CPS 533 Scientific Visualization

Wensheng Shen

Department of Computational Science
SUNY Brockport
Chapter 10: Image Processing

- This chapter describes the image processing components of VTK. The focus is on key representational ideas, pipeline issues such as data streaming, and useful algorithms for improving the appearance and effectiveness of image data visualization.

- First, the data flow or pipeline approach in vtk is applicable to image processing. Second, image processing algorithms can improve the results of visualization.
10.1 Data representation

- A dataset consists of both a structure and data attributes. An image is defined to include four dimensions: three spatial dimensions x, y, and z, and time t. The reason to add the time dimension is that images are frequently generated as a time series, and we often wish to access the data along the time axis. For example, we may plot the value at a point as a function of time.

- An image has both regular topology and geometry. The regularity of the data makes it possible for many special operations, such as data caching and streaming, operating on regions of interest in the data.
A mapper may need only a region of the data for its display, so loading or processing the whole dataset would be inefficient. A viewer displays only one slice of a large structured volume. By loading slices only as they are needed, disk access can be reduced, and the memory conserved. A an image stored in a Cartesian coordinate system is easily divided into smaller rectangular regions, while data sampled on a polar coordinate grid is best divided into pie-shaped regions. Operating on regions of data means that we process “window” of data specified by (min, max) range of each dimension, or axis. For example, a region in a 2D image of dimensions 100×100 might be specified as (25, 49, 0, 49), meaning that we would operate on a (25×25) window.

Region of interest
Streaming and caching

The disadvantage of processing regions of interest is that the same data may be read and processed multiple times. If the viewer needs to loop through the slices, it would be beneficial to have all the data loaded at once. A compromise between the two extreme approaches of maintaining all data in memory or operating on small pieces is to update regions larger than requested, but not as large as the whole image. This is referred to as a data cache.

With the region-processing model, the data objects can be thought of as caches that hold any number of regions. One strategy of data caching is to save only a single region at any one time. If subsequent requests are completely contained in the cached region, no further processing is required. An alternative strategy might divide an image into tiled regions of all the same size. When a region larger than the tile is requested, multiple tiles are updated to cover the region.

Given the ability to operate on regions of data, it is a small step to stream operations on a whole dataset. Streaming is the process of pulling regions of data in a continual flow through the pipeline. For instance, a pixel histogram mapper could request single pixel as it accumulates values in its bins.
Attribute data and components

Visualization algorithms may generate normals, vectors, tensors, and texture coordinates, image processing algorithms generally process attribute data consisting of scalar data. Often the data is a single component (e.g., a gray-scale image), but frequently color images (three components of RGB, for example) may also be processed.
10.2 Algorithms

- Removing noise
- Smoothing
- Reducing sampling artifacts
- Image enhancement
- Segmentation
- Morphological operators such as erosion and dilation
Image restoration

The first step in image processing is often restoration, which is used to remove artifacts with minimum impact on the underlying data. For example, most of the power of typical images lie in low frequencies, while white noise is evenly distributed across the frequency spectrum. In this situation, low-pass filters eliminates much of the noise, but leave most of the image intact.

Los-pass smoothing filters can be implemented using convolution with a kernel with all positive values. The typical kernels used for smoothing are either constant across a circular neighborhood, or have a Gaussian profile. Gaussian smoothing results in better-looking images than smoothing with constant kernels, but can be computationally expensive.

One way to speed Gaussian smoothing is to decompose the filter into two 1D convolution, since the 2D Gaussian function is separable

\[
g(i, j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2 + j^2}{2\sigma^2}\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{i^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{j^2}{2\sigma^2}\right)
\]

Smoothing along the x axis and then along the y axis with 1D Gaussian kernels is equivalent to convolving with a 2D Gaussian kernel.
Nonlinear smoothing

A common problem with simple smoothing to remove noise is that edges are blurred. One approach to smoothing that preserves edges is anisotropic diffusion. This filter smooths relatively flat regions of an image, but does not diffuse across abrupt transition. The diffusion is iterated until the desired level of noise reduction is reached. Some diffusion criteria are: diffuse only when the gradient magnitude is below a specified value, or diffuse two pixels only when the difference between the pixels is lower than a specified constant.

