User Interface for PEM/PET dedicated breast Imaging & Biopsy Device

Gangadhar Jaliparthi

Problem Report submitted to the College of Engineering and Mineral Resources at West Virginia University in partial fulfillment of the requirements for the degree of

Master of Science
In
Electrical Engineering

Approved by
Dr. Yenumula V Reddy, Ph.D., Chair
Dr. Raymond R Raylman, Ph.D., Advisor
Dr. James D Mooney, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2008

Keywords: IDL, Data acquisition, PEM/PET, Bridge

ABSTRACT

User Interface for PEM/PET Dedicated Breast Imaging and Biopsy Device

Gangadhar Jaliparthi

Today different programming environments are used to develop applications. Development of some systems may require more than one application each in different environments. Development of medical imaging systems requires different applications in different environments for image analysis & visualization, image acquisition and scanner control. Each environment provides its own advantages and requires information sharing with other environment. An optimal solution is to develop a single application which has the complete functionality and unified user interface of all applications. Unification of user interface functionality require cross platform communication. This problem report describes a new technology that allows information sharing between two environments. This technology was applied to create the user interface for PEM/PET, a dedicated breast imaging and biopsy device.

A number of dedicated breast imagers have recently been constructed to help diagnose and treat women with breast cancer. Positron emission tomography (PET) and Positron emission mammography (PEM) are emerging nuclear imaging techniques that help breast cancer detection. A high-resolution positron emission mammography/ tomography imaging and biopsy device called PEM/PET to detect and guide the biopsy of suspicious breast lesions is being developed at the WVU Department of Radiology. A set of eight different computers handle the operation of this device. The software that controls the motion of the hardware (Gantry) for imaging and biopsy, and initiation of data acquisition was custom built with Java programming language. The software for visualization of the breast images produced by the device is designed with Interactive Data Language (IDL). Both of them reside on the Gantry control computer and require information and data sharing.

This problem report explains how the communication between the two programming environments (IDL and Java) can be achieved and how the processing power of both the

environments can be employed. The IDL-Java Bridge was used to establish a connection between both the environments. Using IDL-Java Import Bridge, Java methods can be called from IDL. The Java application for gantry control and data acquisition is designed such that its methods are callable from IDL. Thus, a single user interface was created, which will greatly simplify clinical use of this new system.

Acknowledgments

I would like to express my gratitude to my advisor Dr. Raymond R Raylman for his supervision and valuable guidance throughout the project. I am very thankful for his constant support and encouragement. He always helped me to come up with ideas and implement them.

I would like to extend my thanks to Dr. Ramana Reddy and Dr. James Mooney for serving on my committee and supporting the project with their encouragement.

I would like to thank all my friends for their support.

Finally, I am deeply thankful to my beloved parents and brother for their encouragement and support in all the things I do.

Table of Contents

List of Figures & Tables

Chapter 1 Introduction

1.1 Background

Positron Emission Tomography is a nuclear medical imaging technique which produces a three dimensional image of a functional process in a human body (1). A positron-emitting radionuclide (tracer) is introduced into the body and the pairs of gamma rays emitted by the tracer are detected. Images of tracer concentration in three dimensional space within the body can be reconstructed by computer analysis. Positron Emission Mammography is a high resolution PET scanning which gives functional imaging for dedicated breast cancer detection.

Tomographic breast imaging techniques can potentially improve detection and diagnosis of breast cancer in women. A high resolution positron emission mammography/ tomography imaging and biopsy device (called PEM/PET) to detect and guide the biopsy of suspicious breast lesions was developed by the WVU Department of Radiology. PET images are acquired to detect suspicious focal uptake of the radiotracer and guide biopsy of the area. Limited angle PEM images could then be used to verify the biopsy needle position prior to tissue sampling (2). The PEM/PET scanner consists of two sets of rotating planar detector heads. Each detector consists of a flat panel position sensitive photomultipliers coupled with scintillation detectors. Image reconstruction is performed with 3-dimensional, ordered set expectation maximization algorithm parallelized to run on a multiprocessor system. A field programmable gate array (FPGA) data acquisition system is employed to maximize the event detection rate capability. A set of eight computers control the operation of the device. The hardware (called Gantry) of the PEM/PET system is controlled by Gantry control computer. Data Acquisition is controlled by Event Builder computers and image reconstruction is handled by Reconstruction computer. These three systems are connected via a gigabit Ethernet network.

The software for controlling the Gantry was developed using Java programming language and resides on the Gantry control computer. Eclipse 3.1.2, an integrated

development environment for Java based applications was used to develop the software. The communication between the PEM/PET Gantry and the gantry control computer is through USB2 connection. The acquisition of data from the device detectors is controlled by the Data Acquisition software built with Java programming language developed at Jefferson Labs, Newport News, VA and resides on the Event Builder computers (EVBs) and Data acquisition computers (HDs). The raw data of the sum after data acquisition is copied to the Reconstruction computer, which is accessible to the three computers via a gigabit Ethernet. The raw data is processed by the Reconstruction computer and the final image file is created. The image file produced at Reconstruction computer is accessible to the three computers. The image file is analyzed with software developed using Interactive Data Language (IDL) programming environment. This software resides on the Gantry control computer.

There is a necessity for communication between the Gantry control software on the Gantry control computer and the Data acquisition control software on the EVB. The Gantry control software must trigger the start of data acquisition, which is controlled by DAQ software. Moreover, the DAQ time and the name of the image file to be created must be passed to the DAQ software from the Gantry control software. The data acquisition software acknowledges the successful completion of data acquisition to the gantry control computer. This communication is achieved through Remote Method Invocation (RMI).

IDL, The Interactive Data Language, is a computing environment for the interactive analysis and visualization of data (3). IDL supports cross-platform application development. It integrates a powerful, array oriented language with numerous mathematical analysis and graphical display techniques. Thus, IDL is an optimal choice to develop image visualization software for PEM/PET device to analyze breast images, and guide biopsy. In addition to displaying PEM/PET images, the software is used to calculate the coordinates of the area that must be biopsied using a computer-controlled needle.

To perform the required control tasks, some controlled by Java applications, it is necessary to use the IDL-Java Bridge. Using the IDL-Java Bridge, native Java methods can be called within from IDL. Hence the Gantry control Java application is programmed into a new application with methods that are callable from IDL. Now the complete user interface is designed in IDL environment for the gantry control and image display. The underlying events that are generated from user interaction with the Graphical user interface of IDL call the Java methods using IDL-Java Import Bridge. Hence, the PEM/PET user interface resides on a single computer system (it once took three computer interfaces) greatly simplifying the operation of this new medical imaging system.

1.2 Problem Statement

The goal of this project is to integrate the Gantry control module with the Image display module of the PEM/PET imaging and biopsy device software using the Java-IDL Bridge. To achieve this objective the gantry control procedures, written in Java were melded with the image display and biopsy control procedures, written in IDL.

Chapter 2 Description of the PEM/PET Device

2.1 Introduction

The project was supported by funds from the National Institute of Health. The goal of PEM/PET is to improve the detection of the breast cancer in the subset of women that have radio-dense or fibroglandular breasts. In this group standard x-ray mammography does not always show small lesions due to x-ray absorption. PEM/PET utilizes nuclear medicine techniques that are much less susceptible to these effects. Furthermore, this PEM/PET system can perform biopsies of the regions detected on its images. The PEM/PET system was developed by researchers at WVU Department of Radiology in collaboration with Jefferson Labs, the University of Washington and the University of Maryland.

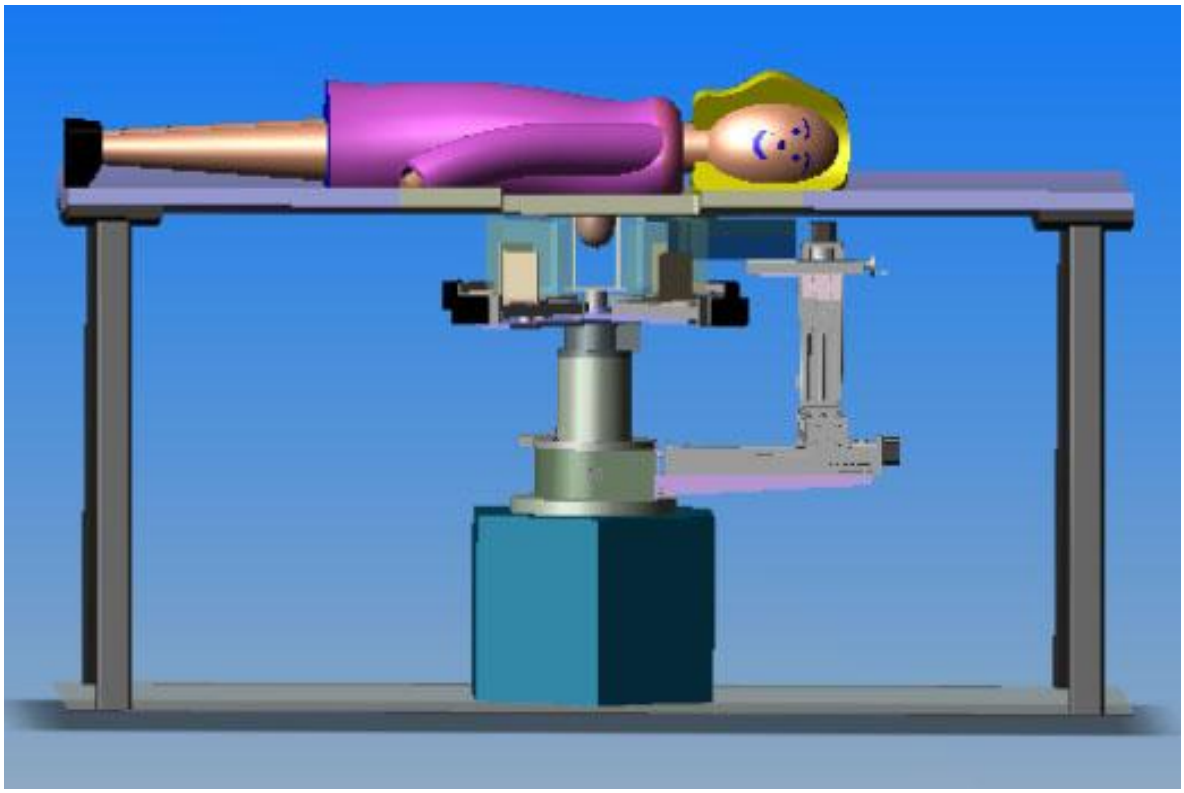

Figure 1: Conceptual drawing of the PEM/PET imaging and biopsy system. (Courtesy: Dr. Ray Raylman, [4]Elsevier Limited)

