Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

A Program for Medical Visualization and Image Processing

Carlos A. Zaffari; Paulo Zaffari; Dario F. G. de Azevedo; Thais Russomano; Sergio Helegda; Marcio V. Figueira;

Abstract—**This article presents a software program for visualization and processing of medical images. It provides an expansible set of techniques to help extracting visual information from medical images to be used in diagnosis support and in advanced scientific investigations.**

I. INTRODUCTION

In the modern Medicine there is a growing need for non-invasive medical exams. To supply this need large investments are made in the imaging area. There are several modalities of medical images like: Magnetic Resonance (MR), Computer Tomography (CT), ultrasonography, endoscopy, angioscopy, Single Photon Emission Coupled Tomography (SPECT), Positron Emission Tomography (PET), etc.

Due to the high cost of the equipments and the specialization level required to generate and analyze diagnostics from them, they are available only in large clinical centers. In the developing countries, these centers are located in capitals and in large cities. Therefore, medical doctors working in small cities, in farming communities or even in peripheral regions of large cities do not have access to visualize or process the resulting images.

The imaging instruments generally use the DICOM standard in order to produce and archive medical images. With software developed for use in a personal computer with Pentium IV processor or compatible using Microsoft Windows 98 or later with Microsoft DirectX9C August update or later, the medical doctors will be able to visualize the medical images in their offices or even in their homes without the use of expensive visualization equipment.

Based on these needs, the VPIM software (“V”,”P”,”I”,”M” are the initial letters for Visualization and Processing of the Medical Image, in Portuguese) was developed. The VPIM is capable of showing medical images using DICOM file format and it is also capable of opening images using other file formats.

Carlos A. Zaffari is with the Pontifical University Catholic of the Rio Grande do Sul. Porto Alegre, RS, Brazil. E-mail: czaffari.ez@terra.com.br.
Paulo Zaffari was with the Federal University of the Rio Grande do Sul. Porto Alegre, RS, Brazil. E-mail: paulo@singularstudios.com
Dario F. G. de Azevedo is with the Pontifical University Catholic of the Rio Grande do Sul. Porto Alegre, RS, Brazil. E-mail: dario@pucrs.com.br
Thais Russomano is with the Pontifical University Catholic of the Rio Grande do Sul. Porto Alegre, RS, Brazil. E-mail: trussomano@hotmail.com
Sérgio Helegda is with the Pontifical University Catholic of the Rio Grande do Sul. Porto Alegre, RS, Brazil. E-mail: sergio.helegda@pucrs.com.br
Marcio V. Figueira is with the Pontifical University Catholic of the Rio Grande do Sul. Porto Alegre, RS, Brazil. E-mail: mvfpoa@gmail.com

Many acquired medical images need additional processing before one can extract further information. In order to help extracting information from the medical images, the program provides a set of tools which can be further extended through the creation of new plug-ins (DLLs).

II. DESCRIPTION

As a medical image visualization software, one of the most important features of the VPIM is it’s capability of loading and displaying image files. The supported file formats are: DICOM (dcm), Windows Bitmap (bmp), Microsof Direct Draw Surface (dds), Windows Device-independent Bitmap (dib), High Dynamic Range (hdr), Joint Photographics Experts Group (jpg), Portable Float Map (pfm), Portable Pixmap (ppm), Portable Network Graphics (png), Targa (tga), and Tagged Image File Format (tif). In order to support such formats, VPIM makes use of the Dicom Toolkit [1], LibTiff [6] and the native support for loading image files present in Microsoft DirectX 9C [5].

In order to display the image files Direct3D, the Microsoft DirectX 9C programming interface is used. When each image is loaded, it is stored in memory as a texture, an IDirect3DTexture9 object, using RGBA format, maintaining its original dimensions. A secondary display texture, containing a triangle filtered copy of the original image, is also created in order to satisfy graphic hardware needs for textures. The display of each texture is performed on a window where a hardware Direct3D device swap chain is rendered.

The rendering of the texture is managed using a ID3DXSprite object on a window by window basis. In the rendering process the object is attached to the current image to be displayed. The object also manages the application of a transformation matrix, a rotation center and the area of the image which should be displayed through a rectangle window.