A median filter also smooths while preserving edges. This filter replaces each pixel with the median value of the scalar values in a neighborhood centered on the pixel. Median filters are most effective on high amplitude noise that has a low probability of occurring. There are two ways to control the amount and scale of noise removed: the size of the neighborhood can be varied, or the filter can be applied multiple times. This median filter preserves edges, but it does round corners and remove lines.
The hybrid median filter was developed to address this behavior. It operates on a 5×5 neighborhood around each pixel. The algorithm consists of two steps: first the median values of an “x” shaped and “+” shaped neighborhoods are computed, then the median of these two values and the center-pixel value is computed to give the final result. The hybrid median has a fixed size neighborhood, but can be applied multiple times to further reduce noise.
When we do subsampling, an aliasing artifacts occurs. Aliasing artifacts are often associated with stair-stepping edges. When a signal is subsampled, its capacity to hold high frequency information is reduced. However, the high frequency energy does not disappear. It wraps around the frequency spectrum appearing as a low frequency alias artifacts. The solution to this is to conduct low-pass filtering before subsampling. Low-pass smoothing reduces the high frequency range of an image that would cause aliasing.
Image enhancement

Often datasets contain information or have dynamic range that cannot be completely displayed in a single image. X-Ray Computed Tomography (CT) datasets can have 10 times the scalar resolution of the typical computer monitor capable of displaying 256 shades of gray. One method used for displaying information buried in the large dynamic range of these medical datasets is to allow a user to interactively set the color map with a window-level transfer function.

Images should have a uniform distribution of intensities. For continuous images, this intensity distribution is called the probability density function (PDF). For continuous images, the transfer function is simply the cumulative distribution function (CDF) which is defined as the integral of the PDF. By definition, the CDF function has a large slop where the PDF has the largest value, and therefore gives the greatest contrast to scalar ranges that occur most frequently in an image. The result of using the CDF as a transfer function is an image with an ideal constant scalar distribution. For discrete images and image histograms, a discrete version of the CDF function can be use.
Frequency domain

The Fourier transform belongs to a class of filter that fundamentally change the representation of an image without change its information. The output of the Fourier transform is in the frequency domain. Each pixel is a complex number describing the contribution of a sinusoidal function to the original image. The magnitude of the pixel encodes the amplitude of the sinusoid, and the orientation of the complex pixel encodes the sinusoid’s phase. Each pixel represents a sinusoid with different orientation and frequency. The reverse Fourier transform converts a frequency domain image back to the original spatial domain. Low-pass and high-pass filtering become trivial in the frequency domain. A portion of the pixels are simply masked or attenuated.

In general, convolution with large kernels is more efficient when performed in the frequency domain. Multiplication, $\alpha \beta$, in the frequency domain, is equivalent to convolution, $a*b$, in the spatial domain (and vice versa). In these equations, $\alpha$ is the Fourier transform of $a$, and $\beta$ is the Fourier transform of $b$. 
Fast Fourier transform

The Fourier transform is decomposable, so a 2D transform can be implemented by first taking the 1D Fourier transform of all the rows, and then taking the Fourier transform of all the columns of an image.

The complexity of one-dimensional Fourier transforms can be reduced with the fast Fourier Transform (FFT) algorithm. It works by recursively factoring the number samples, N, into its prime components. If N is prime and not factorable, then the transform is completed in one step that is order \(O(N^2)\) complexity. If N is divisible by two, the array of numbers is divided into two parts that are transformed separately and then combined. If N is a power of two, then the algorithm executes in order of \((\log N)\) time. For this reason, it is more efficient to process images with sizes that are powers of two (e.g., 512×512) than other sized images. For non-power of two images, it may be faster to pad the image to a size that is a power to two size before processing.
$H(u, v) = \begin{cases} 1 & \text{if } u^2 + v^2 < C^2 \\ 0 & \text{otherwise} \end{cases}$

$H(u, v) = \frac{1}{1 + \left[\frac{C^{2n}}{u^2 + v^2} \right]^n}$

Ideal high-pass filter

Butterworth high-pass filter

These are the two high-pass filters in the frequency domain. The butterworth high-pass filter has a gradual attenuation that avoids ringing produced by the ideal high-pass filter with an abrupt transition.
Image segmentation

Segmentation is the process of classifying pixels in an image or volume. It can be one of the most difficult tasks in the visualization process. One form of segmentation takes an image as input, and outputs a map that contains a classification for each pixel. The output of such a segmentation filter usually has binary or discrete values for each pixel.

