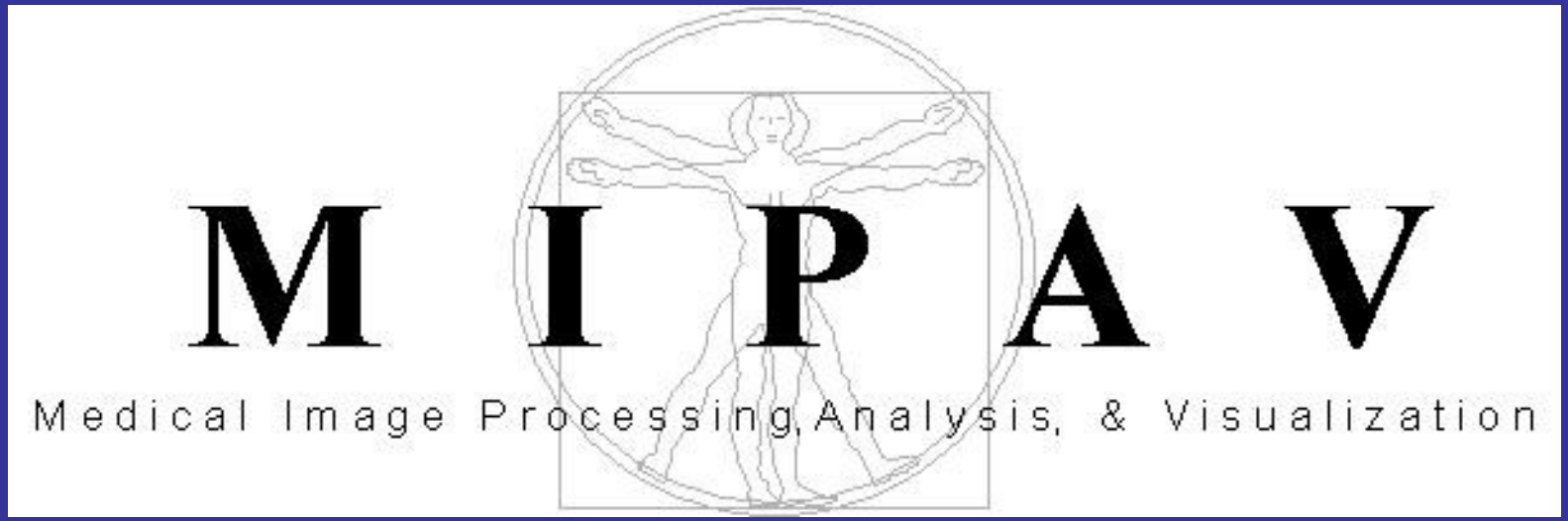
Plugins



http://mipav.cit.nih.gov

Medical Image Processing, Analysis & Visualization & Plugins

Evan McCreedy

email: mccreedy@mail.nih.gov

Biomedical Imaging Research Services Section (BIRSS)