2.2 Description of the Device

The PEM/PET device imaging and biopsy component consists of four custom designed radiation detectors. Each radiation detector head has a radiation sensitive area consisting of an array of scintillation detector elements. One end of this scintillator array is coupled to an array of Position-Sensitive Photomultiplier Tubes (PSPMT) and the other end is covered with white reflective coating. These arrays are designed to maximize sampling of areas close to chest wall and decrease false negative results due to poor sampling. Each array is dry coupled to an array of flat panel position sensitive photomultipliers (PSPMTs). These fours detectors are mounted on a scanner gantry; each detector is in coincidence with its opposing detector.

The data acquisition software to control and readout data from the four detectors was created with Java programming language and resides on event builder computers. The figure 2 shows a schematic diagram of the information flow. The data acquisition system uses a client-server model for achieving high data throughput. The server software runs on networked computers (2). There are four networked server computers namely HD1 to HD4, correspondingly to each of the four detector heads. The server software provides access to several fast analog to digital converters over a transmission control protocol connection. The client computers (Event Builders EVB13 and EVB24) have the graphical user interface which allows users to select and configure servers, and synchronously start and stop acquisition. Multi channel analog to digital converters based on Field Programmable Gate Arrays (FPGAs) are used to digitize the signals from the PSPMTs. Every detector has a dedicated 64 channel ADC-FPGA unit that determines event positioning, event time stamp and energy disposition in the scintillator array. Data is sent via a USB2 connection to the HD data acquisition computers.

The capturing of data is initiated by the coincidence of trigger pulses generated by the signals from each opposing detector head. The data from the HD data acquisition computers is transfer to another computer (Event Builders (EVB)) for indentifying coincidence events. HD1 and HD3 send data to EVB13, HD2 and HD4 send data to EVB24. The coincidence events are detected by the time stamps of events from opposite detectors. These data are stored as list mode files. Step and shoot data acquisition is carried

out for PET imaging by the device. So, the list mode data for each step is stored in a separate file. Therefore for a four step data acquisition process, eight list mode files are produced four for each pair of opposite detectors.

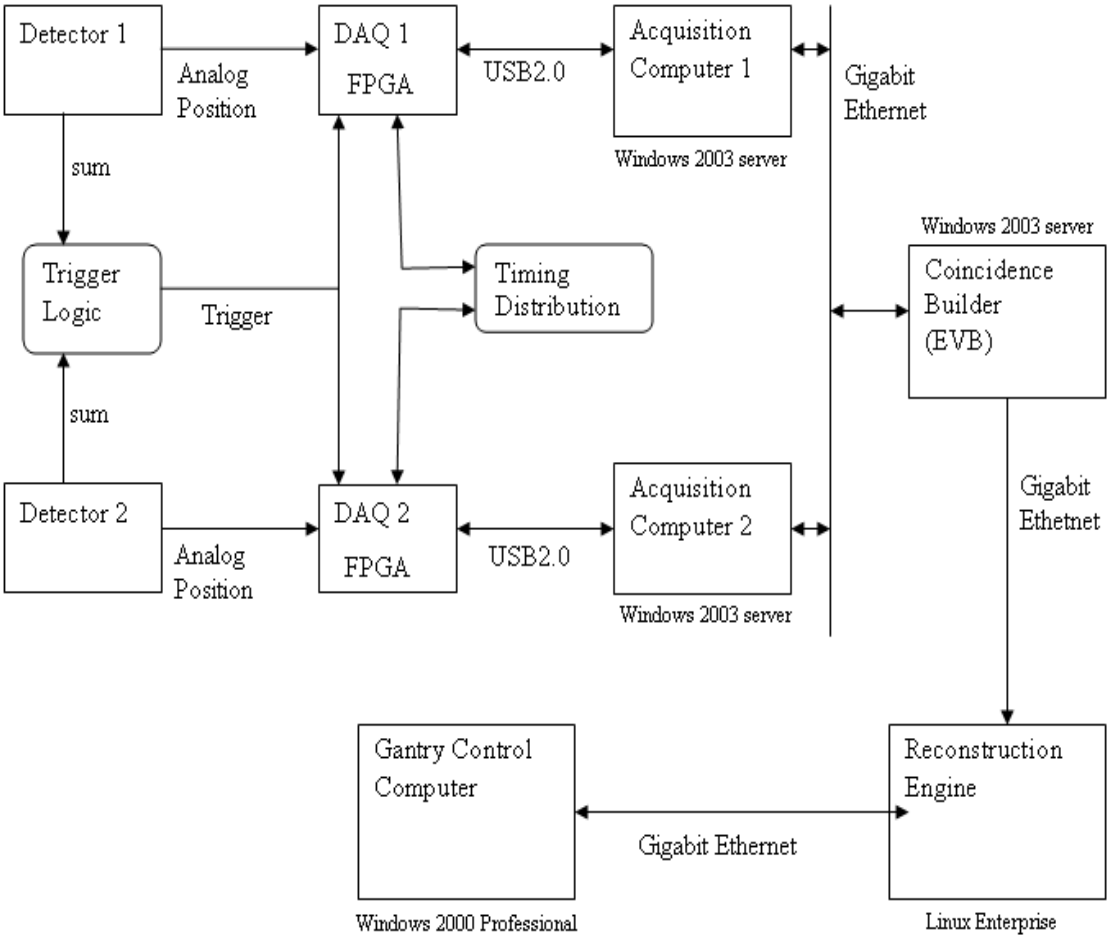

Figure 2 Schematic Representation of the Data Flow of the PEM/PET data acquisition system. Courtesy: [2]

The four detectors are mounted on a custom scanner gantry. In order to facilitate the movement of the detector heads, each detector head was fixed to a computer controlled linear slide. This allows varying the distance of the detector heads from the center of the device. The linear slide can move the detector heads away from the center to keep the device in biopsy position and move towards center for PET imaging position. These linear sliders are mounted on a computer controlled rotary stage. The rotary stage allows the

angular movement of the detectors to facilitate step and shoot data acquisition. The user can select the numbers of steps for data acquisition, angular separation between each step. The user can also select the dwell time for the first step. The system determines the dwell time for next steps based on the radioactive decay of the source. The motion of the linear sliders and rotary stage that determine detector separation and rotation are controlled by custom written Java software.

The biopsy arm of the PEM/PET device is connected to the central axis of the gantry via a thrust bearing. This allows a 360 degrees movement of the biopsy arm around the scanner's field of view (FOV). This allows biopsies from any angle. The position of the arm is monitored by rotary position encoder. The position of the arm is displayed as arm angle on the gantry control software GUI. The biopsy gun is placed on a computer controlled three axis stage. The biopsy gun holds the needle for biopsy purposes. The three axis stage can be moved to position the needle using the images of the breast acquired from the scanner. The arm holding the detectors can swing apart from PET imaging position (the detectors are moved away from the center of the device to extreme ends by the linear sliders) to the biopsy imaging position. In PET imaging position the separation between each detector 90 degrees. In Biopsy imaging position the angular separation is 130 degrees to allow the biopsy apparatus to move close to the breast to perform biopsy. Limited angle PEM (Positron Emission Tomography) images are acquired by the scanner in biopsy imaging position. The PEM images are used to verify the proper positioning of the needle for biopsy operation. The detectors are stationary while acquiring PEM images. The controllers for all of the scanners motion control and monitoring are housed in an equipment bay located beneath the detectors. The communication between the gantry control computer and the device units is through USB2 connection from computer to a USB to RS232 hub located in equipment bay (2). The data acquisition computer, gantry control computer and reconstruction computer are connected via a gigabit Ethernet.

The list mode data files obtained from event builders after the data acquisition is transferred to reconstruction computer. Here the data is reconstructed with a three-dimensional, ordered set expectation maximization-based (OSEM-based) algorithm

specially parallelized to run on a custom eight CPU computer. The final reconstructed image is accessible to the gantry control computer via common sharable directory on reconstruction computer.

2.3 Architecture

A set of eight computers control the operation of the device. The Gantry control computer houses the software that controls the movement and operations of Gantry hardware. The software is custom built with Java programming language. The communication between the gantry control computer and PEM/PET gantry is performed via a single USB2 connection from the computer to a USB-to-RS232 hub located in the equipment bay.

