The *SIMBA* Image Management and Analysis System

Application Programmers Interface V9, September 13, 2012

A. P. Reeves, A. M. Biancardi

Abstract

The SIMBA Image Management and Analysis System provides an interactive web-based facility to meet the needs of research projects that involve image data. This report outlines the SIMBA application programmer's interface (SAPI) that provides a customizable image analysis environment for specific projects.

1. Introduction

A SIMBA Application Module (SAM) is self-contained system of programs and support files that employ and conform to the SIMBA application programmers interface (SAPI). The SAPI supports the custom development of image analysis functions that can be applied to images in the SIMBA web system and facilitates the review of the results through the SIMBA web system. The SAM consists of a number of programs, developed by the user to support a custom image analysis and may be associated with different projects in the SIMBA system. At the core of the SAM is a program called "*compute*" that manages the custom analysis of the image data. This *compute* program may be developed independent of the SIMBA system. There are both front-end (web-based) and back-end (script-based) support to apply the ***compute*** operation to images in the SIMBA system. For either of these schemes a configuration file ***exper.cfg*** must be created to specify the details of the user facility to the SIMBA system. The *compute* program and exper.cfg comprise the two required elements for a SAM.

There are many additional features that are supported by the SAPI. These include: the setting of parameters for the customized analysis, customized web-displays of the analysis results; support for multiple instances of an analysis on a single image with different parameter settings. Furthermore, there is support for on-line documentation for the module, a method for gathering results for a set of images and support for multiple modules for a given SIMBA project. SAMs may access and analyze images from the two primary image SIMBA image archives (uploaded and DICOM) it may also generate new images for the derived image database for detailed image viewing.

From the web users viewpoint the SIMBA system currently supports two image databases: an ***upload database*** that can accept many different image formats and a ***DICOM database*** that has additional features for only DICOM formatted images and can received data directly from a medical PACS system. All images that are to be processed from either database are documented with ***given names*** and the standard image-listing page in a SIMBA case review only lists the given named images. The SAM can access any image with a given name. It is located in a file folder with that module name. When a user reviews a case on the SIMBA web system all given named images are identified by thumbnail images. All SAMs associated with a project may be accessed by selection from a menu at the base of the thumbnail or by selecting from a button with the SAMs name.

1. SAPI Components

The following outlines the basic operation of the main SAPI components. In many case this behavior may be modified by using more advanced features and by options on the SAM configuration file.

(A) Action command

The primary component in a SAM is the **action** command that will perform the main computation or module action. This is a program or script that will process a selected input image. The action may be a script wrapper to execute a program or set of programs developed in any programming language that the server accepts. The SAPI has a single default action command named **compute** but it may be configured to a set of action commands.

The SAPI has the following requirements for action commands:

1. The *action* command must accept at least a single argument, which is the root (<root>) name for the image to be processed. The action command uses the <root> name to prefix all files that is accesses or creates.
2. All files associated with <action> should be accessed or created in the local directory in which <action> is executed (i.e., the action command should not create files with path names contacting directories).
3. The input image will be made accessible to the <action> by having a file name (or link) in the current directory to "<root>.if". Other files may be made available according to the SAPI configuration; all of these files will have the file prefix <root>.
4. All files created by <action> must have the name prefix <root> to make them unique. Further, they must have distinguishing extension, a unique subfile ID (SID). The module program developer must maintain the set of SIDs for a module.
5. The action command should allow two additional parameters that are useful for program debugging
    a. –v, this option specifies a verbose operation of the action command in which messages may be printed. The usual default mode for an action command is to not print any messages unless there is a fatal error.
    b. –d, this option specifies a debug mode in which additional messages may be printed and any temporary files that the action creates will not be automatically deleted.

The action command may be developed independent of the SIMBA system. When used with the interactive SIMBA environment it has been found to be convenient to have a follow-up command that creates any analysis or visualization images for review by a SIMBA user. For this event the SAPI supports an action analysis by execution of the program **"<action>.a**", if it exists, following any successful execution of <action>. The concept is that <action>.a will perform any necessary post-processing; it has the same arguments as the <action> command.

(B) Displaying a result image

The action command may create images that are listed in the SIMBA case review page. This is accomplished by creating a file with the name "<root>.<SID>.web". All files with the .web extension will be listed on the case review page. Such files should be in a format suitable for SIMBA image display; currently, supported Image types include: (a) VisionX Image files in either byte or short integer format, jpeg, and gif (not animated).

(C) Managing a status web-page

The web page associated with a SAPI module contains only the standard command buttons managed by SIMBA for all modules. Specific content relating to a module may be presented on this page as follows:

1. Any html content with the file name *<root>.<SID>.html* will be shown on the web page. These files must contain only body text without <body> tags and may not make links to any local files
2. Any text content with the file name *<root>.<SID>.dtxt* will be shown on the web page
3. Any images with the file name *<root>.<SID>.dim* will be shown on the web page. The image format must be web friendly as no format conversion takes place: the file types supported by most web browsers are `png`, `jpeg` and `gif(including animated gif)`.

(D) Presenting a set of results for a set of cases

Results that span a number of cases may be collected by a SIMBA gather operation or be external analysis. If a file in comma-separated-values (CSV) format is placed in the modules results directory it will be made available to the SIMBA results page view from where it may be viewed or down loaded. In addition, if a row contains a field called "description" that contains the case and image ID then callback links to the case and the module will be present when the results file is viewed.

2. SAPI Program Development

There are three modes for program use and development: local, batch and web based.