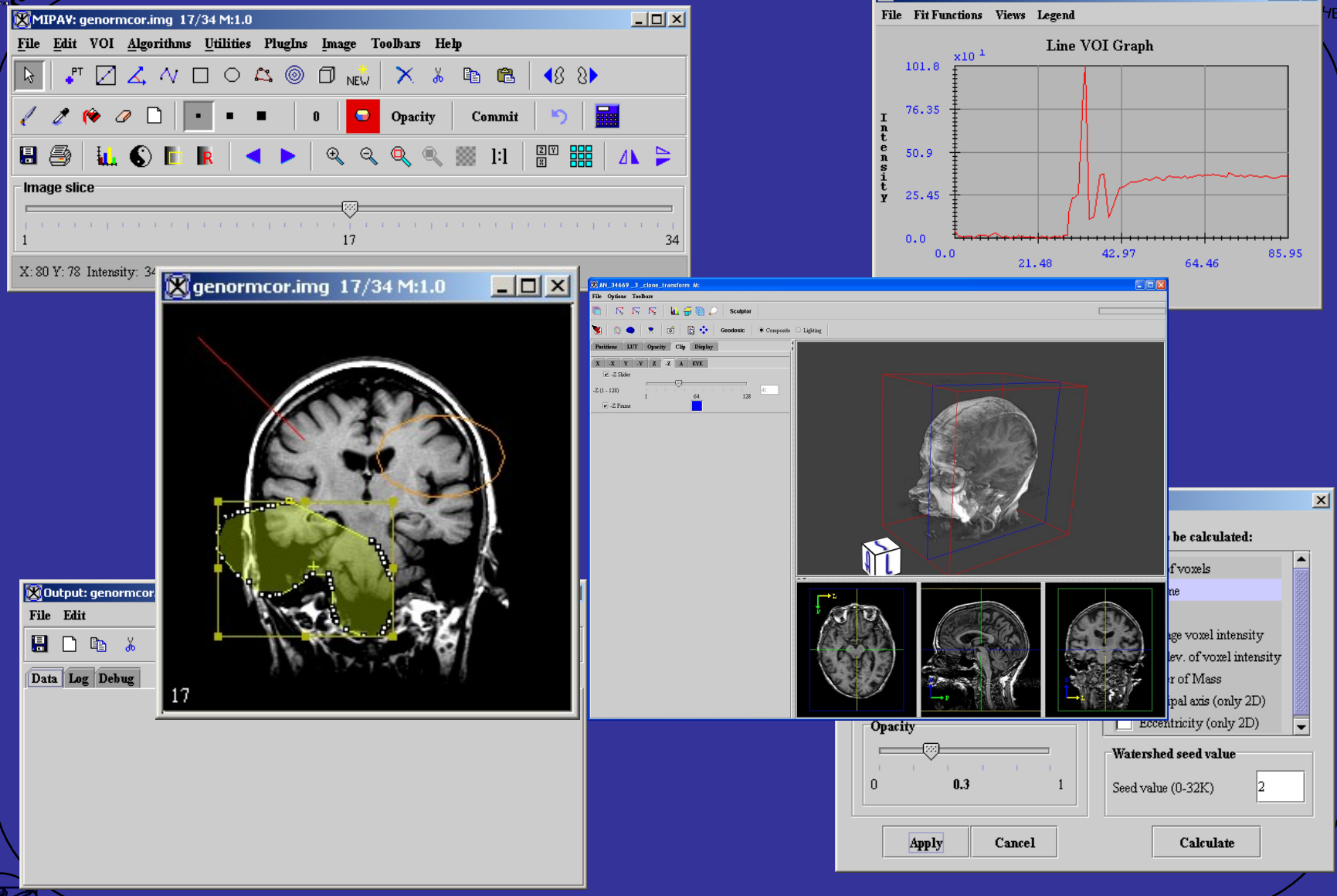
Imaging Sciences Laboratory

Division of Computational Bioscience

Center for Information Technology

(301) 496-3323

http://mipav.cit.nih.gov


MIPAV: genormcor.img 17/34 M:1.0
File Edit VOI Algorithms Utilities PlugIns Image Toolbars Help
Opacity
Commit
Image slice
1
17
34
X: 80 Y: 78 Intensity:
Line VOI Graph
File Fit Functions Views Legend
Line VOI Graph
x10 1
101.8
76.35
50.9
25.45
0.0
Intensity
0.0
21.48
42.97
64.46
85.95
genormcor.img 17/34 M:1.0
17
Output: genormcor
File Edit
Data Log Debug
be calculated:
f voxels
ge voxel intensity
lev. of voxel intensity
r of Mass
pal axis (only 2D)
Eccentricity (only 2D)
Opacity
0
0.3
1
Watershed seed value
Seed value (0-32K)
2
Apply
Cancel
Calculate

Functional Overview

GUI

Views – with data fusion
2D planar,
"Lightbox",
Cine (movie),
Multi-planar,
3D tri-planar,
Surface render, (supports 3D texture mapped volume rendering
Volume render

VOIs
32K
Manual and automated contouring

Algorithms
Filtering
Segmentation/classification
Measurement/quantification
Registration/fusion
Utilities
Plugins

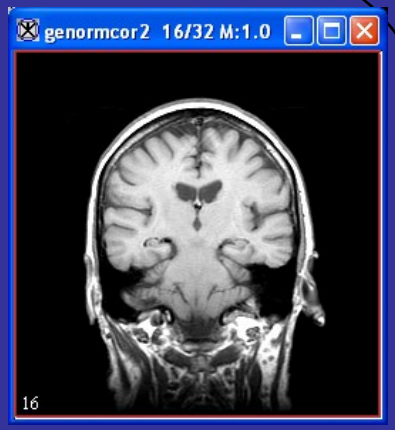
Scripting & Plugs

Data (Image) types: n-dimensional structure
(boolean, byte, unsigned byte, short,
unsigned short, int, long, float, double, Complex, ARGB)

PACS
DICOM 3.0:
Query/Retrieve, Catcher

File types
(Raw, Analyze, DICOM 3.0, GE, Siemens, Bruker, Interfile, Micro cat, MINC, MRC, FITS, Cheshire, AFNI, TIFF, JPEG, GIF, BMP, AVI, QuickTime, Biorad, Ziess LSM510, XML, and more)

