Mathematical Statistician and Engineering Applications
ISSN:2094-0343
2326-9865

MRI and CT Scan Data Based Volume Rendering with Python

A .V. Krishna Rao Padyala [1], Dr. K.Ramkumar [2]
Research Scholar, Department of CSE
SRM University, Sonepat, Delhi-NCR, Haryana
Professor and Associate Dean (E&T)
SRM University, Sonepat, Delhi-NCR,Haryana

Article Info
Page Number: 42 - 52
Publication Issue:
Vol 72 No. 2 (2023)

Article History
Article Received: *15 February 2023*
Revised: *20 April 2023*
Accepted:*10 May 2023*

Abstract

To diagnose a variety of conditions, from torn ligaments to tumors, doctors make use of MRI and CT scans. An application developed using python, installed in a mobile or tablet render the data of MRI and CT scans and let the doctor understand patient health condition. A method for transforming MRI/CT scan data in DICOM form to HDF5 with python. Variety of tools available now to develop android applications for mobile with python.

Keywords: - volume rendering; Medical image data;Direct volume rendering ; ray casting; DICOM to HDF5.

Introduction

Volumetric medical image rendering is a method of extracting meaningful information from a three-dimensional (3D) dataset, allowing disease processes and anatomy to be better understood, both by radiologists as well as physicians and surgeons. Three principle volume-rendering algorithms have been evolved for medical image visualization, i.e., multiplanar reformation (MPR), surface rendering (SR), and volume rendering (VR). Direct rendering includes direct volume rendering (DVR) and direct surface rendering (DSR). Indirect rendering includes indirect surface rendering (ISR).

Multiplanar Reformationis an image processing technique, which extracts two-dimensional (2D) slices from a 3D volume using arbitrarily positioned orthogonal or oblique planes. The observer can display a structure of interest in any desired plane within the data set, and four-dimensional (4D) MPR can be performed in real time using graphics hardware. In addition, to accurately visualize tubular structures such as blood vessels, curved MPR[1] may be employed to sample a given artery along a predefined curved anatomic plane. curved MPR was combined with DVR to enhance the visualization of tubular structures[2] such as trachea and colon.

Surface rendering is a common method of displaying 3D images. DSR, the surfaces are rendered directlyfrom the volume without intermediate geometric representations, setting thresholds or using object labels to define a range of voxel intensities to be viewed.This is often applied to contrast-enhanced CT data for displaying skeletal and vascular structures[3],

and is also usually used in describing vascular disease and dislocations. In the process of detecting acute ischemic stroke, maximum intensity projection (MIP) with shaded surface rendering to visualize proximal middle cerebral artery mainstem occlusion[4].

Direct Volume Rendering displays the entire 3D dataset as a 2D image, without computing any intermediate geometry representations. This algorithm can be further divided into image-space DVR, such as software and GPU-based raycasting and object-space DVR, such as splatting shell rendering, texture mapping (TM), and cell projection. Shear-warp can be considered as a combination of these two categories. In addition, MIP, minimum intensity projection (MinIP), and X-ray projection are also widely used methods for displaying 3D medical images.Volume-rendering pipeline and Numerical Operations used in it are listed below.

- Pre-Processing
- Transformation
- Classification (Color and Opacity mapping)
- Sampling (Interpolation)
- Shading/ lighting (Gradient Computation)
- Compositing (Alpha blending)
- Rendered Image display

To display 3D medical images with DVR, the scalar values must first be mapped to optical properties such as color and opacity through a transfer function (TF), a process referred to as classification. Pre- and post-classification approaches differ with respect to the order in which the TF and sampling interpolation are applied.

Pre-classification suppresses this high-frequency information, so the rendered image appears blurry. post-classification maintains all the high frequencies, but introduces “striping” artifacts in the final images.In DVR, the TF was typically used for tissue classification based on local intensities in the 3D dataset.During the DVR process, a number of composition schemes are commonly employed, including X-ray projection, MIP, MinIP, and alpha blending. In X-rayprojection, the interpolated samples are simply summed, giving rise to an image typical of those obtained in projective diagnostic imaging.

