Compressing Texture Maps for Large Real Environments

Yizhou Yu, Andras Ferencz, and **Jitendra Malik**

Computer Science Division, University of California at Berkeley

We are concerned with texture-mapping recovered geometry with photographs. The photographs overlap, and typically a point in the geometry is covered by multiple photographs. The geometry is represented as a triangular mesh which may be recovered either from photographs or from laser range data. From photographs that have been aligned with the geometry, we wish to manufacture specialized texture maps, images broken up into triangular patches (see Color Plate 1). The naive approach would allocate a fixed sized triangle in the texture map for each triangle in the mesh. However this would be very costly, both in terms of memory and rendering speed. In this abstract, we suggest two ways of reducing the number of texture maps.
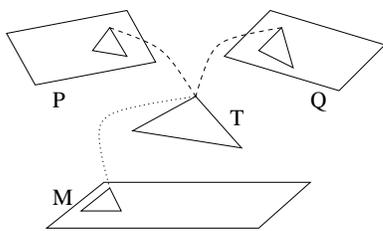

Figure 1: **Texture Map Synthesis** Assume 3D triangle T is covered by two photographs P and Q. We allocate a triangular texture patch in texture map image M corresponding to T, and populate it with a weighted average of the projected areas of T in photographs P and Q.

Our standard texture map creation scheme is as follows. Since each triangle in a mesh may be covered by multiple photographs, we actually synthesize one texture patch (a triangular shaped region) for each triangle to remove the redundancy. This texture patch is an appropriately weighted average of the projected areas of the triangle in all photographs(Fig. 1). We use the scheme in [1] to place the synthetic triangular texture patches into texture maps, and therefore obtain texture coordinates. The size of each triangular texture patch is determined by the maximum number of pixels mapped onto it from any image. Thus, all else being equal, bigger triangles in the mesh will be allocated a bigger patch.

We run into problems with this approach, because it generates too many and too big triangles. Graphics hardware has a limited amount of texture memory (2MByte on our SGI O_2 machine, forcing the maximum size of the texture map image to be 1024x512). If we cannot pack all texture information into that amount of space, we have to compose multiple texture maps and swap texture memory multiple times for each frame. Swapping texture memory is quite expensive (our SGI O_2 needs 0.04 second to load a 1024x512 image).

Texture Patch Resizing

One way that the above texture patch allocation method is wasteful is that it picks sizes for texture patches without considering the amount of variation in the photographs. However, we need fewer texture map image pixels (texels) to represent smooth areas than to encode highly varying regions. By using an information measure on the image, we can better decide the size of each texture patch. We use the response of an edge detection operator (the derivative of the Gaussian) as our information measure, and apply it to all original photographs. For each texture patch, we use the maximum response at its corresponding pixels in the photographs to determine the number of texels it actually needs to keep the original color variations on the triangle. Thus smooth regions where the operator does not respond, are allocated few texels.

Texture Patch Clustering

Additional savings can be achieved by *reusing* texture patches for multiple 3D triangles, when the texture over these triangles look similar. For example, if the walls in a room are all white, it is possible to represent the shading variations on the walls with a small number of texture patches even if the number of triangles for the walls is quite large. This requires that we cluster the texture patches from the previous section and set the same texture coordinates to all triangles in the same cluster.

We first quantize the edge length(number of texels along each edge) of every texture patch to be a power of 2, such as 2, 4, 8, 16 and 32. Then use the K-mean algorithm (Lloyd algorithm) in vector quantization [2] to cluster all the texture patches with the same size. Because of Mach Band effect, slight color difference along the edge shared by two 3D triangles may be rather obvious. We use a larger penalty for difference on edge texels to alleviate this effect during K-mean clustering. Given an error tolerance, we need to run a binary search to find the minimum number of clusters that can achieve that error. This process is quite time-consuming since each step of the binary search needs to run the K-mean algorithm whose complexity is $O(nmd)$ where n is the number of initial vectors, m is the number of clusters, and d is the dimensionality of each vector. This complexity becomes $O(n^2d)$ when m is a large fraction of n. We found out that a two-level scheme can optimize the performance by first grouping the n vectors into $\sqrt{n}$ clusters and then running the binary search on the vectors belonging to each cluster which in turn splits into multiple clusters.

We have applied our texture map synthesis and texture patch resizing techniques to 45 high resolution images of a large interior scene and obtained 20 1024 by 512 texture maps. Then we used our texture patch clustering algorithm on the 20 texture maps to obtain 5 new texture maps. The average error tolerance was set to 9 (of 256) intensity levels per pixel per color channel. The resulting compressed texture maps can still achieve good visual quality with a compression ratio of 4 (see Color Plate 2).

References

[1] SOUCY, M., GODIN, G., AND RIOUX, M. A texture-mapping approach for the compression of colored 3D triangulation. In *Visual Computer* (1996), vol.12, pp. 503–514.

[2] GERSHO, A., AND GRAY, R.M. Vector quantization and signal compression Boston : Kluwer Academic Publishers, 1992.

[3] YIZHOU YU, Modeling and Editing Real Scenes with Image-based Techniques. Ph.D. thesis, Computer Science Division, UC Berkeley, 2000.

Compressing Texture Maps for Large Real Environments

Plate 1. An example of synthesized texture map packed with triangular texture patches

(a)

(b)

(c)

(d)

Plate 2. A comparison (a)-(b) Two synthetic images of a real room rendered with the original set of 20 texture maps; (c)-(d) two synthetic images rendered from the same viewpoints with the compressed set of 5 texture maps. The two pairs of images look similar.