The transformation matrix, mentioned above, describes the level of zoom, pan, mirroring, and aspect ratio of the image to the hardware. Along with the rotation center, this matrix also describes the rotation which must be applied to the image. The video controller board is responsible for the effective application of this information.

The internal organization of the images is based on a hierarchy where a study contains a set of series where each series contains a set of images. When loading DICOM files, this information is mirrored internally since it's available. When using different file formats, the image information is placed on an empty identified series of an empty identified study created for such images. Internally those images are ordered by filename, as opposed to DICOM Images which are ordered by creation time when present, by slice position when present, by instance number when present and by filename when nothing else is present.

In order to manage the display of the multiple images, which can be loaded at a time, the software has a navigation control sub-system. It's responsible for managing each image from each series from each study that is being displayed. It allows the images to be seen in multiple windows and series to be seen in the same window switched one by one manually or automatically. The automatic control can show the image series like a movie with a user defined time interval between images.

In order to visualize, enhance, and show hidden information VPIM has the processing sub-system and the measurement sub-systems.

III. The Measurement and Processing Sub-System

The measurement and processing sub-system built in VPIM was designed in order to allow rapid and simple expansion of functionalities. A set of features, which was found to be of common need for most new modules, was implemented as a built-in set of functions therefore avoiding the constant need to re-implement such techniques. This section will describe the built-in functions, how to use them, which are the existing plug-ins, how to create new plug-ins and which were the results of using the implemented techniques.

A. Built-in Functions

In addition to the geometric transformation functions described earlier in this paper, the built-in functions are: data access functions, transaction control functions and processing functions. The data access functions provide information about the current image being displayed in a given window at the time of the activation of a function. The transaction controls are responsible for processing preview, commit or transaction canceling.

The main data access method is "GetCurrentImage". It returns the IDirect3DTexture9 object being currently displayed in the selected image window, if there is one. This object allows the processing of the image data directly in order to apply other techniques, as convolution filters. Most current plug-ins use this function to access the data needed for image processing.

Frequency-domain techniques, like Butterworth filters, use a different approach in order to access data. The images are stored in the space domain. It is necessary to calculate the frequency representation of the image to apply the proper transfer function. The method "ComputeFastFourrierTransform" generates the frequency-domain representation of the image contained in the currently selected window. This conversion is made using the Aris FFT Library [2].

In order to retrieve the result of the transform, it's necessary to invoke the method "GetImageComplexRepresentation". It returns the image complex array with a real and an imaginary part.

After processing the transformed image, in order to commit the transaction, it's necessary to call the method "ComputeFastFourrierTransform", passing a parameter "true" to make a reverse transform. The method "UpdateImageFromComplexRepresentationUsingOnlyRealPart" will update the image with its new space-domain representation.

There are also others functions, like "CreateTempTexture", "CancelImageOperation" and "CommitImage". These provide transaction control. When "CreateTempTexture" is called, a copy of the current image is created and it is temporarily used by all data access and display functions. When "CancelImageOperation" is called, the copy is destroyed and the original image is then used again. If "CommitImage" is called, the copy overwrites the original image and the copy is then discarded, ending the transaction.

Among the data processing features of the program there is a palette generation sub-system. This palette sub-system allows users to associate a given intensity level interval of an image to an arbitrary color (pseudo-color). This may be used to enhance visibility of certain image features. The system also provides means to equally distribute a set of colors among the interval of all available intensities, for ease of use.

There are also other features included in the processor sub-system like: global and local remapping, histogram equalization, binary combination of images, image subtraction, background subtraction, and others.

The measurement sub-system provides support for statistic and measurement features. The features of this sub-system are: measurement of the linear distance or area (in pixels or cm or cm^2 if the data are present in the DICOM file), image histogram, profile lines of the images, statistic data (mean, standard deviation, maximum and minimum pixel value) and also the specific pixel intensity.

