Computational Vision: Final Project Proposal

Matthew McMullan and Ian Ooi

Due November 26, 2012

1 Problem

Texturing 3D models can be problematic, especially when large models are used and much detail is required in the texture. We encounter this frequently with objects such as the ground and walls. Such objects require expansive textures, which are often too large to store practically on the GPU. Some solutions include tiled textures and streaming large textures from disk. Tiled textures can produce artifacts such as seams, or can be very noticeably repeated, especially in the case of stochastic and irregular textures. Textures that stream from disk are slow, resource intensive, and time consuming to make.

2 Our Solution

Our goal is to synthesize high definition textures using small samples, following the example-based route of texture synthesis. We will implement the algorithm from Parallel Controllable Texture Synthesis. This algorithm uses an example patch to create samples of a theoretically infinite, deterministically synthesized texture. This process starts with optimized segments containing features that are calculated in a pre-computation step. It uses a shifted Gaussian image pyramid, coordinate upsampling, and correction subpasses to synthesize a texture. They also use a coarse feature map to allow a user to place features from the sample image in the resulting image. This can be done exactly, or with allowed variation. There is variation at large and small scales in the generated texture.

In the time allotted for this project, we hope to get a non real-time implementation of the synthesis algorithm working, though we hope to implement some of the user controls. Specifically we will setup the basic algorithm, performing upsampling, jitter and correction (altering the jittered coordinate to match neighborhoods similar to the example patch) using a conventional Gaussian pyramid structure to represent the example patch. We then plan to implement a Gaussian image stack, as specified in the paper. The implementation will be on the GPU using Cg shaders, which we will then use in the Unity 3D game engine to produce our results.

3 Source of Data

Our sample input is primarily the canonical base images used in the texture synthesis field. We will also be using some self-taken images that we believe will highlight the particular strengths and weaknesses of this approach. A folder containing our samples is included with this report.

4 Papers of Importance

Parallel Controllable Texture Synthesis
https://research.microsoft.com/en-us/um/people/hoppe/paratexsyn.pdf

Synthesizing Natural Textures
https://www.cs.utah.edu/ michael/ts/ts.pdf