Code Snapshot



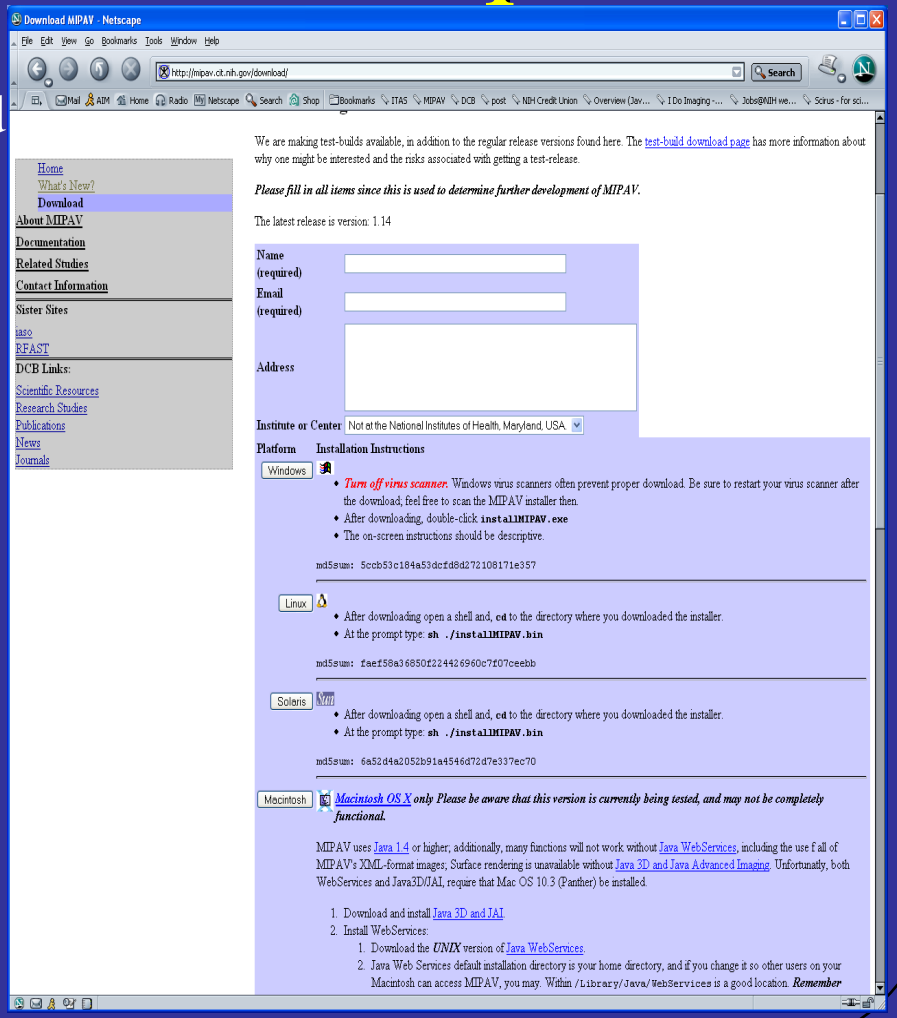

```
int destExtents[] = new int[2];
destExtents[0] = image.getExtents()[0];    // X dim
destExtents[1] = image.getExtents()[1];    // Y dim

// Make a result image of Unsigned byte type
resultImage     = new ModelImage(ModelStorageBase.UBYTE, destExtents, “Result Image”, null);

int length = destExtents[0] *  destExtents[1];
for (int i = 0; i < length; i++){
    destImage.set(i, i%256);
}

ViewJFrameImage imageFrame;
ModelLUT LUTa   = new ModelLUT(ModelLUT.COOLHOT, 256, dimExtentsLUT);
        imageFrame = new ViewJFrameImage(resultImage, LUTa, new Dimension(610,200), userInterface);
```

Major Algorithms Supported

- **Algorithms**
  - Filters: Gaussian blurring, gradient magnitude, Laplacian, curvature, other higher order derivatives, median, anisotropic diffusion, coherence-enhance diffusion, isotropic diffusion, wavelet, unsharp masking, etc.
  - Image Calculator (add, subtract, multiply, divide, AND, OR, XOR)
  - Registration
    - Landmark – least squares, thin-plate splines for both 2D and 3D datasets
    - AIR 5.07 and AFNI
    - Automatic 2D/2.5D/3D registration intra/inter patient, intra/intra modality
      - multi-resolution, user selectable DOF, user selectable cost function (correlation ratio, normalized cross correlation, least squares, mutual information.
  - Image transformations or resample with nearest neighbor, tri-linear, 3rd, 4th bSpline, Sinc, 3rd, 5th, 7th order Lagrangian, etc.
  - Surface extraction with decimation
    - Adaptive skeleton climbing
    - Marching cubes
    - Marching tetrahedrons
  - Skull striping
  - Classification – Fuzzy c-means
  - Watershed
  - Morphological filters (open, close, erode, dilate, etc)
  - Active contour methods (GVF etc.)

Download and Setup

1. http://mipav.cit.nih.gov/download
2. Fill in form
3. Install (e.g. installMIPAV.exe)