A simple example of a one-parameter segmentation is a threshold filter used to mark bone in a CT dataset. Since bone has the largest scalar value, it is easy to select a threshold that separate bone from the rest of the image. For other tissues and other imaging modalities, segmentation is usually more difficult. Noise in the image and overlapping scalar values of tissues can decrease the effectiveness of simple threshold segmentation. By using two parameters, the threshold can segment pixels with a range of scalar values. The extra parameter allows more control over the resulting segmentation, but also doubles the complexity of selecting the parameters.

Images can be preprocessed to segment images based on more complex features such as textures. Sometimes textures in tissues add information useful for segmentation. Texture sensitive filters like Laplacian and gradient magnitude can discriminate between different textures. Additional filters than can be used for texture segmentation are the range, variance, and correlation filters.
A correlation filter is similar to convolution. The kernel is shifted across the image, and for each location the dot product between the image and the kernel gives a measure of correlation between the two. The output of the correlation filter is large everywhere the pattern occurs in the image, but small at other locations.
Postprocessing

Erosion is implemented by removing pixels within a specified distance of a border. For each pixel not in the segmented region, all the neighbors in a circular region around the pixels are turned off. This erosion filter shrinks the segmented region and small isolated regions disappear.

Dilation is the process opposite to erosion. The dilation filter grows the area of segmented regions. Small holes in the segmented region are completely closed. Any pixel not in the segmented region by near the region is turned on.

Closing is the serial application of first dilation and then erosion. When an image is dilated small holes in the map disappear. However, dilation alone also grows the boundaries of the segmented regions. When dilation is followed by erosion in a closing operation, small holes are removed; however the boundary of the segmented regions remain in the same general location.

Opening is opposite to closing. Opening removes small islands of pixels. It is implemented with an initial erosion, followed by a dilation.

Connectivity filter can be used to remove small regions without affecting the remaining boundaries of segmented regions. This set of filters separate the segmented pixels into equivalent classes based on a neighbor relation. Two pixels belong to the same class if they are touching. There are two common neighbor relations in two-dimensional images: four connectivity considers pixels neighbors if they are edge neighbors, and eight connectivity considers pixels neighbors if pixels share any vertex.
Multispectral segmentation

Color images contain more information than gray-scale images, because each pixel contains more information in the red, blue, and green components than a single component gray-scale pixel. Multispectral images can be segmented by separating the RGB components and threshold them individually and then combine the resulting binary images with logic filters.

It is possible to transform the components into a different coordinate system before the threshold operation. This can be done by thresholding a projection of the components. This divides the component space into two areas separated by a hyper-plane. Another example of a coordinate transformation is conversion from RGB color components to hue, saturation, value (HSV) representation. Segmentation of images based on hue and color saturation is difficult in RGB space, but trivial in HSV space.

We can perform multispectral segmentation on an one component dataset, by creating images from spatial information of the images using spatial filters. These multiple images can then be combined into one multicomponent image, then multispectral segmentation can be proceeded.
10.3 Implementation in *vtk*

**vtkReferenceCount**

**vtkImageRegion**

**vtkImageData**

**vtkImageCache**

*vtkImageData* actually represents the image data, and it is an instance of *vtkScalars*, so it could be unsigned char, char, unsigned short, short, int, float. *vtkImageData* is referred to by *vtkImageCache* and *vtkImageRegion*.

*vtkImageRegion* is an object that refers to a region of data within a *vtkImageData* object, and allows axes to be reordered so that data can be processed in a different order.

*vtkImageCache* supports the caching of data. In the imaging pipeline, *vtkImageCache* serves as input and output to the imaging filters.

Data representation
vtkImageCache *image = vtkImageCache::New();
vtkImageRegion *region;
Float *ptr;
image->SetWholeExtent(0, 255, 0, 255, 0, 0, 0, 0);
image->SetUpdateExtent(0, 255, 0, 255, 0, 0, 0, 0);
image->SetScalarTypeFloat();
region = image->GetScalarRegion();

For(j=0; j<256; j++)
{
    y = j/10.0;
    for(i=0; i<256; i++)
    {
        x=i/10.0;
        ptr = (float *)(region->GetScalarPointer(i, j, 0, 0);
        *ptr = 128.0*(sin(x) + sin(y));
    }
}
Region->Delete;

Create an image of two interfering sinusoidal gratings in an image using vtkImageCache
An imaging pipeline to visualize gradient information
Combining the imaging and visualization pipelines to deform an image in the z-direction