Anisotropic Diffusion of Height Field Data using Multigrid Solver on GPU

Won-Ki Jeong Tolga Tasdizen Ross Whitaker

School of Computing, University of Utah

1 Anisotropic Diffusion of Height Field Data

Let $h(x,y)$ and $\mathbf{N}(x,y)$ be the height field and its unit length surface normal vectors, respectively. For feature preserving diffusion of h , we use the two step algorithm described in [Tasdizen and Whitaker 2003]:

Step 1. Anisotropic Diffusion of Face Normals : Anisotropic diffusion of a given vector field $\mathbf{N}$ can be implemented by the following system of non-linear PDE

$$\frac{\partial \mathbf{N}}{\partial t} = \nabla \cdot (c\nabla \mathbf{N}).$$

The conductance value, c, is large for areas of the vector field with low curvature and small for areas with high curvature. This smoothes the vector field while preserving sharp edges. This PDE can be linearized and solved using semi-implicit integration. Stepping forward in time by an amount Δt we get

$$(I - \Delta t L(c))\mathbf{N}' = \mathbf{N} \qquad (1)$$

where $L(c)$ is the linear operator with fixed conductances.

Step 2. Mesh Refitting to Normals : We minimize the following energy to refit the height field $h(x,y)$ to the normal field $\mathbf{N}'$ resulting from equation 1:

$$\int (\mathbf{M}\cdot\mathbf{M})^{\frac{1}{2}} - (\mathbf{M}\cdot\mathbf{N}')dxdy$$

where $\mathbf{M}$ is the normal vector of the current height field h.

2 GPU Multigrid Solver for Anisotropic PDEs

Multigrid approaches are widely used for solving various PDE problems [Briggs et al. 2000]. However, previous GPU multigrid solvers [Goodnight et al. 2003] cannot be used directly to solve the equation 1 due to the spatially varying c term. We solve this problem by pre-computing the conductance values on the finest grid and restricting these values to the coarse grids. The initial conductance values are computed as follows:

$$c(e) = \exp^{-\left(\frac{(1-n_1\cdot n_2)}{k}\right)^2}$$

where n_1 and n_2 are face normals adjacent to the edge e and k is a diffusion constant. Once conductance values are computed, they are stored and reused in later V-cycles. Pseudo code for the main algorithm for multigrid solver and a recursive definition of V-cycle are as follows:

```
// main algorithm. h : input height field data
1.  c <- conductance(h)
2.  x <- h
3.  while( |h - L(c)x| > threshold )
        x <- v-cycle(x,h,c); // repeat V-cycle until converges

// module : v-cycle(x,h,c)
1.  x <- relax(x,h)
2.  if(current level > coarsest level)
3.      r <- h - L(c)x
4.      r <- restrict(r) // now r is 1-level lower
5.      c <- restrict(c) // now c is 1-level lower
6.      x' <- v-cycle(0,r,c)  // initial guess 0
7.      x' <- interpolate(x') // now x' is 1-level higher
8.      x <- x + x'
9.  x <- relax(x,h)
10. return x
```

We employed full-weighting (using 8 neighbors) and injection schemes for restriction/interpolation [Briggs et al. 2000]. For conductance restriction, we developed a circuit-like downsampling method for faster convergence (about 2x).

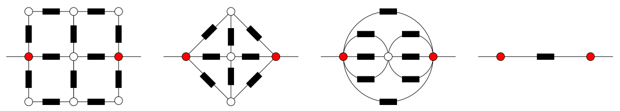
Figure 1: Conductance restriction using circuit-sampling. Left : fine level grid, Middle two : star circuit to delta circuit, Right : coarse level grid

All computations are done on a single 32bit float pixel buffer with multiple surfaces (double buffers + four AUX buffers) to avoid GL context switch overhead. All shader programs are written in ARB vertex/fragment assembly language. FRONT buffer contains results of relaxation, AUX0 contains residuals, and AUX1 contains conductance values. BACK and AUX2 buffers are used as temporary buffers. Several surfaces can be bound at the same time as multiple textures, and any bound surface cannot be used for rendering target.

3 Results

A single multigrid V-cycle on 513x513 dataset takes about 0.2 sec on Nvidia FX 6800 Ultra graphics card. The multigrid solver converges after seven V-cycles and takes about 3.5 sec including the refitting step for the figure 3 ($\Delta t = 100$, $k = 0.05$), about 10x speeding up compared with the CPU implementation [Tasdizen and Whitaker 2003].

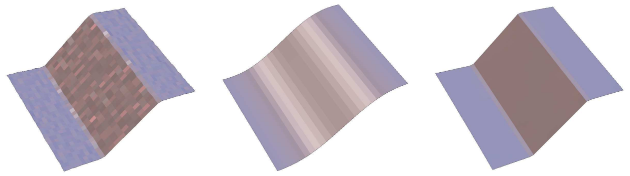
Figure 2: Comparison of smoothing results. Our method (right) preserves sharp creases after smoothing. Left : input noisy mesh, Center : isotropic smoothing, Right : anisotropic smoothing

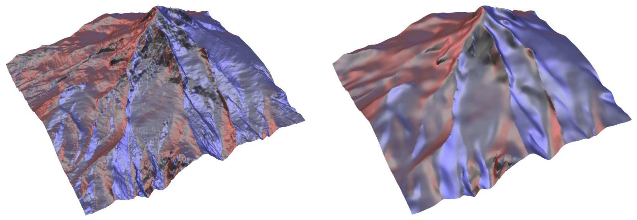
Figure 3: Feature-Preserving Smoothing of Height Field Data. Left : input height field data (513x513), Right : result of anisotropic smoothing

References

BRIGGS, W., HENSON, V., AND MCCORMICK, S. 2000. *A Multigrid Tutorial*. SIAM.

GOODNIGHT, N., WOOLLEY, C., LEWIN, G., LUEBKE, D., AND HUMPHREYS, G. 2003. A multigrid solver for boundary value problems using programmable graphics hardware. In *Proceedings of Graphics Hardware*, 1–11.

TASDIZEN, T., AND WHITAKER, R. 2003. Feature preserving variational smoothing of terrain data. In *Proceedings of IEEE Workshop on Variational, Geometric and LevelSet Methods in Computer Vision*.