MIP and MinIP are widely used techniques in 3D CT and MR angiography. MIP and MinIP images can be generated rapidly and can clearly display vessels, tumor, or bones.MIPs or MinIPs are not particularly useful due to lack of depth information. Alpha blending is a popular optical blending technique, often implemented by using the Riemann sum to discretize the continuous function, resulting front-to-back and back-to-front alpha blending respectively, depending on the compositing order.

In volume illumination, the normal at every sampling point is calculated by interpolation using the intensity changes across that voxel. These approximated voxel normals are then used in a Phong or Blinn–Phong model for shading computations, with the results being employed in the DVR composition. However, clinicians sometimes deliberately disable illumination since it may complicate certain diagnostic tasks.

Literature Survey

Software based Raycasting[5] is a popular technique used to display a 3D dataset in two dimensions, whose basic idea is to cast a ray from each pixel in the viewing plane into the volume, sampling the ray with a predetermined step in a front-to-back or back-to-front order by trilinear interpolation.

Splatting[6] calculates the influence of each voxel in the volume upon multiple pixels in the output image. volume as an array of overlapping basis functions called reconstruction kernels, which are commonly rotationally symmetric Gaussian functions with amplitudes scaled by the voxel values. These kernels are classified as colors and opacities by a TF, and are projected onto the output image plane in a depth sorted order. Each projected kernel leaves a footprint (or splat) on the screen, and all of these splats are composed in back-to-front order to yield the final image.

Shear-warp[7] is a hybrid algorithm that attempts to combine the advantages of the image- or object order based volume-rendering methods. The shear matrix transforms all the viewing rays so that they are parallel to the principal viewing axis in "sheared object" space, allowing the volume and image to be traversed simultaneously. short-comings of this algorithm, such as blurry images on zoom and staircase artifacts on flat surfaces.

The shell rendering algorithm[8] is a software based hybrid of surface and volume rendering, which is based on a compact data structure referred to as a shell. The shell data structure can store the entire 3D scene or only the hard (binary) boundary. For the hard boundary, the shell is crisp and only contains the voxels on the object surface, and shell rendering degenerates to SR. For a fuzzy boundary, the shell includes some voxels in the vicinity of the extracted surface, and shell rendering[9] is identified as DVR.

In **2D texture mapping (2DTM)**, the volume is decomposed into three stacks of perpendicularly object-aligned polygons. During rasterization, each of the polygonal slices is textured with the image information obtained from the volume via bilinear interpolation. Finally, the textured slices are alpha blended in a back-to-front order to produce the final image.

Three-dimensional texture mapping (3DTM)[9] uploads the volume to the graphics memory as a single 3D texture, and a set of polygons perpendicular to the viewing direction is placed within the volume and textured with the image information by trilinear interpolation.

Applications of TM-based medical volume rendering, divided into three main applications. The first is multimodal medical image rendering and tissue separation.The second application

area is the display of specific tissues and organs for diagnosis and therapy. The third area of application is in dynamic imaging and deformation.

The GPU on today's commodity video cards has become a powerful computation engine capable of a variety of applications, especially in the fields of computer graphics and visualization. **GPU-accelerated raycasting**[10] can provide an interactive frame rate for medium-sized dataset visualization on commodity hardware.GPUs incorporate multi-pipelined stream processors and high bandwidth memory access, and include their own DRAM (device memory) optimized to process 2D and 3D geometries in parallel. Data are transferred between the CPU main memory and the GPU graphics memory via a dedicated Accelerated Graphics Port or a Peripheral Component Interconnect Express (PCI-Express) bus. Modern GPUs are programmable and employ a SWAR (SIMD, i.e., Single Instruction Stream Multiple Data Stream, Within A Register) execution model.

