Abstract

Computer tomography (CT) has wide application in medical imaging and reverse engineering. Due to the limited number of projections used in reconstructing the volume, the resulting 3D data is typically noisy. Contouring such data, for surface extraction, yields surfaces with localised artifacts of complex topology. To avoid such artifacts, we propose a method for feature-preserving smoothing of CT data. The smoothing is based on anisotropic diffusion, with a diffusion tensor designed to smooth noise up to a given scale, while preserving features. We compute these diffusion kernels from the directional histograms of gradients around each voxel, using a fast GPU implementation.

Keywords: Feature-preserving smoothing, anisotropic diffusion

1 Introduction

Computer tomography is an effective way of digitally scanning objects with complex and intricate geometry. Example applications include medical imaging and reverse-engineering of closed objects in a non-destructive acquisition process. Volumetric data from inverse tomography are prone to artifacts due to imperfectly acquired projections. These imperfections have multiple causes: high contrasts; lack of captor sensitivity; too few projections; non-monochromaticity of the X-Ray source, imperfect stability of the X-Ray source; pixel defects on the captor; non-uniformity of the absorption process across wavelengths.

A popular approach to smooth such 3D data is to use geometric diffusion. Geometric diffusion smoothes orthogonal to the gradient in the volume and therefore preserves planar regions, while blurring edges and corners. More general anisotropy has also been proposed where the flatness of the diffusion kernel is adapted to the gradient of the volume [Schaap et al. 2008], or the structure tensor [Frangakis and Hegerl 2001]. In all these cases, the diffusion kernel is radially symmetric (two eigenvalues are equal).

We propose a new method of computing a diffusion tensor for each voxel based on the distribution of gradients in its neighborhood. We build a radially asymmetric diffusion tensor by combining the geometric diffusion tensors of the gradients in the neighborhood of each voxel. Two input parameters control the extent of smoothing: the size of the neighborhood and the number of diffusion steps. We propose an efficient method to compute these tensors on GPU.

2 Multi-Scale feature-preserving diffusion

We need our diffusion kernel to respect the constraints of the individual geometric-diffusion kernels for each gradient in the neighborhood of the current voxel. Thus, we use the geometric average of the geometric diffusion kernels. To avoid discretization artifacts, we compute the continuous directional histogram [Kass and Solomon 2010] $h_s$ of gradients $\nabla_f(x)$ of the volumetric data $f$ around each point $x$ in the volume using a Von Mises kernel $K$. The value of the histogram for a given direction is obtained by integrating over gradients of nearby voxels with a gaussian weight $g_\omega$ variances $\alpha$:

$$h_s(\omega) = \frac{1}{K} \int_{\mathcal{V}} K(\nabla_f(y), \omega) g_\omega(x-y) dy = \frac{1}{K} K(\nabla_f(\cdot), \omega) \otimes g_\omega$$

where $\alpha$ is the input feature size, $K$ is the normalizing constant that accounts for the gaussian $g_\omega$ and the Von Mises kernel $K$. Thus, we compute $h(\omega)$ for the entire volume at once along each histogram direction $\omega$ using only two 3D FFTs. For a direction $\omega$, the geometric diffusion kernel is a 3D gaussian $e^{-\alpha \omega M_\omega x}$ where $M_\omega$ is a 3D symmetric matrix. To build a diffusion kernel that accounts for the entire histogram of directions around voxel $x$ we compute the geometric average of gaussian filters weighted by the histogram for each direction. The matrix $M_\omega$ of this kernel is therefore the integral of matrices $M_\omega$ for all directions weighted by the gradient histogram in this direction:

$$M_\omega = \frac{1}{H} \int_\Omega h_\omega(\omega) M_\omega(\omega) d\omega$$

Because we compute $h(\omega)$ for all voxels at once, we can numerically compute $M_\omega$ for all voxels $x$ by sweeping across a finite set of directions $\omega$. The diffusion then uses the three eigenvectors $v_i$ and eigenvalues $\lambda_i$ of $M_\omega$, and the Hessian $H$ of the volume:

$$\frac{\partial v_i}{\partial t} = \sum_{i=1,2,3} \lambda_i \left( v_i H v_i \right)$$

The memory cost of computing the diffusion kernel per voxel is that of storing the symmetric matrix $M_i$ plus the normalizing constant $H$, i.e. only 7 floats per voxel. We can therefore treat volumes up to 512³ voxels in 4GB of memory. It is possible to compute the diffusion kernels block per block as well, with a proper overlap between the blocks. We used our method to improve tomographic data (See Fig. 1). Our method is fast, as we are able to compute kernels in a 256³ volume in 700 seconds for 642 directions using a 10³ feature size (while the same calculation needs hours on CPUs). Each diffusion step then takes less than 5 sec. Our technique can be extended to repair surfaces as well, while eliminating artifacts of arbitrary topology up to a given scale, if by first converting the mesh into 3D level set data.

References

