**Detailed Design & Implementation**

**Course:** CS 474 Object-Oriented Design

**Project Name:** RADIS

**Team Members**

Andrew McClellan

Austin Traverse

Kelsey Hilton

**Professor:** Rob Byrd, iSchool, Abilene Christian University

**Last updated:** 3 May 2012

**Document Revision Number:** 2

|  |  |
| --- | --- |
| **Detailed Design and Implementation** | |
| **UML** | **% Done** |
| 1. Final UML .pngs |  |
|  |  |
| **Source Code** | |
| 1. Header Files |  |
| 1. Main |  |
| **Test Data** | |
| 1. Explanation |  |
| 1. Images |  |
| 1. Patient (user input) |  |
| **User’s Guide** | |
| 1. Installation Guide |  |
| 1. Tutorial on how to implement the API |  |
| 1. Sample Application |  |
| 1. Doxygen Documentation |  |
| **Appendices** | |
| * 1. dA Function Detail |  |
| * 1. Functions Inherited from CVIP |  |

**UNIFIED MODELING LANGUAGE**

1. **Final UML .pngs**

**SOURCE CODE**

1. **dA Header Files**
2. **Main**

**TEST DATA**

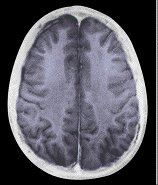
1. **Images**

The main image processing technique for this project was comparing threshold values of different images. A base threshold value for the type of image was determined by finding the average of the means of selected images. The base cases and images used for preprocessing are below:

Brain

Average Threshold Value: 128.916

Base-Case Images:



Lung

Average Threshold Value: 128.916

Base-Case Images:

Chest

Average Threshold Value: 147.5

Base-Case Images:

1. **Patient: Example User Input**

**USER’S GUIDE**

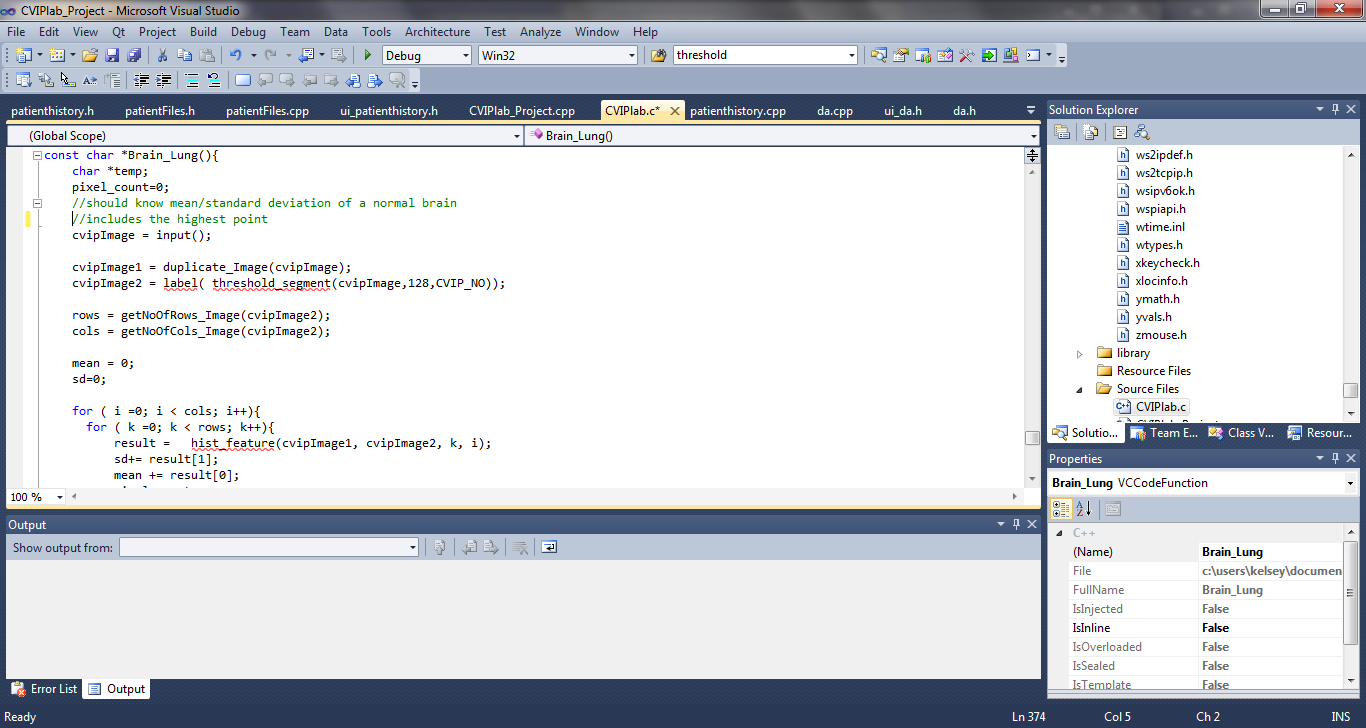
1. **Installation Guide**

It is assumed that you are working on a Windows platform.