Programmable GPU for raycasting, the main core of this algorithm is to generate a single ray per screen pixel and to trace this ray through the volume on a GPU fragment shader.

The algorithm is described by the following items:

- Store the dataset as a 3D texture,
- Render the front face of the color cube and store the result into an intermediate texture
- Render the back face, subtract the color values of the back face from the front face, normalize the result, and store it in a 2D texture, which represents the direction vectors of the casting ray for each pixel. The origin of the ray is the pixel of the bounding cube front face.
- Cast a ray for each pixel into the 3D texture volume, uniformly sample the volume along the ray.
- Blend the TF-mapped color and opacity at each sampling point along the ray in a front-to-back order.

Fourier Domain Volume Rendering algorithm goal is to compute 2D Fourier projections of 3D data, reducing the computational complexity that exists in spatial domain.Levoy and Malzbender independently applied this theorem to DVR, while Napel et al. proposed an application specific to MR angiography. Additionally, Jansen et al. performed Fourier domain volume rendering (FDVR) directly on the graphics hardware by using the split-stream fast Fourier transform (FFT) to map the recursive structure of the FFT to the GPU.While GPU-accelerated FDVR is efficient, this algorithm has three drawbacks. First, it has limited ability for 3D texture handling; secondly, two copies of the data must be present in the graphics memory, one as the source and the other one as the destination array; finally, it is not possible to implement any local (image-space) operations in the Fourier domain, eliminating the possibility of incorporating lighting models.

Volume Rendering with Python

Python has been recognized as one of the most popular languages in recent years. Some of the advantages of the Python framework for android app development are

- Faster Programming Performance-Unlike other programming languages, python programs are directly executed by the interpreter.
- Test-Driven Development Compatibility-Python makes prototype creation for applications effortless. It fully supports prototype development, and through refactoring them, allows you to build applications directly from the prototypes.
- Variety of Libraries- Android app development using Python allows you to decide on modules from its extensive range of effective and robust libraries as per your requirements without much coding.

volume renderings to visualize 3D simulation data cubes. This technique is incredibly useful when you have space-filling data you would like to visualize. Such data shows up often in astrophysical datasets but also in other areas of computer graphics and medicaldata (CT scans & MRIs). The rendering algorithm run on data and that may look like in figure below.

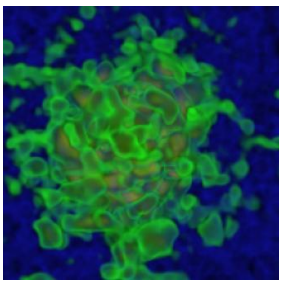

Figure 1: Data to be rendered with python

Data is an Nx × Ny × Nz datacube of a density that is to be visualized. Define a custom transfer function that will return a red, green, blue, and opacity value (r,g,b,a) as a function of input density. The volume rendering algorithm, in a sense, creates a 3D iso-contour plot[11]. A sketch of what a transfer function may look like is shown below. In this example, two contour values are picked out.

Transfer function that picks out certain density values by placing narrow Gaussians at x=3, x=1, and x=-1. Each Gaussian has its own peak opacity a, and colors r,g,b.

```
def transferFunction(x):

r = 1.0 * np.exp( -(x - 3.0)**2/1.0 ) +  0.1*np.exp( -(x – 1.0)**2/0.1 ) +  0.1*np.exp( -(x - -1.0)**2/0.5 )

g = 1.0*np.exp( -(x - 3.0)**2/1.0 ) +  1.0*np.exp( -(x - 1.0)**2/0.1 ) +  0.1*np.exp( -(x - -1.0)**2/0.5 )

b = 0.1*np.exp( -(x - 3.0)**2/1.0 ) +  0.1*np.exp( -(x - 1.0)**2/0.1 ) +  1.0*np.exp( -(x - -1.0)**2/0.5 )
```

```
a = 0.6* np.exp( -(x - 3.0)**2/1.0 )+ 0.1*np.exp( -(x - 1.0)**2/0.1 ) + 0.01*np.exp( -(x- -1.0)**2/0.5 )

return r,g,b,a
```

Density values on a datacube are called as the camera grid. create an R,G,B image of the rendering. initialize each color channel to 0. Each pixel in the image represents a ray being cast from the back of the volume to the front, that will pass though the density field and alter its color until it reaches the front, yielding the final image. At each step, the transfer function returns values r,b,g,a, and the image is updated as

R = a*r+(1-a)*R,

G= a*r+(1-a)*G,

B= a*r+(1-a)*B.