Scripting & Plugins

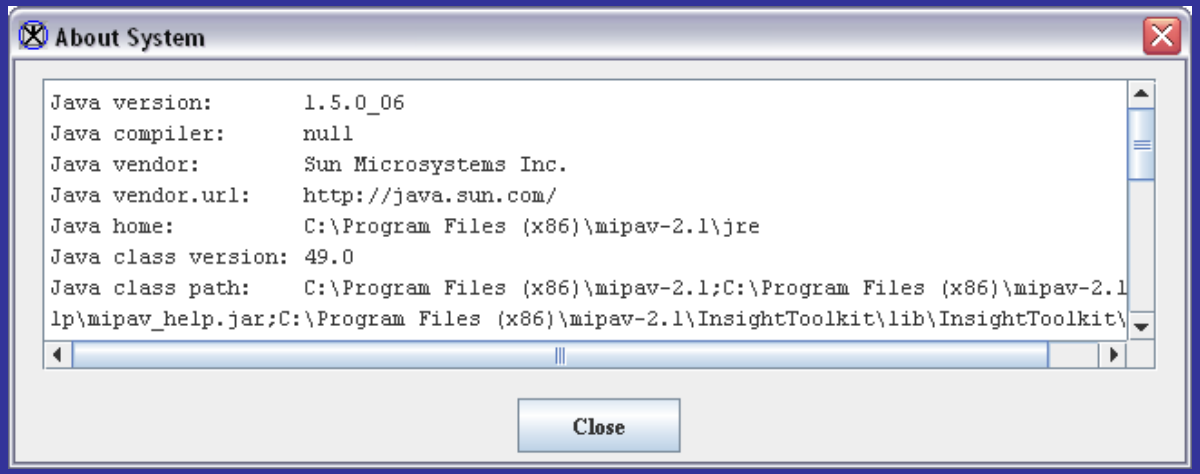
- Scripts - Automation of MIPAV functions applied to a group of datasets to increase efficiency and productivity
- Plugins – Ability to add unique functionality to MIPAV by developing new algorithms using MIPAV's API

Plugins

- Pure Java/MIPAV plugin
  - Unique functionality.
- MIPAV to C, system call, Python, Perl etc.
- MIPAV to ITK (a C++ imaging library).

Plugins

- Four steps to building a plugin
  - 1. Determine type of plugin (usually Algorithm)
  - 2. Determine Java version to use
    - Use the same version javac as MIPAV (Help -> About Java)
  - 3. Write the plugin
  - 4. Compile the plugin



Plugins

- PlugInAlgorithm – Develop new functionality and ability to call functions already in MIPAV.
- PlugInFile – Develop files readers to support unique file formats. Rarely used since MIPAV supports numerous file formats.
- PlugInGeneric – Generic plugins that do not require an open image
- PlugInView – Develop new visualizations of datasets.

Plugins

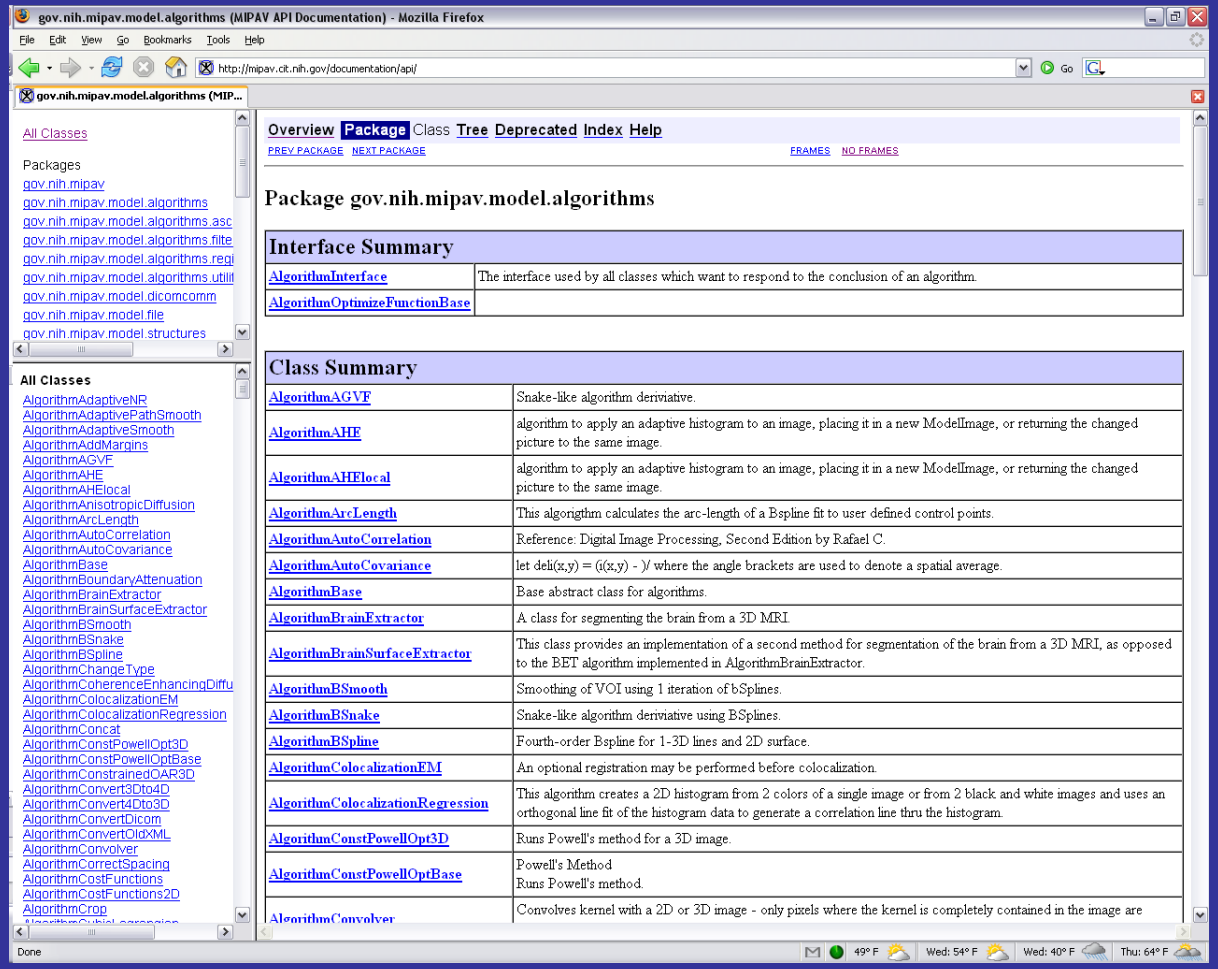
- To build a plugin, three files are typically required:
  - PlugInFoo.java
    - interface to MIPAV and the plugin
  - PlugInDialogFoo.java
    - invokes the dialog to get user supplied parameters – can be hidden when no parameters are required
  - PlugInAlgorithmFoo.java
    - The actual algorithm to be implemented. It can be a mixture of calls to MIPAV's API, C programs, Perl, ITK, etc.