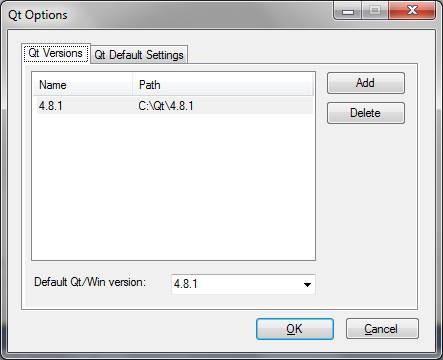
1. If you don’t already have Microsoft Visual Studio Professional Edition 2010 you will need to either, purchase it, or download a trial edition from [www.microsoft.com/visualstudio](http://www.microsoft.com/visualstudio).
2. Download and install *both* Qt Visual Studio Add-In *and* Qt libraries 4.8.1 for Windows from <http://qt.nokia.com/downloads>

NOTE: does not work with the Express Edition of Visual Studio

1. Now, start Visual Studio, click on “Qt” on the toolbar and select “Qt options” from the drop-down menu.



1. On the Qt Versions tab click “ADD”.



1. Put a version number and the path where Qt is. e.g. C:\Qt\4.8.1
2. Click “OK”
3. Done. Qt should be integrated with Visual Studio 2010.
4. **Tutorial**

Prerequisites:

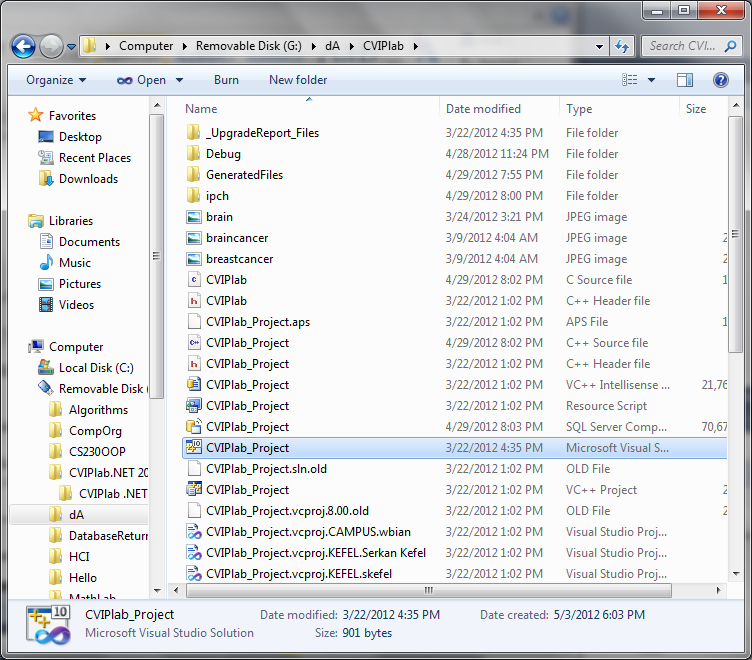
Microsoft 2010 Visual Studio Professional Edition