The following few lines of code perform this

```
image=np.zeros((camera_grid.shape[1], camera_grid.shape[2], 3))

for dataslice in camera_grid:

    r,g,b,a = transferFunction(np.log(dataslice))

    image[:,:,0] = a*r + (1-a)*image[:,:,0]

    image[:,:,1] = a*g + (1-a)*image[:,:,1]

    image[:,:,2] = a*b + (1-a)*image[:,:,2]
```

Interpolating Datacube onto the Camera Grid- visualize the original datacube as is, in which case set these things equal (camera_grid = datacube). But to rotate the camera view of the rendering, which can help to visualize the dataset better and more easily pick out 3D structure. In this case, one needs to interpolate the original datacube onto the camera grid that is aligned with the angle at which dataset to be viewed. This is not too difficult to do with SciPy's interpn function, which define two sets of coordinates and interpolate from one to the other. The following snippet of code does this.

cameragrid.py

```
# Load the Original Datacube

f = h5.File('datacube.hdf5', 'r')

datacube = np.array(f['density'])

# Construct the Corresponding Datacube Grid Coordinates

Nx, Ny, Nz = datacube.shape

x = np.linspace(-Nx/2, Nx/2, Nx)
```

```
y = np.linspace(-Ny/2, Ny/2, Ny)

z = np.linspace(-Nz/2, Nz/2, Nz)

points = (x, y, z)

# Construct the Camera Grid / Query Points -- rotate camera view

angle = np.pi/3.0  # my viewing angle

N = 180  # camera grid resolution

c = np.linspace(-N/2, N/2, N)

qx, qy, qz = np.meshgrid(c,c,c)  # query points

# apply rotation

qxR = qx

qyR = qy * np.cos(angle) - qz * np.sin(angle)

qzR = qy * np.sin(angle) + qz * np.cos(angle)

qi= np.array([qxR.ravel(),qyR.ravel(), qzR.ravel()]).T

# Interpolate onto Camera Grid

camera_grid=interpn(points,datacube,qi,

method='linear').reshape((N,N,N))
```

Running the entire code allows creation of a volume rendering of a simulation datacube and compare it against a simple projection. Notice how the volume rendering picks out more structure that gets washed out in the projection:

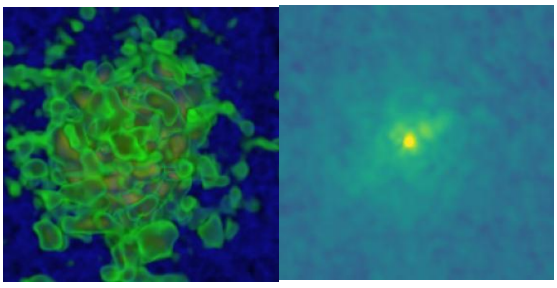

Figure 2: Volume rendering(left), Projection (right)

Computational astrophysicists use this type of visualization quite frequently to render interstellar/intergalactic gas, galaxies, and other systems. A very nice Python package for advanced visualization that recommend is the YT-Project.The volume rendering technique has applications far beyond astrophysics. volume rendering of the human skull with a CT scan created using the same algorithm shown in figure 3.

Figure 3 human skull with python

Medical Image Data Rendering

CT and MRI images are acquired in a special digital format, called the DICOM (Digital Imaging and Communications in Medicine) format. DICOM ensures that the high quality of the images is retained. Each CT or MRI scan contains multiple images in the DICOM format that need to be stored in a safe and secure manner.