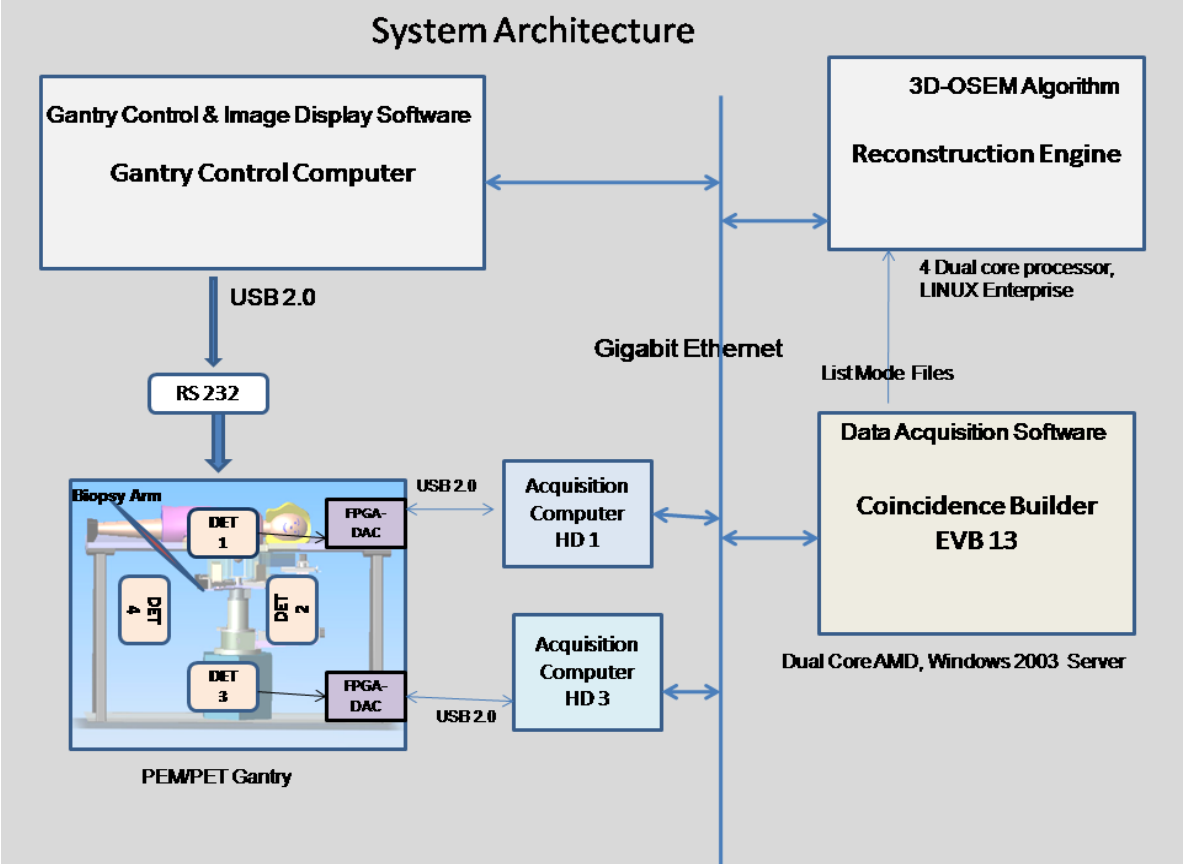

Figure 3 Schematic Representation of PEM/PET System

There are four HD computers each acquiring data from one detector. The acquired data from HD Computers is sent to Event Builders - EVB (Coincidence Builder)

Computers. There are 2 EVB computers each getting input from the two opposing HD Computers. Thus, EVB13 gets input from HD1 and HD3, EVB24 gets input from HD2 and HD4. The data is processed here for coincidences and the output is sent to reconstruction computer where the image file is produced. The event builder, reconstruction computer and gantry control computer are connected via a gigabit Ethernet. A common folder on the reconstruction computer is accessible to both EVB and Gantry control computer. Event Builder Computers store their output raw data in the common folder on the reconstruction engine (a quad core system). The final image after reconstruction is also stored in this common folder which is accessible to Gantry control computer.

Computer	Operating System	Configuration
Gantry Control Computer	Win 2000 professional	Dell PowerEdge 850, rack mount
Event/ Coincidence Builder EVB (2)	Win 2003 Server	Dual core AMD Opteron 885 -based, rack mount
Data Acquisition Computer HD (4)	Win 2003 Server	Dual core AMD Opteron 885 -based, rack mount
Reconstruction Engine	Linux Enterprise	Four Dual core Opteron processors

Figure 4 System Configuration

Chapter 3 Implementation

3.1 Introduction

Custom built Java based software that resides on the Gantry control computer controls the operation of the PEM/PET System hardware for Imaging and Biopsy. This software controls the motion of the motors and coordinates the initiation of data acquisition. The motors set the position of the apparatus of the device to either PET or PEM position to acquire images. The motors also control the motion of needle during biopsy. This software was initially designed as a single application with a graphical user interface. The user interacts with the interface to operate the device. The user can initialize the device, emergency stop the device, move the detectors from PET to PEM position and vice versa, and initiate data acquisition. The device operates in two modes: PET and Biopsy. In PET mode, the device will acquire images of breast using positron emission tomography. In this mode of operation each detector is 90 degrees apart from its adjacent detector and the detectors are close to the centre of FOV (Field of view) of the device. In Biopsy mode the detectors are moved away from the center of FOV and separated by 130 degrees with one other detector. This creates enough room for the biopsy apparatus to come close to the breast to perform biopsy. In Biopsy mode, Positron emission mammography images are acquired for verification of the position of needle. The needle is guided to the specific co-ordinates obtained from PET images to get the tissue sample of the lesion. The main components for building the gantry control software are discussed below.

3.2 INITIALIZATION:

Initialization is the first step required to operate the device. Initialization establishes communication between the gantry control computer and the PEM/PET device, moves all the apparatus to the initial (Home) position, initialize all the required variables, initializes the entire device motors and starts the RMI client to communicate with data acquisition computer. Initialization is done in a sequential manner and the software checks for the success after each step. The control proceeds to the next step only if the present step is

successful. If any step of initialization returns failure, then the process is stopped and an error message is displayed accordingly. A success message is displayed after each successful step. The current scanning mode (PET or PEM) status is stored in a variable. This variable is initialized to PET mode by default. Initialization of the device is required to ensure proper operation of the device whenever it is operated.

3.2.1 Initialization in Normal Mode (PET):

Communication between the gantry control computer and the PEM/PET Gantry is through a single USB2 connection from the computer to a USB-to-RS232 hub located in the equipment bay. There are six communication ports with USB2.0 connection that need to be established for the communication between the device and the control computer. This is done only once whenever the gantry control application is started. These six communication ports are opened for communication with the device hardware. The Arm Angle controller communication port is opened first and the status of the Arm Angle controller is checked. The Arm Angle controller reads the current angle value of the three-axis armature stage that holds the biopsy needle. The Arm angle is updated continuously for entire time of device operation. The process to update and display the current armature angle is done in a separate thread as it takes considerable amount of time and runs for entire time of device operation or until emergency stop. The Armature motors are initialized next. The Arm Gun motor port and the motor parameters are set, and then the Arm Display motor port, Arm control motor is set. The communication ports for these motors are opened. The initialization of the rotation motor is done next. This includes opening the communication port for rotation motor, setting the rotation motor online and starting the RMI client for communication with data acquisition software. Finally, initialization of the detection motors is performed.

3.2.2 Initialization in Biopsy Mode:

When entering the biopsy mode, the software checks whether biopsy gun is present. If the biopsy gun is not present an error message is popped up to insert it. The software also checks if the detectors have been moved back to their proper position and if the detector arms have been moved to the open position. If all of these criteria have been met, the biopsy arm is moved into position.

3.3 EMERGENCT STOP:

Emergency stop is a condition where the motions of all PEM/PET systems are stopped by command of the user. Though the occurrence or usage of emergency stop is rare, it is very important to protect the device and patient from any harm. The sequences of steps that occur to stop the operation in the state of emergency are: The software checks for the current mode of the device. If the mode is PET, the arm control motor, arm gun motor, detection motor and rotation motor are stopped, and stop the reading of current arm angle. If the current mode is Biopsy, the Arm control motor is stopped.

3.4 BIOPSY MODE:

This mode of operation allows biopsy of suspicious lesions. In this mode detectors are shifted away from the center of the FOV to a separation of 41 cm and each pair of detectors heads will rotate to a stereotactic imaging geometry (where the arms holding the detectors will swing apart for detector separation of 130 degrees) so that the biopsy apparatus can be brought close to the breast (4). Setting the device to biopsy mode is accomplished in a sequential way. The sequences of steps are as follows: The software checks for the current mode of the device. If the device is in PET mode then detector motors moves the detectors to PEM position (shift the detectors heads away from the center of FOV). All the 4 motors will be in their axis extreme end. If this method returns success then the motors are in PEM position. The second step involves removing the detector switch and manually moving the detectors 130 degrees apart and closing a micro switch. Then the software checks for current mode. If the current mode is PET, then it checks for the status of the detector arm micro switch. If the switch position is closed, then software checks for the status of arm gun motor switch. The biopsy gun switch specifies whether the gun is present (closed position). After all these steps are performed successfully, the software sets the current mode variable to “Biopsy”. If any of the steps fail, the operation stops at that point and displays the error message and the action to be taken to rectify the issue.

Once the biopsy mode is set, the user can give the coordinates of the position to which the needle should be guided for biopsy. The user selects this co-ordinates from the PET image acquired from the device. The software communicates with the armature control motors to move the needle to the specified position. The position of the needle will be verified by acquiring a stereotactic set of stationary PEM images prior to the acquisition of the tissue sample. The needle tip contains a Na source for detection in PEM.