Qt add-in for Visual Studio

Qt 4.8.1

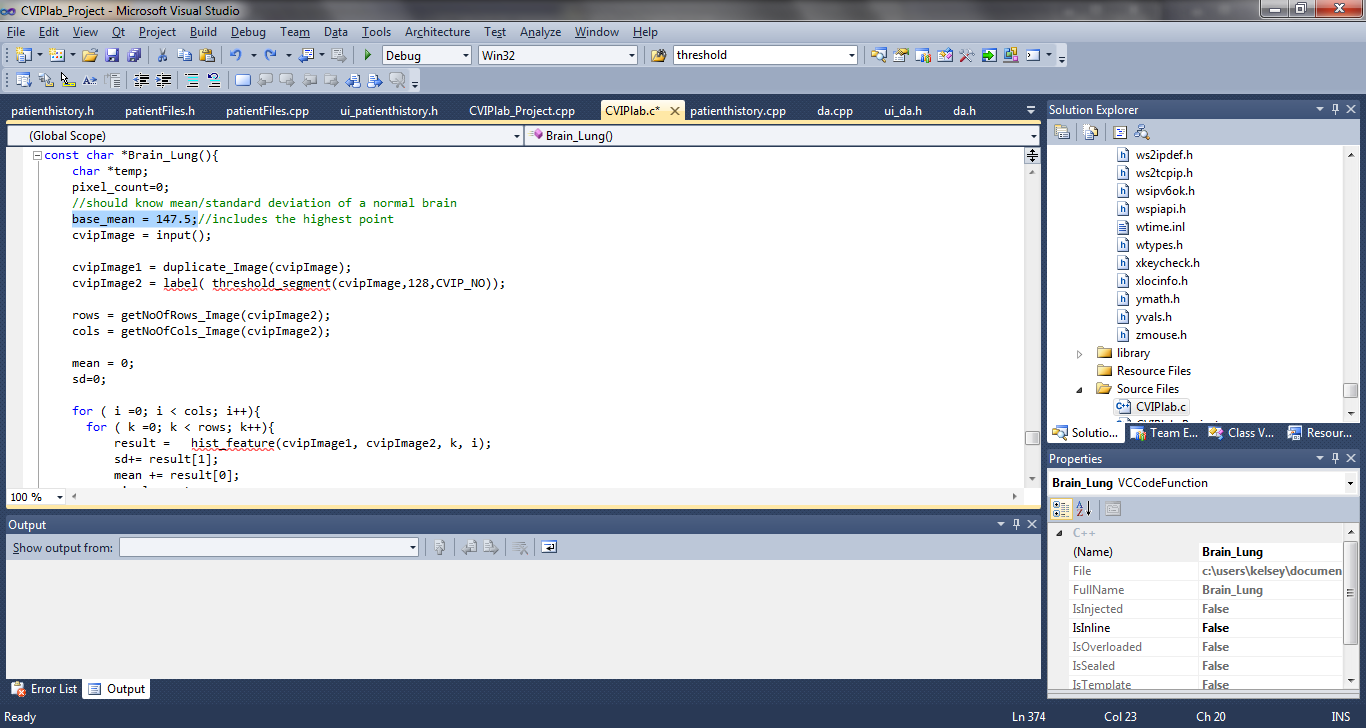
If you do not meet the prerequisites see the Installation Guide.

1. Open the folder named dA. Inside, there should be 3 folders; “include”,”lib” and “CVIPlab”. Open CVIPlab.
2. Inside CVIPlab you will find a conglamoration of files and folders. The number of Visual Studio files can be confusing. Select CVIPlab\_project with the Type Microsoft Visual Studio Solution.



The images included in the CVIPlab folder are the images that were used to create the base cases if you are interested.

1. Double click on the selected file. This should open Visual Studio automatically. If it does not, open Visual Studio manually. On the pull down menu, choose “open” and search for CVIPlab\_Project in the dA folder.
2. Once the project is open you will be able to see the source code for both the application and the underlying API.
3. To compile and run the project hit F5 on the keyboard or press the green play button on the top toolbar in Visual Studio.



1. **Sample Application**
2. **Doxygen Documentation**

**APPENDIX A**

**dA Function Definitions**

Image **\*input**( )

Image **\*threshold\_Setup**(Image \*inputImage)

<inputImage> image to be scanned in jpg format

const char **\*Brain\_Scan**( )

const char **\*Chest\_Scan**( )

const char **\*Lung\_Scan**( )

See below for full details for each dA function.

**NAME**

threshold\_Setup **–** prepares for threshold process

**SYNOPSIS**

Image \*threshold\_Setup(Image \*inputImage)

< inputImage > pointer to an image

**PATH**

$dA\Scans.h

**DESCRIPTION**

Allows the user to enter a threshold value. After it gets the threshold value, it will call the threshold\_Image() function to complete the threshold process.

Thresholding is the process of converting a grayscale image to a binary image. Once a certain threshold value has been chosen, the image is scanned. Any pixel with a value above the threshold value is labeled with a 1 (white) and any pixel with a value below the threshold is labeled with a 0 (black). Algorithms to determine an appropriate threshold value exist, but in this case the user is allowed to choose an arbitrary value.