B. Plug-in

The plug-in system is intended to be the main VPIM expansion mechanism. The currently existing plug-ins illustrate how to create image filters and how to set-up the user interface to activate them. The currently implemented plug-ins are: Butterworth Filters, Median Filter, High-Pass Filter, Low-Pass Filter, Laplacian Filter, Sobel Filter, High-Reinforcement Filter, and Line Detection Filters.

The Butterworth filter plug-in implements four Butterworth filters: Low-Pass, High-Pass, Pass Band and Stop Band. Each of them, as described in [3], requires a frequency-domain representation in order to operate correctly. For this reason, its operation is: activation of the user interface, definition of user options such as which filter frequencies, transformation of the image into the frequency domain, application of the required enhancement process, reverse transformation, and commitment of changes to the filtered output image.

To implement the convolution plug-in filters, appropriate 3x3 convolution matrices were used. This way the moving average Low-Pass, High-Pass, high-boost, Laplacian, Line detection (vertical, horizontal, +45º, and -45º) and Sobel filters were implemented with their respective matrices [3]. Since these filters use space-domain convolution, the image has no need to be transformed. After being activated by the user interface, the convolution matrix is applied to the image. A generic 9x9 convolution matrix filter is also implemented where elements and multiplication coefficient can be user defined.

The median filter plug-in was implemented calculating by the median among the pixel itself and its 8-neighbors in the original image.

C. *How to build a plug-in*

In order to create plug-ins for VPIM, the following Windows DLL interface must be implemented:

```
class CPluginAbstractInterface
{
  public:
    typedef CPluginAbstractInterface*
(*TDEntrypoint)(CKernelAdvancedInterface*);

    static VPIMAPI CPluginAbstractInterface*
CreatePlugin(CKernelAdvancedInterface* poKernel);
};
```

The macro VPIMAPI is defined as:

```
#ifdef _USRDLL
#define VPIMAPI __declspec(dllexport)
#else
#define VPIMAPI __declspec(dllimport)
#endif
```

All terms, including CKernelAdvancedInterface, can be found in file PluginAbstract.h.

A typical plug-in starts by registering itself into the VPIM user interface. In order to do so, the plug-in must invoke the method GetWindow from CKernelAdvancedInterface, asking for the “MainWindow”. It will return a CBaseWindowAbstractInterface object. It is then necessary to retrieve the derived object CMainWindowAdvancedInterface by means of the method GetAdvancedInterface.

The programmer may then choose to call one of the two overloads of RegisterPluginCommand:

```
  const bool RegisterPluginCommand(const char*
szCommandName,CMainWindowAdvancedInterface::TDCa
llback  pfnCallback,void* poOwner);
 const bool RegisterPluginCommand(const char*
szCommandName,const char** szMenuPath,const size_t
nLevels,CMainWindowAdvancedInterface::TDCallback
pfnCallback,void* poOwner);
```

In the first overload, an entry will be added into the filter menu having the name contained in szCommandName bound with the callback function pfnCallback which when called will receive poOwner as argument. The second overload is similar to the first, but allows the menu item to be added anywhere through the parameters szMenuPath and nLevels that describe the full menu path where to place the item.

For a plug-in to have access to image data, it should use the currently selected window. In order to retrieve it, method GetSelectedImageWindow must be invoked from CMainWindowAdvancedInterface obtained earlier. Frequency-domain methods, as described earlier, may be used as well.

All plug-ins which are expected to be loaded by VPIM should be placed into a plug-in directory under the VPIM executable directory.

D. *Comparison among VPIM and others visualization softwares*

Functionalities	**eFilm [7]**	**ezDicom [8]**	**VPIM**
Read, write and visualize DICOM files	Yes	Yes	Yes
Visualization of series and studies	Yes	Yes	Yes
Visualization of all images as a movie	Yes	No	Yes
Calibration of image for measurement	Yes	No	Yes
Distance measurement between two points	Yes	No	Yes
Area measurement	Yes	No	Yes
Horizontal and vertical flipping	Yes	No	Yes
Rotation of 90º (to left and to right).	Yes	No	Yes
Free rotation	No	No	Yes
Data of the studies	Yes	Yes	Yes
FOV	Yes	No	Yes
Image inversion (negative).	Yes	Yes	Yes
Possibility of inclusion of digital filters through DLL routines	Yes	No	Yes (2)
Magnification and reduction of images	Yes	Yes	Yes
Pan	Yes	No	Yes
Adjustment of image contrast and luminosity	Yes	Yes	Yes
Pseudo-color	No	Yes(1)	Yes