3.5 PET MODE:

This mode allows the acquisition of tomographic images of the breast from multiple projection angles. In this mode the detector heads are placed close to the center of the FOV (Field of view) to acquire PET data. In this mode the arms holding the detectors are spaced by 90 degrees apart from each other. Setting to PET mode is accomplished by performing the following steps by the software. The software checks for the present mode variable. If the present mode is “biopsy” then, the software checks for the biopsy gun motor switch status. If it is open (gun is not present), then the gun motor is moved to its initial position and is set online. The success of the above events follows with the movement of the arm motor to home position. Then software checks for the status of the detection arm position switch. A warning message is displayed to check detector position if the detector position switch is open. If the detection position switch is open then the four detectors are moved to PET position i.e. they are moved towards the center. After moving the detectors to the PET position, the present mode variable is set to “PET”.

The current arm angle is constantly updated after the initialization of the device and is displayed. This process runs in a separate thread and stops when the application is stopped or in the case of emergency stop. The current position of the three arm axis is displayed whenever requested.

3.5 DATA ACQUISITION:

The software to control and readout data from the four detector units was created with the Java Programming Language. This data acquisition control software resides on the Event Builders (EVB13 and EVB24). The initiation of the Data Acquisition is coordinated

by the gantry control software. This is done by calling the remote method of the Data acquisition software that starts DAQ from gantry control software.

PET Data Acquisition: The software checks for the mode of operation, detectors position and status of detector switch. If the present mode is 'PET', detectors are in PET position and the detector switch is open, then PET data acquisition is enabled by the software. The Gantry control software takes the input arguments as the filename, scan time, EVB selection, number of steps and increment angle from the user. The filename parameter is used for naming the image files created by data acquisition software. Since the Imager operates in step and shoot mode, the number of steps for data acquisition, the angular separation between each step and the scan time (dwell time) between data acquisition are specified by the user. The dwell time for the next steps is calculated by the software based on previous dwell time to account for the radioactive decay of the source. The gantry control software calls the remote method on the specified EVB (or both EVBs) to start the data acquisition. The Gantry software polls the DAQ software on the EVB continuously for the successful completion of data acquisition. After each successful data acquisition the Gantry software rotates the detectors one step by the specified angle and then calls the remote method to start the next data acquisition. This continues until all the request data acquisitions have been performed.

PEM Data Acquisition: If the present mode of operation is 'Biopsy', detectors are in 'PEM' position and the detector switch is closed, then the software enables PEM data acquisition. The software takes the file name and the dwell time as the input arguments for the data acquisition from the user. The gantry control software invokes the remote method on both EVBs to initiate data acquisition with the given two parameters. The detector position is not changed in PEM data acquisition and only once data acquisition is initiated.

3.6 REMOTE METHOD INVOCATION:

Java Remote Method Invocation known as Java RMI enables the creation of distributed Java based applications (5). Using RMI, the methods of remote Java objects can be invoked from other Java virtual machines on same or different hosts.

RMI applications consist of two separate programs, a server and a client. Typically, the server program creates a remote object, makes references to these objects accessible, and waits for clients to invoke methods on these objects. Typical client program obtain a remote reference to one or more remote objects on a server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth.

A simple RMI application is shown in this schematic.

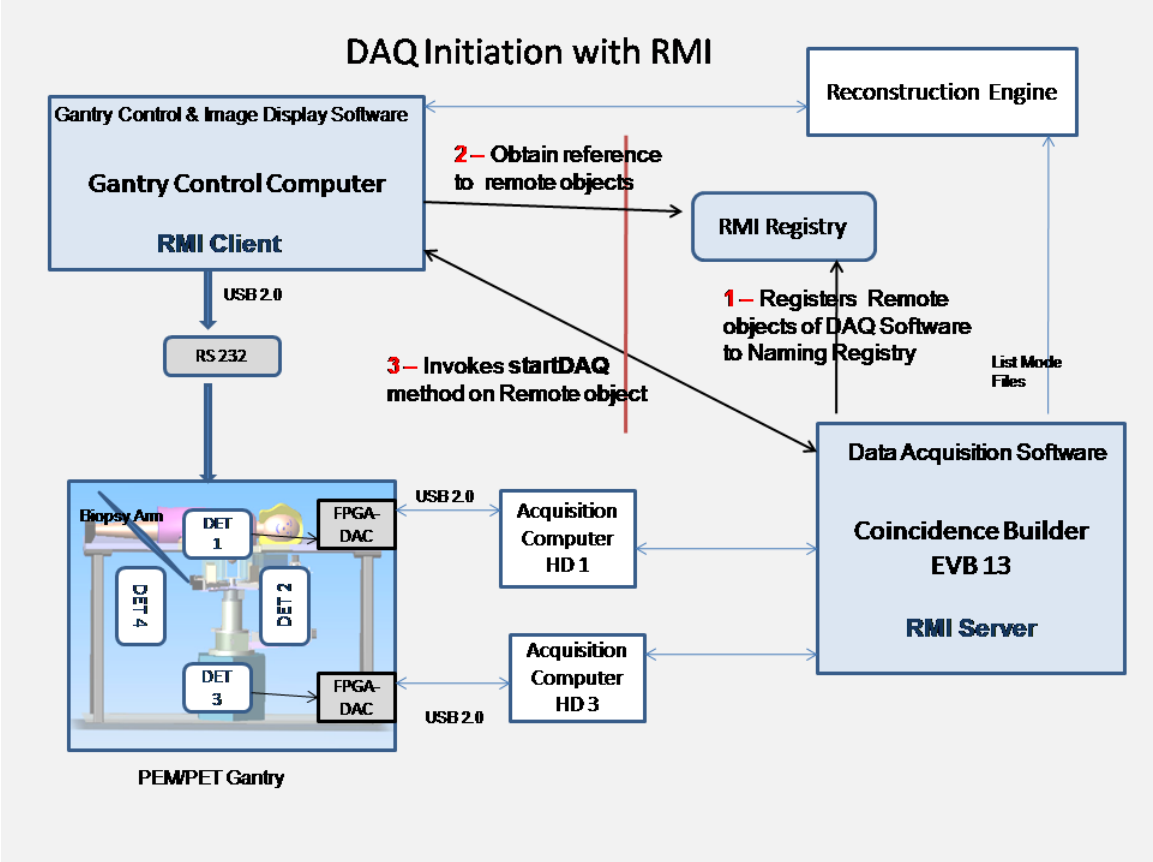

Figure 5: Remote Method Invocation

The server creates remotes objects and registers its remote objects with RMI's simple naming facility, RMI Registry. The client looks for the references to the remote objects in the RMI registry which is accessible to it. The reference to the remote object is obtained and the client invokes the methods on the remote object as regular Java method invocations. RMI handles the communication between remote objects and client applications.

In the PEM/PET the server programs reside on the event builder computers. They implement the software that handles the data acquisition. The client software is located on the gantry control computer. The DaqOperation class of the gantry control software looks for the reference to the remote objects on event builders and invokes the StartDAQ method on the remote object of the Event builders. The arguments passed to the method that starts the Data Acquisition are name of the file to be created by data acquisition and the data acquisition scan time. The client keeps polling the daqReady method on the event builder for the completion of the data acquisition. As soon as the data acquisition is completed, the software on the Event builders sets a boolean variable that is returned by daqReady method to true. The client program reads the boolean value returned by daqReady method and concludes that the data acquisition is complete. It then transfers the generated file on the event builder to the reconstruction computer.

3.7 METHODS:

The Gantry control software is designed as an application with methods that are callable from IDL. The application consists of a main class and other supporting classes. Different classes are implemented for control of Arm motors, Detection motors, Rotation motors, and Arm angle display. All the methods in the supporting classes can be called from the main class file. A separate class stores the constants. All the methods that are callable from IDL are in main class as the IDL-Java Bridge calls the constructor of the main class. The methods that are called from IDL are:

1. INITALIZE

   The Initialize method wraps all the functionality required to initialize the PEM/PET device. This method returns an integer value. This method returns “Zero” if the device is successfully initialized and “One” if a failure occurred during the process. It takes into account the present mode of operation of the device and initializes it accordingly. If the device is in PET mode it establishes the communication between the gantry and the gantry control computer and initializes the entire motors and required variables. If the device is in Biopsy mode it initializes the entire arm motors.

2. Emergency STOP

The Emergency stop method wraps all the functionality to stop the device in the case of emergency. All the motors are stopped whenever this method is called.

3. set_PEM_Position

The set_PEM_Position method wraps the steps to move the detectors from PET position (All the fours detectors towards the center of the FOV of the device) to PEM position (All detectors are moved to their extreme ends). This method returns integer value “Zero” if moving the detectors to PEM position is successfully and integer “one” if failure occurs.

4. set_Biopsy

The Set_Biopsy method includes all the steps required to set the PEM/PET device in biopsy mode. This method checks for the status of the detector position switch and arm gun switch. If they are OK, then it moves the three arm axis to positive home position. After success, this method sets the mode variable to “Biopsy” and returns an integer value of “zero”.

5. MOVE

The MOVE method contains the functionality to move the biopsy needle to the specified location. This method takes two input string parameters i.e., the co-ordinates obtained from the PET image about the location of the suspicious lesion. This method verifies the values of the co-ordinates to where the needle. Integer value “zero” is returned if the needle is successfully moved to the specified location, otherwise “one”.

6. set_PET_Position

The set_PET_Position method wraps the functionality to move the detector position from PEM (all detectors at extreme ends) to PET position (detectors towards the center of device FOV). This method returns integer value “Zero” upon success and “One” on failure.

