Real-Time Ultrasound Image Analysis for the Insight Toolkit

David Wang[1,2,3], Wilson Chang[2,3], and George Stetten[1,2,3]

[1] University of Biomedical Engineering, Carnegie Mellon University, Pittsburgh PA 15213, USA
[2] University of Pittsburgh Medical Center, Pittsburgh PA 15261, USA
[3] Department of Biomedical Engineering, University of Pittsburgh, Pittsburgh PA 15261, USA

Abstract. We have successfully created a software environment in which ultrasound data can be manipulated by, ITK (the Insight Tool-Kit), in real-time. We were able to access each frame generated within the resident computer of a Terason™ Ultrasound Machine, convert it into the ITK image format, and demonstrate the concurrent operation of ITK on the same computer by writing the images to an external hard drive. At a rate of 10 frames per second, 512 by 512 pixel grayscale frames were written by ITK methods to the external hard drive through USB 2.0 while the ultrasound scan was occurring without thrashing or delay in system performance. This simple exercise demonstrates the potential of ITK in processing ultrasound images in real-time in addition to the more traditional off-line processing.

1 Introduction

Of all medical imaging modalities, ultrasound is perhaps the one most often used when real-time imaging is required. Therefore, image analysis of ultrasound data is potentially useful when done in real-time as well. It is, then, necessary to create a software environment with convenient APIs that allow testing of various image analysis algorithms when ultrasound data arrives in real-time of at a rate of at least 10 frames per second. The Insight Tool-kit (ITK) offers a diverse collection of open source image analysis algorithms ideal for this purpose. The problem is how to get the ultrasound images into ITK in real time. Most ultrasound machines present their data to the outside world as a video signal, which can be captured on a separate computer and sent to ITK. One particular brand of ultrasound scanner, from Terason, Inc. (Burlington, MA), is based upon a standard laptop computer running Microsoft Windows™, and permits simultaneous operation of other software in such a way that the image can be shared in real time. We have successfully developed and tested the interface between Terason's real-time ultrasound software and ITK.

We will present some necessary background information on ultrasound in Section 2. We will then explain the API for ITK and the Terason software in Section 3. In

Section 4 we will review the performance of the system. Finally we will conclude with pointers to future work.

2 Ultrasound

Ultrasound is a method of imaging that permits real-time visualization of the internal anatomy of a human. Like CT and MRI, ultrasound is non-invasive, but ultrasound does not carry any radiation exposure risk and is relatively inexpensive. Because of the low risk and wide availability of ultrasound, it is being increasingly used in many areas of medicine: radiology, cardiology, women's healthcare, emergency medicine, family practice, urology, nephrology, vascular surgery, general surgery, sports medicine, ophthalmology, gastroenterology, telemedicine, veterinary medicine, etc.

Clinical ultrasound typically uses frequencies of 2 to 30 MHz. Pulses of sound energy travel from the ultrasound probe to an anatomical interface, which reflects the sound back to the probe. Most ultrasound machines make the assumption that the speed of sound passing through biological tissues is 1540m/s [1,2]. By measuring the time it takes the sound wave to travel round trip, the distance from the probe to the reflector can be calculated as:

$$\text{Distance (m)} = 1540\ \text{(m/s)} \times \tfrac{1}{2}\ \text{time (s)}\ . \qquad (\mathbf{1})$$

The intensity of the reflections from a particular ultrasound pulse as a function of time (distance) represents a given interrogation of the tissue. A series of such interrogation, sweeping through a plane, are collected for each scan. Each scan is then converted in real-time to a 2D grayscale image, through a process called scan-conversion. It is this 2D image, with its regular isotropic pixels on a rectilinear lattice, which is the usually starting place for image analysis.

3 Method

The underlying ultrasound system we used consists of the Terason SmartProbe™ and their proprietary software running on a standard laptop. The SmartProbe includes a transducer, which generates and receives ultrasound and a small electronic package for steering the ultrasound beam. The interface to the laptop include power and firewire connections.

The Terason software provides remote control capabilities for other programs on the same computer by exposing a Windows COM Automation interface. An external application enabled as a COM controller can instantiate the Terason Ultrasound application as a COM server or attach to a running instance of the Terason Ultrasound application. The COM controller can initialize and perform essential control over b-mode and color Doppler ultrasound acquisition. However, there is no built-in mecha-

nism to perform real-time image analysis. We solve the problem by modifying the software that runs the COM controller for the Terason ultrasound machine.