Plugins - Location

- Stored for each user in $home/mipav/plugins
  - Windows - C:\Documents and Settings\user_name\mipav\plugins
  - Unix/Mac - usually /home/user_name/mipav/plugins
  - Note: By default, the home directory of the user who installed MIPAV is used. If you are running plugins as a different user, set the classpath before running mipav:
    - % CLASSPATH=$HOME/mipav/plugins/ /path/to/mipav

Plugins - MIPAV API

http://mipav.cit.nih.gov/documentation/api/

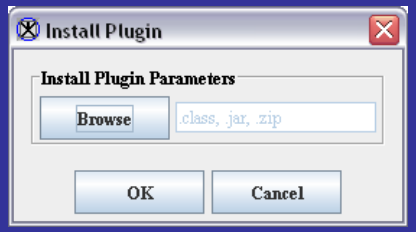


Plugins - Documentation

- Plugin development documentation
  - http://mipav.cit.nih.gov/documentation/userguide/volume1/MIPAV_PlugIns.pdf
- Using ITK from MIPAV
  - http://mipav.cit.nih.gov/documentation/techguides/TechGuide2_UsingInsightToolkit.pdf
- This presentation will be posted at http://mipav.cit.nih.gov/documentation/presentations/ in the next couple of weeks.

Plugins – Installing

- Installation of a simple plugin (Plugins -> Install Plugin)



Copies files

- Chosen .class files, or the contents selected of jar/zip files

into the directory

- $home/mipav/plugins

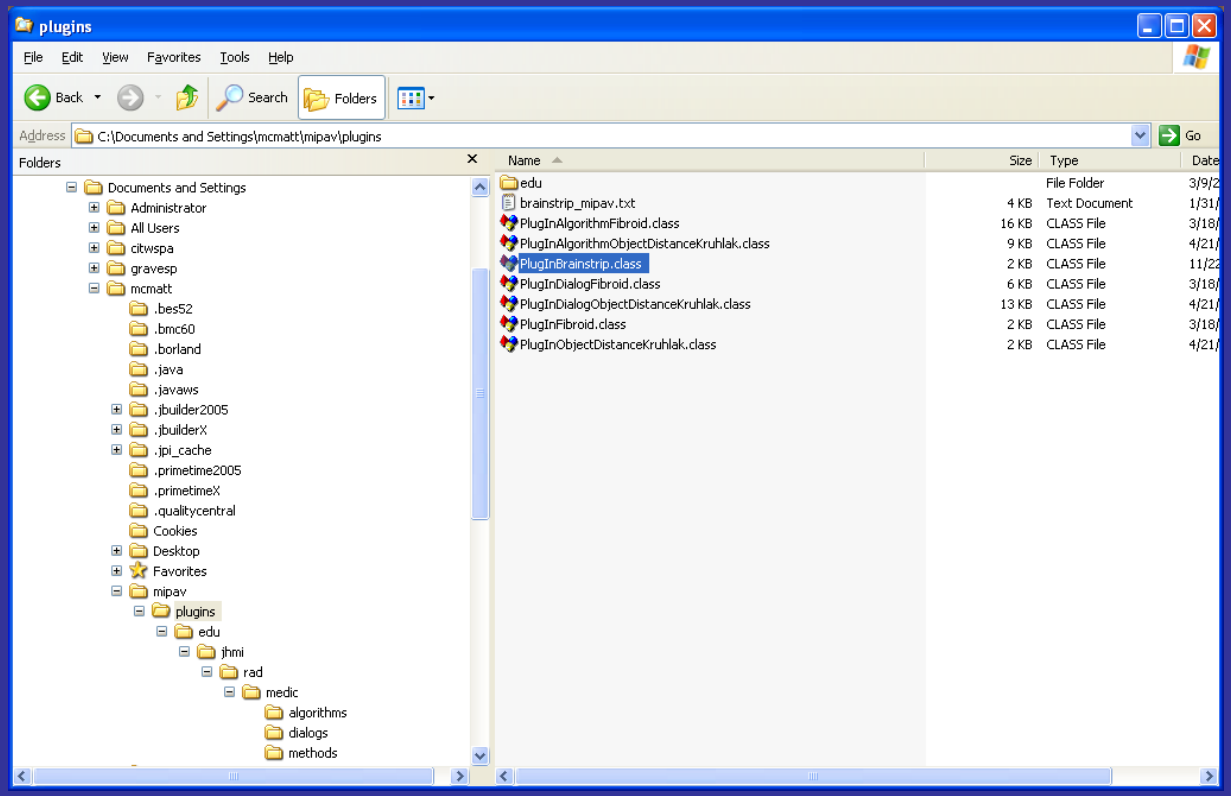
When MIPAV starts, it parses this directory and builds the plugins menu.