7. set_PET_Mode

The set_PET_Mode method wraps all the steps to set the device to PET mode. This method checks for the status of the arm gun switch and then moves the arm gum motor and arm motor to home position. If success, then updates the present mode variable to “PET” and returns integer value “Zero”.

8. UPDATE

The UPDATE method updates the GUI display values for positions of the biopsy arm motor. This method returns a String variable which contains the concatenated values of the positions of the three axis.

9. start_DAQ

The start_DAQ method wraps the functionality to initiate data acquisition. This method takes three input parameters: the name of the file created by data acquisition process, the scan time for data acquisition and the current event builder selection. This method checks for the status of the detector position switch. If the detector switch is closed (Detector arms in biopsy position), a warning message stating “move detector arms to normal position and retry” is displayed. If the detector position switch is closed then, it checks for the detector heads position. If the detectors are in PET mode (all the detectors are close to center) a method on the RMI client is called to start the data acquisition. If the detector position is PEM (all detectors at extreme ends) then, arm gun motor is kept online and initialized. Then the arm gun motor is kept offline and the method on RMI client is called to start data acquisition. The startDAQ method returns integer value “zero” for successful data acquisition and “one” for failure.

10. HOME

The HOME method moves the rotator motor to home position. This method return integer value “zero” on success. This method checks for the status of the detector position switch. If the detector switch is closed (Detector arms in biopsy position), a warning message stating “move detector arms to normal position and retry” is

displayed. If detector position switch is closed then, it checks for the detector heads position. If the detector position is PEM (all detectors at extreme ends) then, arm gun motor is kept online and initialized. Then the arm gun motor is kept offline and the rotator motor is moved to initial home position.

11. get_Rotator_Angle

The get_Rotator_Angle method returns the current angle of the rotator motor in degrees. This is required to know the current position of rotator motor during the data acquisition process. This method is called after the successful completion of each data acquisition and after moving the rotator motor to home position to verify the current position.

12. Rotate

Rotate method moves the rotator motor specified degrees from the current position. This method takes the increment angle required as a string parameter.

The gantry control software was originally developed as an application with a user interface designed in the Java programming language. This is a complete application and has all the functionality to operate the gantry of the PEM/PET device. The user interface of this application is shown below.

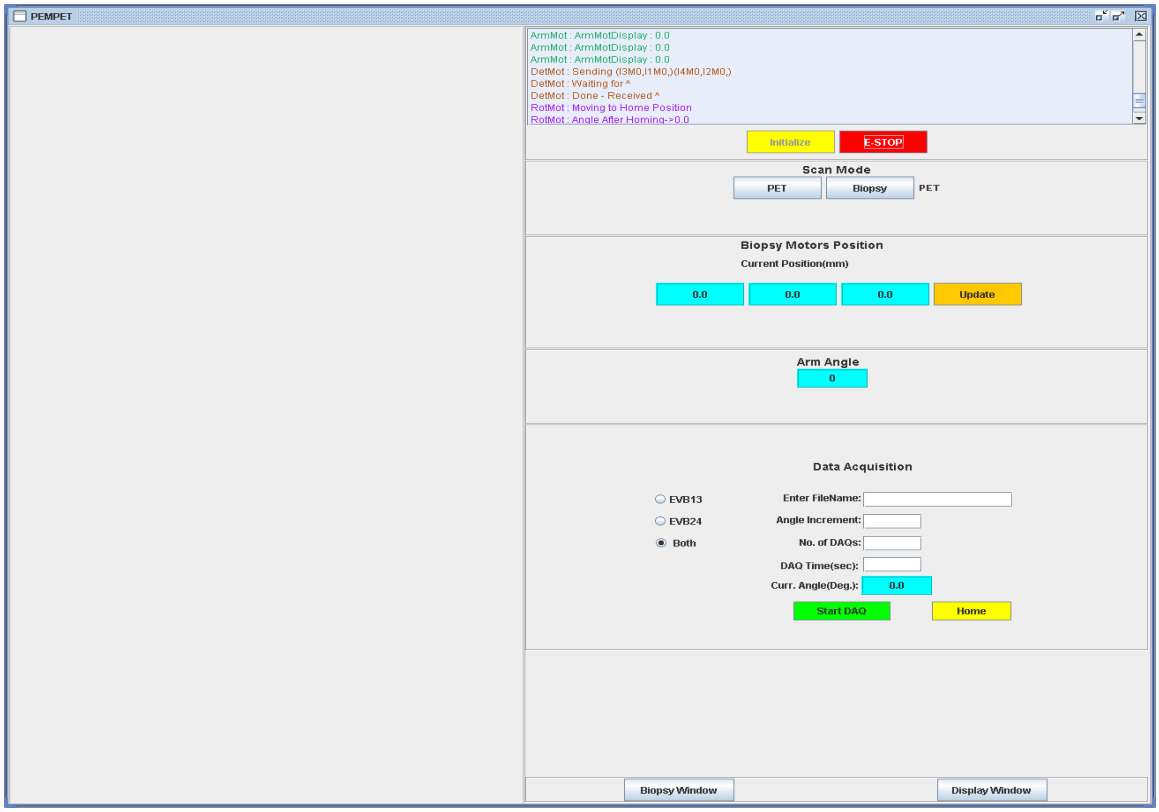

Figure 6 User Interface for Gantry Control`

3.7 Introduction to IDL

IDL, The Interactive Data Language is a complete computing environment for the interactive analysis and visualization of data (3). IDL supports cross-platform application development. IDL integrates a powerful, array oriented language with numerous mathematical analysis and graphical display techniques. IDL Programming is a time-saving alternative to FORTRAN, Java or C Language Programming.

IDL is very powerful for data analysis. Many numerical and statistical analysis routines including Numerical recipes routines are provided for analysis and simulation of data (6). Compilation and execution of IDL commands provides instant feedback and hands-on interaction. Operators and functions work on entire arrays without using loops.

This greatly simplifies interactive analysis and reduces programming time. The flexible input/output facilities of IDL allow reading any type of custom data format.

Visualization advantages of IDL include rapid 2D plotting, multi-dimensional plotting, volume visualization, image display, and animation. IDL supports for OpenGL-based accelerated graphics. IDL is a complete, structured language that can be used interactively and to create sophisticated functions, procedures, and applications. IDL widgets can be used to quickly create multi-platform graphical user interfaces to IDL programs. IDL programs run across all supported platforms (UNIX, Macintosh and Microsoft Windows) with little or no modification.

The various image processing techniques provided by IDL are: Images can be modified in IDL by transforming, cropping, padding, shifting, reversing, transposing, rotating, translating and resizing. Dimensionality can be added to image by mapping images onto surface overview, sphere and elevation data. IDL provides masking and clipping of images, image wrapping, locating pixels values in images and image statics. Creating region of interests and working on them is allowed by IDL. Transforming images between domains is supported in IDL (from frequency to time). Enhancing Image contrast and filtering, smoothing, sharpening and noise reduction can be achieved through IDL. Extracting and analyzing the shapes can be done.

3.8 Image Display and IDL

IDL provides numerous advantages for data analysis, visualization and processing. The advantages provided by IDL in the field of image processing and visualization are promising compared to Java programming language. Therefore the software for image visualization and analysis of the PEM/PET imaging device was developed with IDL. The image file created by the reconstruction computer is displayed with this software. This software provides the ability to analyze the breast image file for the detection of suspicious lesions. The co-ordinates of any point on the PET image can be obtained. These co-ordinates represent the real location of the breast tissue with respect to the center of the FOV of the device. Therefore, suspicious lesions of the human breast can be detected and

their location co-ordinates are obtained using this software. These co-ordinates can be sent to the gantry control software to perform biopsy operation.

The user interface of the Image Display program of IDL is shown below. The PEMPET Image display widget is the user interface for analysis, visualization and processing the data file produced by the reconstruction computer. The data is visualized in the image area of the PEM display widget. The biopsy control widget handles the movement of needle for biopsy purposes. The image display status widget displays the status messages from the IDL.