To store such a large volume of medical images, each hospital usually has a PACS server. PACS (Picture Archiving and Communication System) is a central server on which images are stored, and from which they can be retrieved when needed. Usually, hospitals have an on-site, standalone PACS, and invest plenty of money into upgrading the storage capacity of the PACS when it gets full. Back-ups can come at a higher cost.

Post DICOM's cloud-based PACS solutions offer convenient, off-site storage for DICOM images. Because DICOM files are hosted on the internet, they are safe from data loss and can be accessed from any device. Cloud-based PACS have three layers of security, so patient data remains confidential.

Hierarchical Data Format (HDF) is a set of file formats (HDF4, HDF5) designed to store and organize large amounts of data. Originally developed at the U.S. National Center for Supercomputing Applications, it is supported by The HDF Group.

The Process used to render the 3D volume with mobile app is shown in the figure 4.

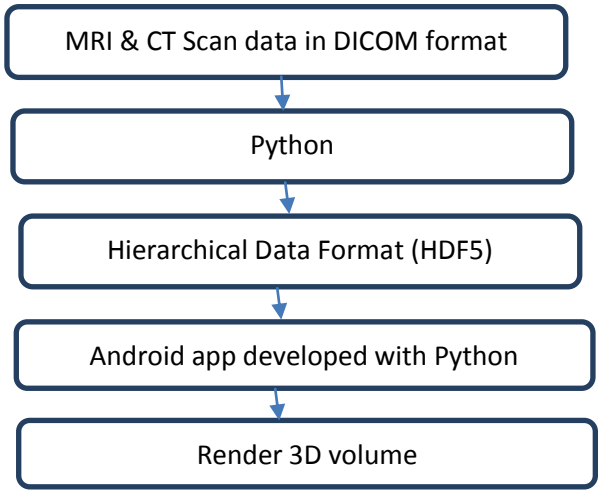

Figure 4: 3D volume rendering with mobile app developed using python

The following code shows export of pixel array from DICOM file to HDF5.

```
import dicom

import h5py

def export_pixel_array (in_file, out_file,

dataset_name="data")

pixel_array=dicom.read_file(in_file).pixel_array

h5 = h5py.File(out_file)

h5.create_dataset(dataset_name, data=pixel_array)

h5.close()
```

Now the data in hdf5 format can be rendered with procedure mentioned in the previous section.

Tools for developing android app with python

The Tools for android app development with python includes KIVY, BEEWARE, SL4A (Scripting Layer for Android), Pyqtdeploy, Chaquopy.

KIVY- Android app development using Python has been made possible only because of an open-source Python library for developing mobile applications and other multi-touch application software that is Kivy.Kivy is an open-source Python library for the swift development of cross-platform UI applications. It allows the developer to build python GUIs across different platforms like Windows, Linux, iOS, Android. In addition, it has a graphics engine developed over OpenGL, so it can manage GPU-bound workloads when needed. Kivy has a tailored UI toolkit that offers text stickers, text entry forms, buttons, etc. So, the tools are not characterized using the native user interface controls, but the tool certifies your mobile application's reliability and flexibility from one platform to another. Kivy has a python-to-android project that allows you to port Python applications to Android. It has a similar toolkit for iOS but can only be used with Python 3.4.

BEEWARE-BeeWare is another popular development framework that permits you to write applications in Python and compile them for cross-platform deployment on different operating systems, including Windows, MacOS, Linux GTK, and mobile platforms like Android and iOS development. BeeWare uses a Native UI toolkit, you can build an excellent Native look and feel UI on a Cross-platform. BeeWare tool suite is BSD licensed and accessible to anyone for use and modification.