Note:

Simply use the OS's tool for coping the files into the directory listed above is also acceptable. The user will be required to restart MIPAV to have the new plugin appear in the menu.

Plugins – Installing

- Installation of a complex plugins
  - Medic Talairach plugin



Plugins – Installing Lab

- Install
  - PlugInCT_MD.class
  - PlugInDialogCT_MD.class
  - PlugInAlgorithmCT_MD.class

Medic Plugins

http://medic.rad.jhmi.edu/download/public/index.shtml

The Medic plugins are being ported to be compatible with the 3.0.0 release of MIPAV. 3.0.0-compatible versions of their plugins are expected soon.

Plugins

```
package gov.nih.mipav.plugins;

import gov.nih.mipav.model.structures.*;
import gov.nih.mipav.view.*;

import java.awt.*;


public interface PlugInAlgorithm extends PlugIn {

    /**
    *   run
    *   @param UI              MIPAV main user interface.
    *   @param parentFrame     Frame that displays the MIPAV image.
    *                          Can be used as a parent frame when building dialogs.
    *   @param image           Model of the MIPAV image.
    *   @see   ModelImage
    *   @see   ViewJFrameImage
    */
    public void run (ViewUserInterface UI, Frame parentFrame, ModelImage image);

}
```

Plugins – Hello World

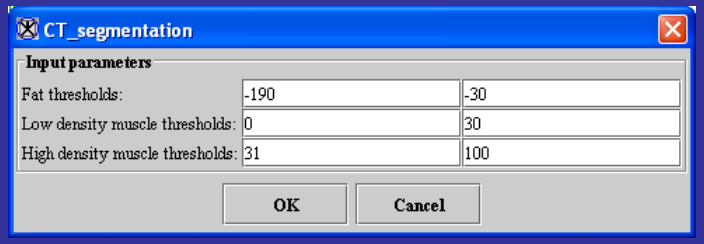
```
import gov.nih.mipav.plugins.*;          //needed to load PlugInAlgorithm / PlugInView / PlugInFile interface
import gov.nih.mipav.view.*;
import gov.nih.mipav.model.structures.*;

import java.awt.*;
import javax.swing.*;
```

Plugins - Example_

```
import gov.nih.mipav.plugins.*;                //needed to load PlugInAlgorithm / PlugInView / PlugInFile interface
import gov.nih.mipav.view.*;
import gov.nih.mipav.model.structures.*;

import java.awt.*;

/**
*   This is simple plugin for the University of Maryland to simple segment an image based on CT Hounsfield units
*
*   @see PlugInAlgorithm
*/

// This is a Algorithm type of PlugIn, and therefore must implement PlugInAlgorithm
// Implementing the PlugInAlgorithm requires this class to implement the run method
// with the correct parameters
public class PlugInCT_MD implements PlugInAlgorithm {

  /**
     *   Defines body of run method, which was declared in the interface.
     *   @param  UI               User Interface
     *   @param  parentFrame  Parent frame
     *   @param  image            Current ModelImage - this is an image already loaded into MIPAV. Can be null.
     */
     public void run (ViewUserInterface UI, Frame parentFrame, ModelImage image) {

        if (parentFrame instanceof ViewJFrameImage)
           new PlugInDialogCT_MD (parentFrame, image);

        else
           MipavUtil.displayError ("PlugIn CT_MD only runs on an image frame.");
       }
   }
}
```