When a frame containing one scan-converted image of ultrasound data is generated by the Terason SmartProbe, the `OnFrameReady()` method in the Terason API is invoked. We modify this method to grab the frame from memory and place it into a local two-dimensional texture array in OpenGL format. Using OpenGL for display allows us to tap into the hardware acceleration built into most computer systems today. By linking the local two dimensional array to an OpenGL texture, we effectively tell OpenGL to obtain the image input from the two dimensional array whenever a new image is ready. We define a function, `DisplayFunction()`, which is invoked whenever OpenGL is notified that a new frame is ready and which refreshes the display. Besides displaying the image on the monitor, `DisplayFunction()` also invokes our function `outputImage()` where we pass the two-dimensional image to ITK, but other ITK functions could be added here for real-time analysis.

```
#include "itkImage.h"
#include "itkImportImageFilter.h"
#include "itkImageFileWriter.h"

class ITKCode {
public:
  typedef unsigned char BytePixelType;
  typedef itk::Image <BytePixelType, 2> ByteImageType;
  typedef itk::ImportImageFilter <BytePixelType, 2> Im-
  portFilterType;
  ImportFilterType::Pointer importFilter;

  typedef itk::ImageFileWriter< ByteImageType > Writer-
  Type;
  WriterType::Pointer writer;

  char filename[10];

  ITKCode(int x, int y) {
    importFilter = ImportFilterType::New();
    ImportFilterType::SizeType  size;
    size[0]  = x;  // size along X
    size[1]  = y;  // size along Y
    ImportFilterType::IndexType start;
    start.Fill( 0 );
    ImportFilterType::RegionType region;
    region.SetIndex( start );
    region.SetSize(  size  );
    importFilter->SetRegion( region );

    double origin[ 2 ];
    origin[0] = 0.0;    // X coordinate
    origin[1] = 0.0;    // Y coordinate
    importFilter->SetOrigin( origin );
```

```
    double spacing[ 2 ];
    spacing[0] = 1.0;    // along X direction
    spacing[1] = 1.0;    // along Y direction
    importFilter->SetSpacing( spacing );

    writer = WriterType::New();
    writer->SetInput(importFilter->GetOutput());
  }

  void outputImage(unsigned char* greyScaleTextureData,
     int x, int y, int count) {
    importFilter->SetImportPointer( (BytePixelType*)
     greyScaleTextureData, x*y, false);
    sprintf (filename, "%d.tif", count);
    writer->SetFileName(filename);
    try { writer->Update();}
    catch( itk::ExceptionObject & exp )
    { std::cerr << "Exception caught !" ;}
  }
}
```

The `greyScaleTextureData` variable is the two-dimensional array containing the image frame. The `x` and `y` variables are, respectively, the number of pixels in the x and y dimensions. Finally, the `count` variable allows sequential numbering of output files. The `outputImage()` function connects the two-dimensional array to an ITK import filter which then pipes the output to an ITK writer. For the sake of this proof-of-concept test, the writer sends the image frame, uncompressed, either to the internal hard drive or through a USB 2.0 connection to an external hard drive. In the results section, we review the performance of these two systems.

4 Results

We tested the above setup and code on a Dell Latitude™ with an Intel Pentium™ 2.4 GHz CPU and 512 MB of RAM where each image frame is set at 512 x 512 pixels.

Our initial use for ITK was simply to store the data, and as such we were limited by the rate of actually writing to the disk. Table 1 shows that at approximately 20 frames per second (0.05 seconds per image), the computer was unable to keep up with writing to the internal hard drive (0.067 seconds per image in one trial and 0.084 seconds per image in another). The situation improved by writing to an external hard drive. However, the computer was only just barely keeping up when writing to the external hard drive (0.060 and 0.047 seconds per image). The burden on the system was also demonstrated by the between-frame lag displayed by the real-time ultrasound video.

Table 1. System performance with various system parameters on a Dell Latitude with an Intel Pentium 2.4 GHz CPU and 512 MB of RAM where each image frame is set at 512 x 512 pixels. (HD=hard drive, Int.=internal, Ext.=external, Fps=frames per second).

HD	Fps	Seconds	Images	Sec/Image	Video Quality
Int.	20	3.956	47	0.084170	Lots of lag
Int.	20	16.073	239	0.067251	Lots of lag
Ext.	20	2.553	54	0.047278	Lots of lag
Ext.	20	43.793	734	0.059663	Lots of lag
Ext.	10	20.219	213	0.094925	Smooth
Ext.	10	304.398	3133	0.097159	Smooth
Ext.	10	58.404	590	0.098990	Smooth

Writing the images at approximately 10 frames per second (i.e., writing every other frame), allowed the system to output these images at a rate that did not overwhelm the system. The images were written at approximately 0.096 seconds per image, near the expected 0.10 seconds per image, and the ultrasound video was smooth.

5 Conclusion

We have successfully linked an ultrasound scanner with ITK in real-time without requiring conversion to and from a video signal. We share the data within the actual computer used by the ultrasound scanner. Although we have only used ITK here to write to disk, and as such have been limited by the transfer rate at that point, we feel that real-time image processing of ultrasound images is possible, especially if all the original data does not need to be stored. ITK appears to be a promising software architecture for such real-time image processing. In the future, we plan on using this architecture to run various image analysis and manipulation algorithms already present in ITK as well as those that we are writing ourselves.

References

1. Douglas Christensen, Ultrasound Bioinstrumentation, John Wiley and Sons, 1988.
2. J. Harness and D. Wisher, Ultrasound in Surgical Practice: Principles and Clinical Applications: 1st Ed. pp 40-60. Wiley-Liss, New York, 2001.