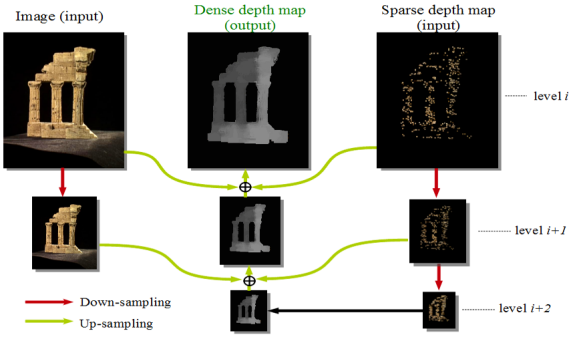

Figure 2: *Illustration of hierarchical upsampling. At each level i, the reconstructed depth map at level D^{i+1} is upsampled under the guidance of image R^i to fill in the missing depth values in D^i.*

For high-resolution inputs, the spatial neighbor size Ω typically has to be large. To improve the algorithm efficiency, we adopt a hierarchical approach. Figure 2 illustrates the process. First, we generate image pyramids for both the image R and the sparse depth map D. For R we downsample k levels using a 2×2 box filter; and for D we downsample $k+1$ levels using a min filter. Next, the algorithm goes through k iterations from the bottom level. During each iteration, we use Equation 1 to upsample the depth values from D^{i+1} to D^i, guided by the pixel colors in R^i. For the newly interpolated pixels, we use R^i to initialize their colors, which are further stored in the depth map. The algorithm then proceeds to the next level.

Results We have implemented our algorithm on modern GPUs. On an NVIDIA GTX 280, our implementation can process up to 15 images per second at 640×480 resolution. Figure 1 shows an example of reconstruction from 2 input images. The results can be used directly for 3D modeling applications. In addition, the dense depth maps recovered can be used to for digital refocusing. Refer to the supplemental video for demos of both.

References

KOPF, J., COHEN, M. F., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM Trans. Graph. 26*, 3.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph. 25*, 3, 835–846.

*e-mail: {blfoster, ruiwang}@cs.umass.edu