Plugins - Dialog

PlugInDialogCT_MD.java

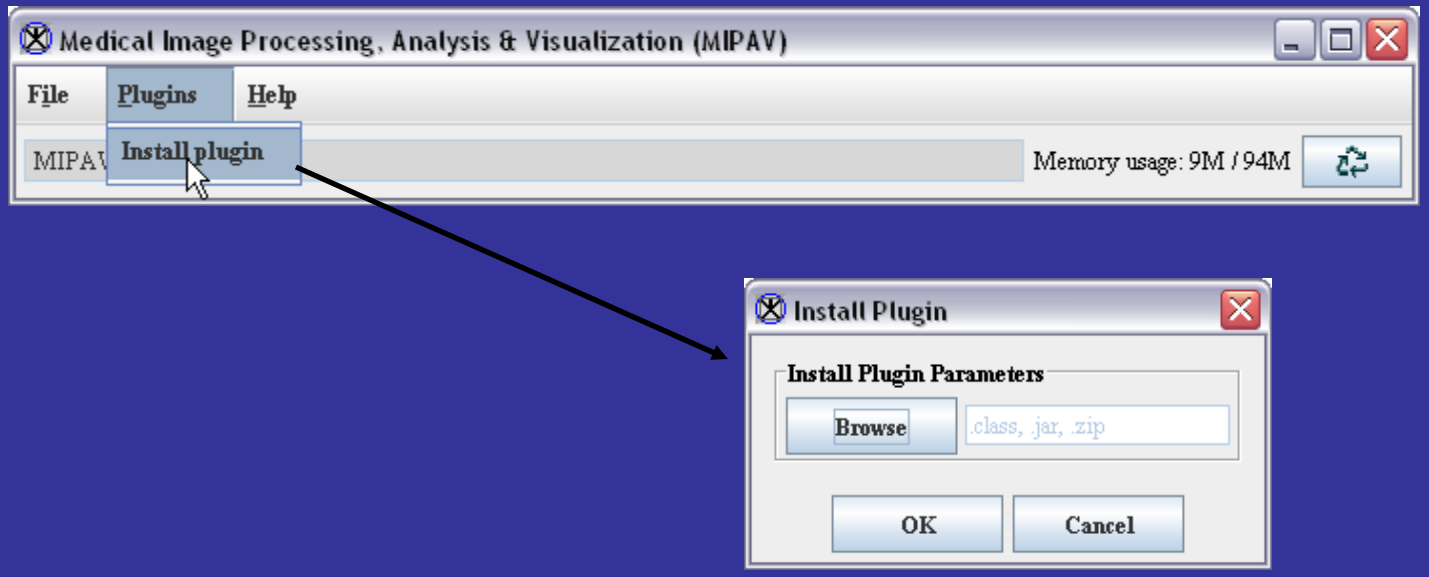


Plugins - Algorithm

PlugInAlgorithmCT_MD.java

Simple algorithm to segment an image based on Hounsfield values

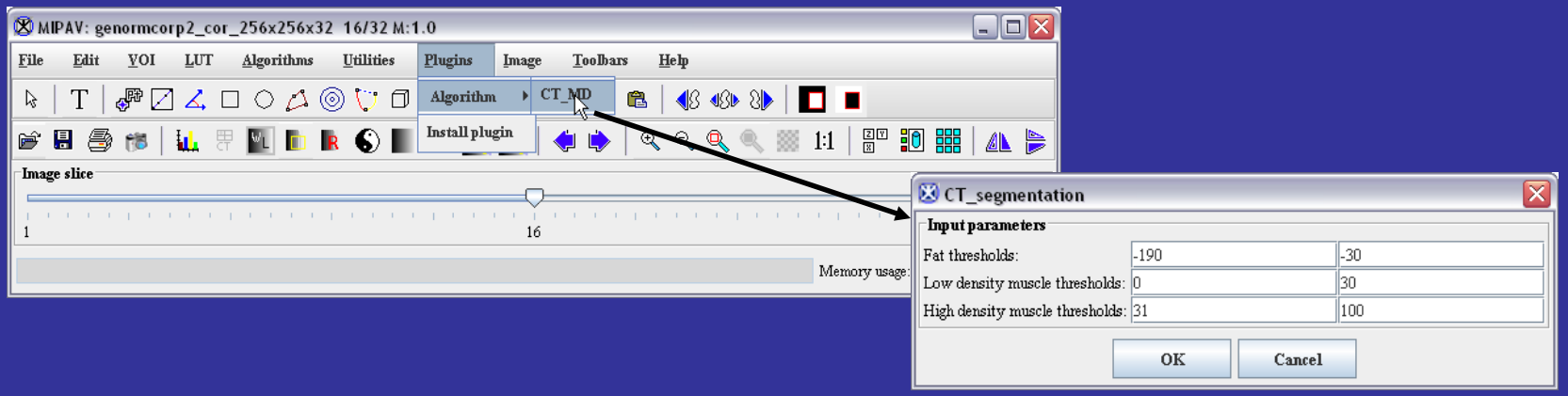
Plugins - Installing

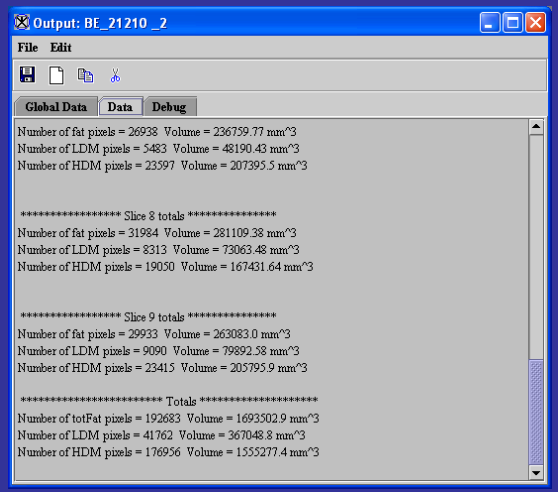


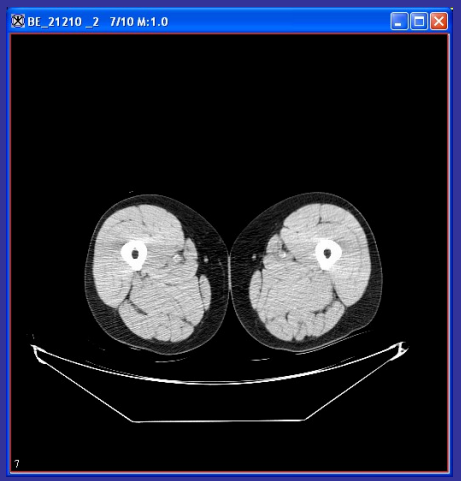
Be sure to select the files to be moved into the user's directory (e.g. PlugInFoo.class, PlugInDialogFoo.class and PlugInAlgorithmFoo.class).

** This is nearly the same as manually coping the files into the user's directory.

Plugins - Executing







Plugins to system calls

```
if (os != Preferences.OS_MAC) {
        int response = JOptionPane.showConfirmDialog(this, "Restart MIPAV to apply memory changes?",
                                "Restart needed",
                                JOptionPane.YES_NO_OPTION, JOptionPane.INFORMATION_MESSAGE);
         if (response == JOptionPane.YES_OPTION) {
              if (os == Preferences.OS_WINDOWS) {
                        // "./mipav.exe" could be replaced with a precompiled c++  program for example.
                        Runtime.getRuntime().exec("./mipav.exe");
              }
              else {
                        Runtime.getRuntime().exec("../runmipav");
              }
              System.exit(0);
         }
}
```

Setting up development environment

- To build a new plugin for MIPAV, the user must first install a build environment, alter the PATH environment variable, and compile the plugin files.
  - Install the Java 2 SDK version 1.5.
  - Install Apache Ant 1.6.X.
  - Add JAVA_HOME to your environment, pointing to the directory where the SDK is installed (e.g. C:\Program Files\Java\jdk1.5.0_06).
  - Add ANT_HOME to your environment, pointing to the directory where Ant is installed (e.g. C:\Program Files\apache-ant-1.6.3).
  - Add the bin directories under ANT_HOME and JAVA_HOME to your PATH environment variable (e.g. add - %JAVA_HOME%\bin;%ANT_HOME%\bin to the end of PATH).

Setting up development environment (continued)

- Retrieve our example Ant build file and place it in the same directory as the plugin .java files you want to compile.
- Alter the dir.mipav and dir.jdk properties within the build.xml to point to the directory where MIPAV and the SDK are installed, respectively.
- Type in "ant compile" (without quotes) into a terminal window (e.g. cmd on Windows or xterm on unix platforms).