Table 1: Comparison with other visualization softwares

(1) Fixed palettes.
(2) Implemented filters: convolution filters in space domain: low-pass (mobile means), high-pass, median filter, Sobel, Laplacian and line detection; filters in frequency domain: Butterworth (low-pass, high-pass, bandpass, and band-reject).
The following features are only present in the VPIM:

1. Local and global re-mapping
2. Power specter of the image

3. Windowing ($\cos^4$ and Blackman)
4. Histogram
5. Profile lines of an image
6. Image clipping
7. Global threshold
8. Convolution filter with parameters programmable by the user

E. Results

In the following figures some results are shown, using the last version of the VPIM. The original image was obtained from the file brain_001.dcm [4]. This image is shown in the Fig. 1.

Fig. 2 shows the result of the application of the local remapping over the original image. Fig. 3 shows the application of the Sobel filter.

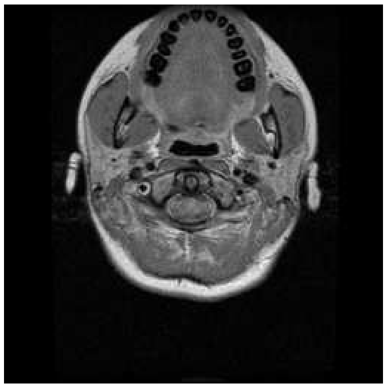

Fig. 1: Original Image

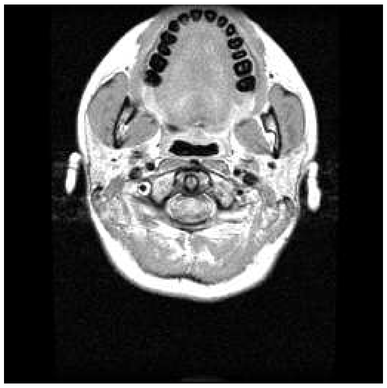

Fig. 2: Image using local remapping

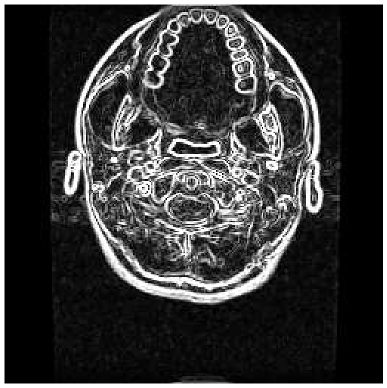

Fig. 3: Application of the Sobel filter

IV. Conclusion

The VPIM has been presented. It includes a set of tools capable to provide appropriate medical image visualization and scientific processing. The design team is developing a video monitor calibration hardware. Together with this hardware, the VPIM will provide an appropriate tool to be used by the medical doctors in the development countries, running over inexpensive PC-based platforms.

References

[1] DICOM@OFFIS, DICOM Tool kit DCMTK 3.5.3 source code and documentation. Available http://dicom.offis.de/dcmtk.php.en.

[2] FFT & DCT Library. Available: http://www.codeproject.com/library/ArisFFTDFTLibrary.asp

[3] Gonzalez, Rafael C., Woods, Richard E.. Processamento de Imagens. Editora Edgard Blücher Ltda., 2000.

[4] MATLAB CENTER. Available: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=2762&ObjectType=File

[5] Microsof DirectX Developer center. Available: http://msdn.microsoft.com/directx/

[6] TIFF Library and Utilities. Available: http://www.remotesensing.org/libtiff/

[7] Programa eFilm Workstation 1.5.3; Version 1.5.3; Built: May 9, 2001 15:40:11.

[8] Krug, Wolfgang; Rorden, Chris. Programa ezDICOM [version 1.0, rev 12, release 19]. July 27th, 2001. In: http://www.psychology.nottingham.ac.uk/staff/cr1/ezdicom.html#users Accessed in: 17/04/2005