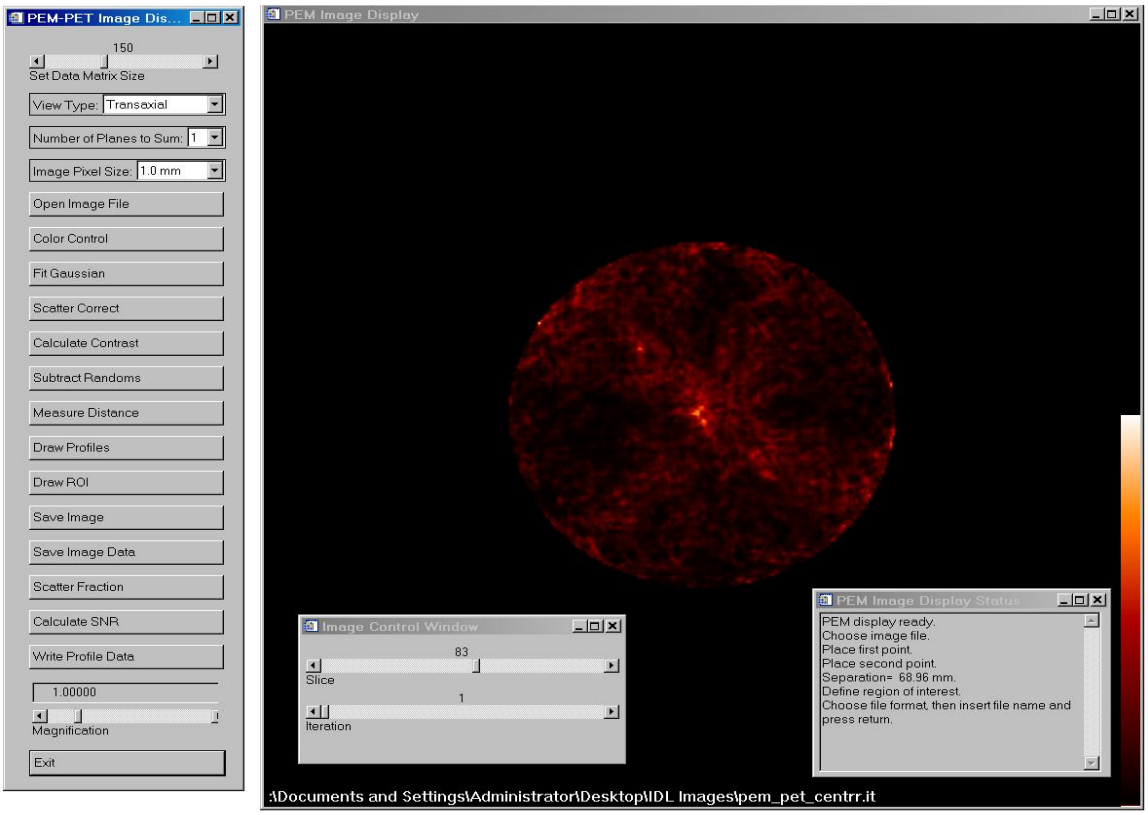

Figure 7: User Interface of Image Display Software

Graphical user interface can be created and manipulated using widgets in IDL. Widgets are simple graphical objects such as simple buttons, text areas that allow user to interact with a point device like mouse or keyboard. Widget objects like simple buttons, radio buttons, text labels, text areas and sliders can be added to base of widget. These

widget objects can generate events with user interaction. This interaction can be a simple mouse click or a key stroke. For example, a button click can generate an event. An event handler will handle all the events generated. Each widget object can specify the underlying procedure to call when an event occurs. This is handled by 'XMANAGER', an event handler.

The operation of PEM/PET imaging and biopsy device requires the Java application for Gantry control and the IDL application for Image display. Each application has a separate user interface. To simplify the operation of PEM/PET system for clinical use a unified user interface is necessary. The Java-IDL Bridge can be exploited to accomplish this task. Communication between both application environments (IDL and Java) can be achieved with this technology. Specifically, a single user interface can be designed in the IDL environment for both gantry control and image display. Java methods can be called from IDL to control the gantry.

3.9 Connecting Java and IDL

A bridge is a technology path that lets applications in different programming languages or environments share information (7). With bridge technology, you can use an application that manipulates data in its native language (e.g., Java) by calling on objects and processes from another language (e.g., IDL). In this way, you can take advantage of both environments to solve a problem that might be otherwise difficult for either environment separately.

IDL supports import and export bridge technology. The Import Bridge lets you import the functionality of a Java object to an IDL application. The Export Bridge lets you export the functionality of an IDL object to JAVA application.

With Export Bridge, interaction with IDL is through native Java wrapper objects that are generated for each IDL object with which client applications want to interact. The wrapper objects manage all aspects of IDL Loading, initialization and process

management, so that the users need only familiar with client applications and basics of IDL.

The IDL-Java Import Bridge allows you to access Java objects within IDL code. Java objects imported into IDL behave like normal IDL objects. The IDL-Java Bridge allows the arrow operator (−>) to be used to call the methods of these Java objects just as with other IDL objects. The bridge also provides IDL with access to exceptions created by the underlying Java object. This access is provided by the IDLJavaBridgeSession object, which is a Java object that maintains exceptions (errors) during a Java session.

The IDL-Java Bridge uses the Java Native Interface (JNI), the reflection API, and the JVM to enable the connection between IDL and the underlying Java system (8). The IDL OBJ_NEW function can be used to create a Java object. A Java-specific class token identifies the Java class used to create a Java proxy object. IDL parses this class name and creates the desired object within the underlying Java environment.

The Java-specific token is a case-insensitive form of the name of the Java class. Besides the token, the case-sensitive form of the name of the Java class is provided because Java itself is case-sensitive while IDL is not. IDL uses the case-insensitive form to create the object definition while Java uses the case-sensitive form. After creation, the object can then be used and manipulated just like any other IDL object. Method calls are the same as any other IDL object, but they are vectored off to an IDL Java system, which will call the appropriate Java method using JNI. The OBJ_DESTROY procedure in IDL is used to destroy the object. This process releases the internal Java object and frees any resources associated with it.

3.10 Initializing the Java-IDL Bridge

The IDL-Java Bridge must be configured before Java objects can be created and used within IDL. IDL initializes the bridge when it first attempts to create an instance of

IDLjavaObject. Initializing the bridge involves starting the Java Virtual Machine, creating any internal Java bridge objects including the internal IDLJavaBridgeSession object.

When an IDLjavaObject is created, the IDL-Java Bridge loads configuration information from a file named idljavabrc in windows. The IDL-Java Bridge only reads the configuration file once during an IDL session. The file is read when the first instance of the IDLjavaObject class is created in the session. If you change the configuration after the first instance is created, you must exit and restart IDL to update the IDL-Java bridge with the changes.

The configuration file contains the following settings:

1. JVM Classpath: This specifies the locations for the user's java class files. It must point to the location of any class files to be used by the bridge. On windows, paths should be separated by semi-colons. This path may contain folders that contain class files or specific jar files. The classpath environment variable of JVM can also be added.

   JVM Classpath = $CLASSPATH; C:/RSI/resource/bridges/import/java/javaidl.jar

2. JVM Lib location: This tells IDL-Java Bridge which JVM shared library within a given Java version to use. Various versions of java ship with different types of JVM libraries.

   JVM LibLocation = C:\Program Files\Java\jdk1.6.0_6\bin\client

3. Log Location: This specifies the directory where the Java-IDL Bridge log files will be created. The default location provided by the IDL is C:\temp on windows.

4. Bridge Logging: This variable specifies the type of logging. SEVERE specifies bridge errors to log in the above log file. CONFIG specifies configuration settings to be logged in the log file.

3.11 Working

IDL initializes the bridge when it first attempts to create an instance of IDLjavaObject. IDL loads the bridge configuration file "idljavabrc". The path of the configuration file can be specified by the environment variable IDLJAVAB_CONFIG. If

this environmental variable is not set then, the configuration file can be found in the default location given below:

IDL_DEFAULT>/resource/bridges/import/java/idljavabrc

The Java source files are compiled and the class files are archived as a JAR (java archive) file. The location or path of the jar file is specified by the JVM Classpath variable of configuration file. The JVMClasspath variable is specified as below:

JVM Classpath = C:\RSI\resources\bridges\import\java\javaClassFile.jar

The JVM LibLocation points to the Java Runtime Environment (JRE), java compiler of the Java Development Kit- JDK 1.5.7. The JVM LibLocation is shown below:

JVM LibLocation = C:\Program Files\Java\jdk 1.5.7\jre\bin\client

The bridge logging variable is set to SEVERE for bridge errors and log location is specified by log location variable as c:/temp. The Java Communications API is a Java extension for building platform independent communications applications for technologies such as embedded systems. The Java communications API which is known as javax.comm provides applications access to RS-232 hardware serial ports. Since we are using RS-232 as the medium of communication between the device and the gantry control computer, we require the Java communications API (javax.comm). Java communications API library package "comm.jar" (Java archive of javax.comm) must include in the ext directory of JDK installation. JVM also requires WIN32COM.dll file to be included in the library directory of the java installation. The lib directory can be found at

C:\Program Files\Java\jdk 1.5.7\jre\lib

The compiled class files archive of the data acquisition software should be included in the ext folder of JDK installation. The ext folder of the JDK can be found as bellow:

C:\Program Files\Java\jdk 1.5.7\jre\lib\ext

A Java object is created using the IDL OBJ_NEW function. Keying off the provided class name, the underlying implementation uses the IDL Java subsystem to call the constructor on the desired java object. The basic syntax for creating a new java object within IDL using OBJ_NEW is given below.

oJava = OBJ_NEW(IDLjavaObject$JAVACLASSNAME, JavaClassName [arguments])

JAVACLASSNAME is the class name token used by IDL to create object. JavaClassName is the class name used by Java to initialize the object. Arguments are comma separated list of data parameters required by the constructor of Java class. The OBJ_NEW function creates the java object within IDL by calling the constructor of the corresponding Java class.

The java class file that contains the main() is edu.wvu.hsc.cai.Main.FrameTest.java. The IDL-Java Bridge is initialized by this statement of the IDL code:

oPemPet = OBJ_NEW('IDLjavaObject$FrameTest','edu.wvu.hsc.cai.Main.FrameTest')

As soon as the IDL looks at the 'IDLjavaObject' parameter of the OBJ_NEW method, it loads the Bridge configuration file 'idljavabrc'. The OBJ_NEW method calls the constructor of the main java class file – edu.wvu.hsc.cai.Main.FrameTest.java and the IDLjavaObject "oPemPet" is created. IDLJavaBridgeSession object is also started to handle java exceptions.

Once the IDLjava object is created, the methods of the Java class can be called from the IDL. The methods of Java class can be called as regular methods of the IDL object. When a method is called on IDLjava object, the method name and arguments are passed to Java-IDL subsystem and the Java Reflection API to construct and invoke the method call on the underlying java object. The general syntax to call a java method that returns a value is given below.

Result = ObjRef ->Method (Arguments)

When a method is called on the instance of IDLjava object, IDL uses the method name and arguments to construct the appropriate method calls on the underlying Java object. IDL follows an algorithm for case sensitive incompatibilities between IDL and Java. To reconcile instance where class names are the same, IDL first searches for java class method that matches the method name provided except for case matching. If different methods with same name but different case are found then one with all uppercase is taken,

else an error was issued. Then the arguments are matched and the method with correct arguments is called.