**RETURN VALUES**

Image

**EXAMPLE**

int main(){

Image \*threshold\_Setup(Image \*inputImage){

unsigned int threshval; /\* Threshold value \*/

//Gets a value between between 0 and 255 for threshsold value

print\_CVIP("\n\t\tEnter the threshold value: ");

threshval = getInt\_CVIP(10, 0, 255);

return threshold\_lab(inputImage, threshval);

}

**}**

**NAME**

Input – input an image and convert to CVIPImage type.

**SYNOPSIS**

Image \*input( )

**PATH**

$dA\Scans.h

**DESCRIPTION**

Takes a file name, opens an image and converts the image to CVIPImage type so that it is compatible with CVIP functions.

**RETURN VALUES**

Image

**EXAMPLE**

int main(){

Image\* input(){

char \*inputfile;

Image \*cvipImage;

print\_CVIP("\n\t\tEnter the Input File Name: ");

inputfile = getString\_CVIP();

//creates the CVIPtools Image structure from the input file

cvipImage = read\_Image(inputfile, 1);

if(cvipImage == NULL) {

error\_CVIP("init\_Image", "could not read image file");

free(inputfile);

return NULL;

}

//display the source image

view\_Image(cvipImage, inputfile);

//IMPORTANT: free the dynamic allocated memory when it is not needed

free(inputfile);

return cvipImage;

}

}

**NAME**

Scan

**SYNOPSIS**

intScan( )

**PATH**

$dA\Scans.h

**DESCRIPTION**

Compares an input image of a brain to an existing threshold value determined by the average of the mean threshold values of several “normal” images of a brain. If the average threshold value of an input image falls within a certain range of the pre-determined threshold value for that particular type of image, Scan returns a value which indicates that the image is “normal”. Otherwise the input image is classified as “abnormal”.

**RETURN VALUES**

char value indicating whether or not an input image is within a certain range of the overall average of the threshold value.

**EXAMPLE**

int main( ){

int Scan(){

char \*temp;

pixel\_count=0;

//should know mean/standard deviation of a normal brain

base\_mean = 128.916;//includes the highest point

cvipImage = input();

cvipImage1 = duplicate\_Image(cvipImage);

cvipImage2 = label(threshold\_segment(cvipImage,128,CVIP\_NO));

rows = getNoOfRows\_Image(cvipImage2);

cols = getNoOfCols\_Image(cvipImage2);

mean = 0;

sd=0;

for ( i =0; i < cols; i++){

for ( k =0; k < rows; k++){

result = hist\_feature(cvipImage1, cvipImage2, k, i);

sd+= result[1];

mean += result[0];

pixel\_count++; }

}

//find average

sd = sd/pixel\_count;

mean = mean/pixel\_count;

mean = mean+sd;

//compare base to input

if(mean > base\_mean)

temp = "abnormal";

else

temp = "normal";

return temp;

}

}

**APPENDIX B**

**Functions Inherited from CVIP**

The following CVIP functions were used in the development of our dA API:

void **display\_image**(const char\*image\_name, IMAGE\_FORMAT format)

<image\_name> name of the image file

<format> the image format

void **view\_image**(Image \*inputImage, char \*imageName)

<inputImage> pointer to the input image

<imageName> character string as the image name in the display window

double **\*hist\_feature**(image \*originalImage, \*labeledImage, int r, int c)

<originalImage> pointer to the original image

<labeledImage> pointer to the labeled image

<r> row coordinate of a point on the labeled image

<c> column coordinate of a point on the labeled image

Returns 5 histogram feautres: mean, standard deviation, skew, energy and entropy

Image **\*read\_Image**(char \*filename, int showmessages)

<filename> - pointer to an character string containing the file name

<showmessages> - show messages

unsigned **GetNoOfCols\_Image**(Image\* image)

<image> pointer to an image

unsigned **GetNoOfRows\_Image**(Image\* image)

<image> pointer to an image

Image \***duplicate\_Image(**const Image \*a)

<a> pointer to an image

Image **\* label**( const Image \*imageP )

<imageP> - pointer to an Image

Image **\*threshold\_segment**(Image \*inputImage, unsigned int

threshval, CVIP\_BOOLEAN thresh\_inbyte)

<inputImage> - pointer to Image structure

<threshval> - threshold value

<thresh\_inbyte>

- CVIP\_NO apply threshval directly to image data;

- CVIP\_YES threshval is CVIP\_BYTE range; remap to

image data range before thresholding.