SL4A (Scripting Layer for Android) - SL4A, originally named ASE (Android Scripting Environment), is a facade set that exposes a greatly-simplified Android API subset. SL4A brings scripting languages to Android by editing and running interactive scripts and interpreters straight on the Android device. These scripts have many APIs to complement Android apps, but a significantly simplified interface makes it more accessible.

Pyqtdeploy - Pyqtdeploy is a PyQt application deployment tool. It supports desktop platforms like Linux, Windows, and OS X and mobile platforms like iOS and Android. Pyqtdeploy functions by taking individual modules from a PyQt application, freezing them, and placing them in a Qt resource file converted to C++ code by Qt's RCC tool. Likewise, the standard Python library is supported.

Chaquopy- Chaquopy is a plugin for Android Studio's Gradle-based build system. Chaquopy allows you to easily intermix Java and Python in your app, using whichever language is suitable for your needs for Python development for Android platform. With the Python API, you can write code for android app development using Python. If you use Android Studio, you can start using Chaquopy in 5 minutes without changing your existing development process.

Conclusion and future work

As the capabilities of programmable GPUs continue to increase, the distinction between algorithms that are appropriate for the GPU and those written for the CPU will disappear. The evolution[12] from software-based raycasting to GPU based raycasting is a good example. For DVR algorithms such as splatting, shell rendering, and shear-warp, hardware accelerations have been proposed and implemented; however, most of these improvements are based on fixed graphics pipelines and do not take full advantages of the programmable features of GPU.

Python is language with rich library and the functions in it are very much useful for volume visualization. In this paper presented conversion of MRI data in DICOM format to HDF5 format, then rendering it with python and the tools that are available for developing android apps with python. Mobile device should be equipped with special hardware to capture images and for generating special kind of rays during this process.

As the present system used for capturing and maintaining medical image data is very costly, introduction mobile device and searching for other alternatives for data management is very much essential.

References

[1] van Ooijen PMA, Ho KY, Dorgelo J, Oudkerk M: "Coronary artery imaging with multidetector CT: visualization issues. Radiographics" 23: e16, 2003

[2] Kim HC, Yang DM, Jin W, Park SJ: "Added diagnostic value of multiplanar reformation of multidetector CT data in patients with suspected appendicitis". Radiographics 28:393– 405, 2008

[3] Park SH, Choi EK, et al: "Linear polyp measurement at CT colonography: 3D endoluminal measurement with optimized surface-rendering threshold value and automated measurement". Radiology 246:157–167, 2008

[4] Schellinger PD, Richter G, Köhrmann M, Dörfler A: "Noninvasive angiography (magnetic resonance and computed tomography) in the diagnosis of ischemic cerebrovascular disease". Cerebrovasc Dis 24(1):16–23, 2007

[5] Drebin R, Carpenter L, Hanrahan P: "Volume rendering". In: Proceedings SIGGRAPH88, 1988, pp 65–74

[6] Lee RK, Ihm I: "On enhancing the speed of splatting using both object- and image-space coherence". Graph Models 62(4):263–282, 2000

[7] Schulze JP, Niemeier R, Lang U: The perspective shear-warp algorithm in a virtual environment. In: Proc. of the conference on Visualization '01. IEEE Computer Society, 2001, pp 207–214

[8] Grevera GJ, Udupa JK, Odhner D: "An order of magnitude faster isosurface rendering in software on a pc than using dedicated, general purpose rendering hardware". IEEE Trans Vis Comput Graph 6(4):335–345, 2000

[9] Sato Y, Westin C-F, Bhalerao A, Nakajima S, Shiraga N, Tamura S, Kikinis R: Tissue classification based on 3D local intensity structures for volume rendering. IEEE Trans Vis Comput Graph 6(2):160–180, 2000

[10] Krüger J, Westermann R: "Acceleration techniques for GPU-based volume rendering". In: VIS'03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03), IEEE Computer Society, Washington, DC, USA, 2003, pp. 287–292

[11] Philip Mocz, "Create Your Own Volume Rendering (With Python)", https://medium.com/, November 22, 2020.