As mentioned earlier, the Graphical user interface for gantry control is designed with IDL widget technology. The user can interact with a widget through its various objects. Whenever the user interacts with widget objects, like buttons or labels, these widget objects generate events. Events are handled by event handler called "XMANAGER". Whenever an event is generated by user interaction, the event specifies a particular procedure to handle the event. For example, if the user clicks the "initialize" button on the widget, it generates an event which calls the "init" procedure. The "init" procedure contains the functionality to initialize the PEM/PET device. This procedure calls the initialize method on the Java program with the help of the IDLjava object created. Some procedures may require modifying the graphical user interface features. For example, if user clicks the "update" button, it generates an event to call the update procedure. The update procedure in turn calls the Java method for updating the values of the arm motor axis positions. The Java method returns the present axis position. The updated axis is displayed on the GUI axis labels. Therefore, these procedures need the reference of the IDLjava object and widget objects. The IDLjava object reference is required to call the Java methods and the reference to widget objects is required to modify them if requested. To make these objects available to all procedures, a pointer array is created with references to these entire objects. The WIDGET_CONTROL routine is used to set the same pointer array value as the widget base. Every procedure can request the widget_control routine to deliver reference to the objects.

The syntax to create pointer array and the widget_control routine are shown below.

```
sState =  {
           oPemPet: oPemPet, $
           WID_BASE_1:WID_BASE_1, $
           INIT_BUTTON: INIT_BUTTON, $
           ARMANG_LABEL_2: ARMANG_LABEL_2, $
           ARMMOT_BUTTON_1: ARMMOT_BUTTON_1, $
           ARMMOT_BUTTON_2: ARMMOT_BUTTON_2
          }
pState =PTR_NEW(sState, /NO_COPY)
WIDGET_CONTROL, WID_BASE_1, SET_UVALUE=pState
```

The “init” procedure is called by the event generated when the user clicks the initialize button. In this procedure, the reference to the IDLjava object “oPemPet” is obtained from the pointer array “pState”. Then the initialize method of java class is called with the arrow operator as below.

Status = (*pState)oPemPet ->initialize()

The java method returns the success of the method to the “status” variable. If status is equal to integer value “zero”, then initialization is successful. Then the other user interface buttons are enabled.

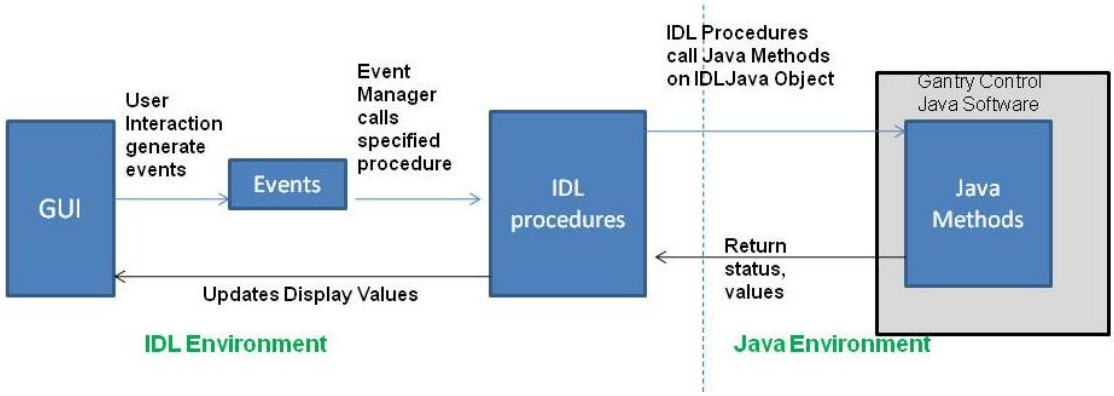

Figure 8: Working of IDL-Java Bridge

IDL and Java use different data types. A integer data type in Java is treated as Long in IDL. IDL handles the data type conversion between IDL and Java. The data types of the parameters that are passed from IDL to Java are mapped to Java data types and the results returned are mapped back to IDL data types.

The OBJ_DESTROY routine is used to instances of java-idl objects.

3.12 Platforms

The tools used in the process of developing the software for the implementation of this work are:

1. J2SE (Java 2 Standard Edition): Java SE Development Kit (JDK) 1.6.4 is installed on the gantry control computer. JDK includes the Java virtual machine (JVM), Java

runtime environment (JRE) and the command line development tools that are useful for developing Java applications. Java swing and Java Beans are used for developing the GUI in java environment. Java RMI is used for remote method calls.

2. Eclipse IDE 3.1.2: Eclipse is an integrated development environment for Java based applications. It comprises extensive application frameworks, tools and runtime libraries for Java based application development. This platform was used to develop the gantry control software.
3. IDL 6.0: IDL 6.0 is an integrated development environment for IDL (Interactive Data language) applications. It comprises the application frameworks and tools for IDL application development. It also includes IDL Virtual Machine for the development and distribution of IDL applications, and the IDL-Java Bridge for employing the power of Java in IDL applications. The software for breast image visualization and analysis of PET and PEM images was developed here. Java methods are called from IDL-Java Import Bridge.
4. OS Platform: We used Windows 2000 professional to run all the above programs. All software developed is platform independent and therefore can be run on any available platform with little or no modifications.

Chapter 4 Results and Discussion

The Figure below shows the complete user interface for the gantry control and Image display of the PEM/PET imaging and biopsy device. The user interface consists of a set of IDL widgets.

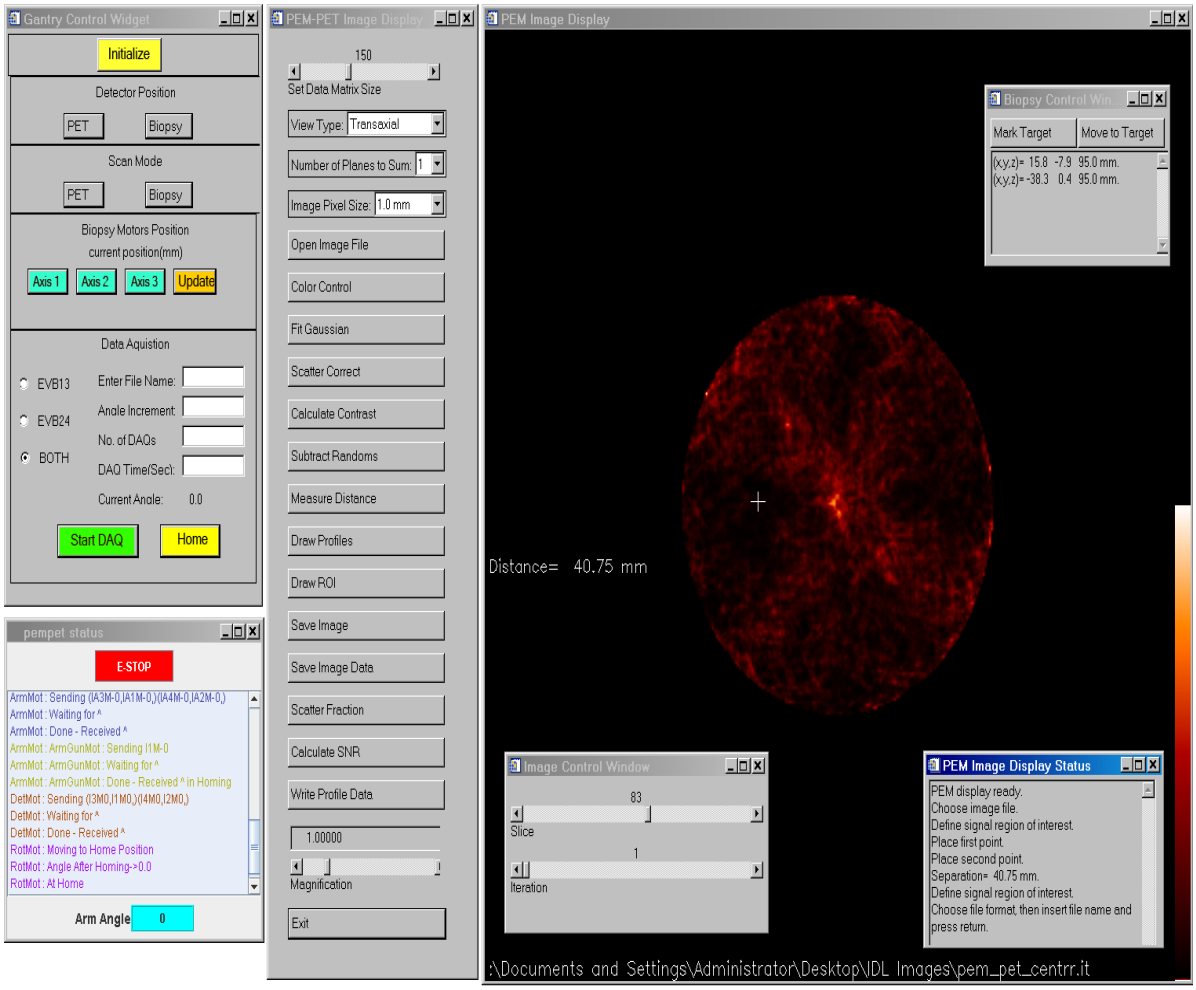

Figure 9: Complete User Interface for PEM/PET device

The "Gantry Control" widget is the GUI for controlling the motion of gantry hardware and initiation of data acquisition. The Gantry Control widget is enhanced by the use of bitmap buttons. The base of the widget houses five child bases. The initialize button is present in the first base. It handles the initialization of the PEM/PET device. The detector position base has the buttons to move the detectors between PET and PEM positions. The scan mode base has buttons to set the device in biopsy scan mode or PET scan mode. The biopsy motors position has the labels and button to display the current position of the arm

motors. The data acquisition base allows you handle initiation of data acquisition. The user can select the event builders by the radio buttons and the give the parameters required for initiation of DAQ in the text fields provided.