- BUILD SUCCESSFUL should appear at the end of the Ant output.
- Copy the .class files that Ant produced into MIPAV's plugin directory (C:\Documents and Settings\username\mipav\plugins or /home/username/mipav/plugins).
- Install the plugin file using MIPAV (Plugins -> Install Plugin).

Build XML example

```
<!-- build file for MIPAV plugin class -->
<project basedir="." default="compile" name="mipav_plugin">
    <property name="dir.mipav" value="c:\\Program Files\\mipav\\"/>
    <property name="dir.jdk" value="c:\\Program Files\\Java\\jdk1.5.0_06"/>
    <target name="init">
        <tstamp/>
        <path id="build.classpath">
            <pathelement path="${dir.mipav}"/>
            <pathelement location="${dir.mipav}/InsightToolkit/lib/InsightToolkit/InsightToolkit.jar"/>
            <fileset dir="${dir.mipav}">
                <filename name="*.jar"/>
            </fileset>
        </path>
        <property name="build.cp" refid="build.classpath"/>
    </target>
```

Build XML example (cont)

```
<target name="compile" depends="init">
        <echo>classpath: ${build.cp}</echo>
        <javac debug="true" deprecation="true" description="Builds MIPAV" verbose="no“
                listfiles="yes" nowarn="no" fork="true" memoryInitialSize="220M“
                memoryMaximumSize="1000M" id="mipav build" source="1.4“
                target="1.4" destdir="." srcdir="." compiler="modern">
          <classpath refid="build.classpath"/>
        </javac>
    </target>
    <target name="clean" depends="init">
        <delete>
          <fileset dir=".">
            <include name="**/*.class"/>
          </fileset>
        </delete>
    </target>
</project>
```

Modify Plugin Lab

- Edit PlugInHello.java
  - Change the message from “Hello World!” to:
    “Hello ” + image.getImageName() + “!”
- Compile PlugInHello.java using the plugin build.xml
- Install the generated PlugInHello.class into MIPAV
- Open an image
- Run the Hello plugin

Summary

- Plugins
  - Unique functionality using MIPAV API
  - Can be used to call C, System, Python, JPython, Perl, etc. programs

MIPAV Team

William Gandler,

Ruida Cheng,

Evan McCreedy

Contractors

David Eberly, Geometric Tools Inc.

Ben Link, SAIC

Nish Pandya, SSAI

Mayur Joshi, MSD

Olga Vovk, MSD

MIPAV

Medical Image Processing, Analysis, & Visualization