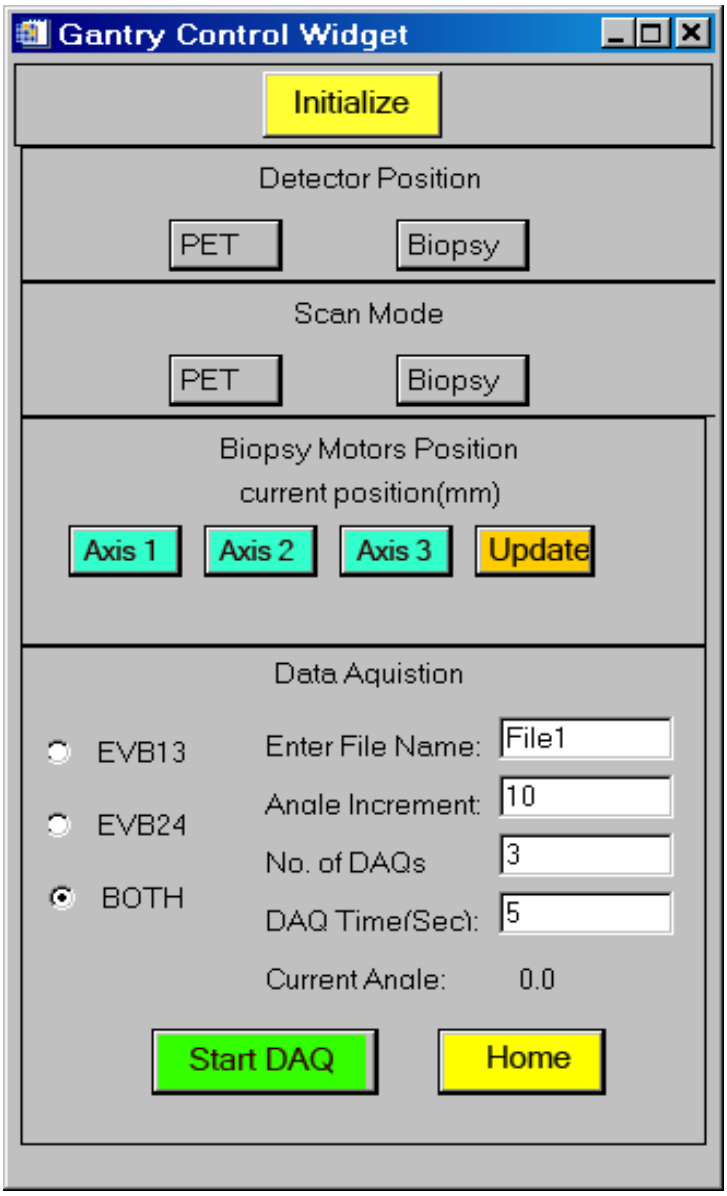

Figure 10: Gantry Control Widget

The biopsy control widget handles the movement of biopsy needle. The mark target button gets the co-ordinates of the point on the PET image of the breast specified by the user. This image is displayed on the PEM Image Display widget. The “move to target” button is activated only with the PEM/PET apparatus are in biopsy position. The user has

to mark the target before moving the needle to the target. When the user clicks the "mark target" button, the generated event calls the underlying procedure to store the co-ordinates of the point specified by the user. The variables that store the co-ordinates are declared as common and are accessible to other procedures that have the same common variable. When the user clicks "move to target" button, the generated event calls the "move arm" procedure. The co-ordinates specified by the user are available to this procedure via a common variable. The move_Arm method on the java class is invoked by this procedure with the co-ordinates as the input parameters. The IDL Java object "oPemPet" is already initiated on which java methods can be called. The syntax for the calling the java method with IDL-Java Import Bridge on the IDL Java object is given below.

Status = (*pState).oPemPet->move_Arm(pointX, pointY)

The status variable stores the success value returned by the java method.

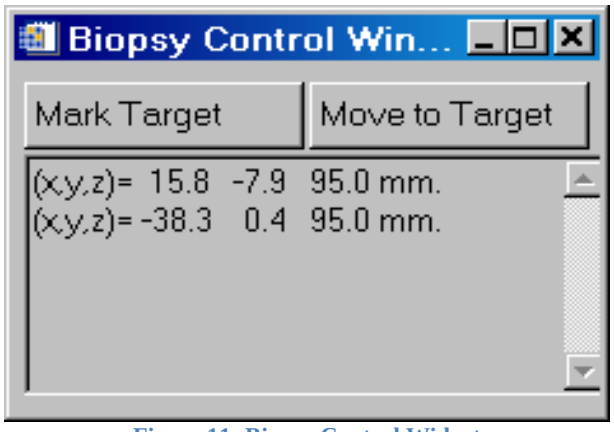

Figure 11: Biopsy Control Widget

The mechanism to display the status of each step in the operation of the PEM/PET device is achieved through a custom write text pane. This text pane is designed with the Java programming language. It is very important for the user to know the status of each step while operating the device. Information about the success or failure of each step is displayed with the necessary action to rectify the error and the corresponding module of the device. Warning messages save the device from damage by malfunction. For

example, a warning message is shown if the user tries to move the detectors to PET position (detector heads at extreme ends) when the device is in biopsy scan mode.

An IDL text widget is designed to display these status messages. It requires frequent transfer of control between IDL and Java and has long delay times. To rectify this problem, a Java text pane was developed and integrated in the constructor of the main class of the java software. The java text pane is initialized when the IDL Java object is created.

The system to monitor the angular position of the biopsy arm has to constantly be updated. This requires the process to be run in a separate thread. To display the Arm angle in the IDL GUI, a separate process has to be created to update the display value constantly. The delay in the updating of angle is very larger compared to a Java process since IDL multithreading does not support multiple processes on a single processor computer. Therefore the arm angle display is embedded in Java with the text pane as a separate frame.

Figure 12: Status Messages Widget

IDL calls a Java method using the IDL-Java Bridge and waits until the Java method returns a value. Meanwhile, there may be something that goes wrong within the Java

method and a need to emergency stop may arise. The Emergency stop on IDL cannot be initiated until the Java method returns a value. A crash in the Java method may never return the control and IDL has to wait forever for the control. To avoid this situation, the emergency stop button is also embedded in the Java.

Chapter 5 Conclusion

5.1 CONCLUSION

The unified user interface for the PEM/PET imaging and biopsy device was designed with Interactive Data Language. The user can interact with this interface to control gantry, initiate data acquisition and visualize acquired images. The software to control gantry and initiate data acquisition was developed as a Java application with methods callable from IDL. The IDL-Java Import Bridge was utilized to establish communication between IDL and Java environments. We are able to control gantry and initiate data acquisition from IDL by calling the methods of Java application within from IDL. The IDL-Java Bridge provided the IDL application with the power of Java to control the PEM/PET hardware and communicate with event builder computers to initiate data acquisition while having efficient image analysis and visualization techniques. The burden of handling two different application, two different user interfaces and manual transfer of information between these two applications is removed. Thus, the use of PEM/PET device has been simplified by creating a single application with a complete user interface for operating the device, which will greatly simplify clinical use of the system.

REFERENCES

1. **wikipedia.** Positron Emission Tomography. *http://www.wikipedia.org.* [Online]
2. *The positron emission mammography/tomography breast imaging and biopsy system (PEM/PET): design, construction and phantom-based measurements.* **Raymond R Raylman, Stan Majewski, Mark F Smith, James Proffitt, William Hammond, Amarnath Srinivasan, John McKisson, Vladimir Popov, Andrew Weisenberger, Clifford O Judy, Brian Kross, Srikanth Ramasubramanian, Larry E Banta, Paul E Kinahan and Kyle Champl.** s.l. : IOP Publishing, January 2008. http://www.iop.org/EJ/article/0031-9155/53/3/009/pmb8_3_009.pdf.
3. **ITT Visual Information Systems.** Overview of Using Java Objects. *http://idlastro.gsfc.nasa.gov/idl_html_help/.* [Online] November 2007. http://idlastro.gsfc.nasa.gov/idl_html_help/Overview_of_Using_Java_Objects.html .
4. **Raymond R Raylman, Stan Majewski, Brian Kross, Vladimar Popav, James Profitt, Mark F. Smith, Andrew G Weisenberger, Randy Wojcik.** *Development of a dedicated positron emission tomography system for the detection and biopsy of breast cancer.* s.l. : Elsevier, September 2006.
5. **Sun Microsystems.** An overview of RMI Applications. *http://java.sun.com.* [Online]
6. **http://www-vis.lbl.gov.** whats new in IDL 6.0. *http://www-vis.lbl.gov/NERSC/Software/idl/help/docs6.0/whatsnew.pdf.* [Online]
7. **http://idlastro.gsfc.nasa.gov.** What is a Bridge? *http://idlastro.gsfc.nasa.gov/idl_html_help/What_Is_a_Bridge.html.* [Online]
8. **http://idlastro.gsfc.nasa.gov.** Overview of using java objects. *http://idlastro.gsfc.nasa.gov/idl_html_help/Overview_of_Using_Java_Objects.html .* [Online]