A. Local program development

The local mode allows a user to develop an action command on their local computer independent of the SIMBA system. For this reason action commands have no direct awareness or interaction with the SIMBA system. All that is needed is that test images be given a "`.if`" file name extension and that they be in the same directory that the action command is executed.

B. Interactive web operation

For interactive web operation the user must be logged onto the SIMBA project. The usual operation is to view the case on the case review page and to select the module (either by the module name or by a pull-down menu from the desired input image). Once the module web page is accessed it makes available an "actions" list in which all permissible actions are listed. Clicking on an action button will execute the action for that case. If execution of the action on a set of cases is required then the user can

click on the “next” button to move to the next case where the action can again be selected. This is not convenient for a large number of cases and the batch mode outlined below should be used.

C. Batch mode

The module contains a command “vsimba” that facilitates single and batch mode execution of actions from the back-end of the SIMBA system. Wen provided with an action and an image it creates the necessary directories for processing the image and executes the action in the appropriate directory. Vsimba may also be provided with a list of images in which case it will sequentially execute the action on each image in the list.

The Vsimba command for processing a single case has the syntax:

vsimba p=< project > c=<case>a i=<image> [-v] [-d] a=<action> [-delete]

For a list of images the command is

vsimba il=<image-list> [-v] [-d] a=<action>

where <image-list> is a csv file in which, after the initial row containing the project name, each row contains a case name and a given file name of the image to be processed.

The –v option species the verbose mode (which is passed on to the action script). In this mode that action may provide text output that will be recorded in the event log.

The –d options specifies a debug mode (which is passed on to the action script). In this mode the script may choose to not delete any temporary files providing more information for debugging purposes.

If the –delete option is specified then all files and directories associated with that case for that module will be deleted. Therefore, it is not a good idea to store manually created data in these directories.

3. SAM Configuration

A file called “exper.cfg” specifies the description and configuration of a module. The contents of this file are a set of lines with the syntax “<name>=<value>”. The only required line is the module description, in which case the value is a text string.

A module is associated with a SIMBA project by making a link to the module from the xbin/< project > directory to the module directory.

4. Multiple SAMs and module interactions

5. Event and Error Logging

Whenever *vsimba* is executed or a recordable event occurs in the SIMBA system an event record is made in the appropriate SIMBA log. If an action command run by SIMBA or vsimba generates any output to either standard out or standard error (i.e. it prints any messages such as if the –v option is set

or there is a detected error) then that text is recorded in the event log for that project. The event log file is located in ~reeves/simlog/XX where XX is the project name. Events are recorded one to a line in the event log in the following format:

<date-time>:<SAPI-name>:<case>:<image>:<action or event>

Text generated by the action will also be recorded in the event log. Each line in the event log reported from an action output will be identified by a prepended "=" character.

6. Local Program Development

Tools are provided to assist with program development outside the SIMBA web environment in a users local account. Normal *action* programs are standard vision programs or scripts that accept a single root argument. They can be developed in a local users directory by creating input files with the SAPI name structure and executing the action directly with the root name of the image file as an argument.

Development that requires applying an action to a set of cases (a requirement for gather and collect actions) is supported by the development command vxsimba. Vxsimba performs similar operations to the vsimba command except that it is used within the users local development directory rather than the SIMBA web environment.

Appendix A. SAM organization:

Consider a module with name XX. The module contains the following files:

File name	Function
exper.cfg	This is the configuration file for the module and species any special names or features associated with the module
page.sd	This is an optional script that generates the web page that is seen when the module is entered through the SIMBA web interface. In general a module developer will select a page from a number of templates; advanced users may construct more complex interfaces
option	This is the bridging library between the module and the interactive use from the SIMBA system. It is not usually used or modified directly by the API developer. In some cases it may not be needed and a system wide version of option may be used
vsimba il=<image-list> a=action [action-type]	This program manages the execution of the modules main functions. It can apply an action to a set or to all cases and it is used by option to correctly execute functions by SIMBA. The optional action-type specifier is needed only when executing non-default gather or collect actions (see later)
<action>	Actions are special programs or scripts that are designed to be executed by "vsimba" and do the actual data manipulation for a module. The may be run from SIMBA front end or in batch mode and may run in a stand-alone development environment. The following standard functions are defined
compute <root-name>	compute is a default <action> program or script that computes the main module function. It takes an image input file <root-name>.if. It may optionally create output files and/or request access to external resources such as SIMBA chest segmentation map or files from another module according to what is specified in the exper.cfg file.
<action>.a <root-name>	The optional <action>.a analysis action is executed following the execution of an <action>, if it exists. It takes comp output and performs any post-processing functions. For both comp and compute.a, image files ending with the extension .web will be maintained as links in a local *images* directory for interactive web access
<gather> <root-name> [-i] [-p]	Gather is an extended action type that gathers a set of results and place then into a standard location for review from SIMBA
<collect> <root-name> [-i][-p]	Collect is an extended action type that collect a set of images into a standard *images* directory for review by SIMBA

Appendix B. Standard File Locations

The standard location for SIMBA files are shown blow:

Key: the name of the SAPI module is "XX, the name of the SIMBA project is "YY", the name of a case is "ZZ".

The API module XX	/home/reeves/exper/XX
The image and other data for P=YY c=ZZ	/home/reeves/xbase/YY/ZZ/exper/XX/
The images to be displayed for P=YY c=ZZ	/home/reeves/xbase/YY/ZZ/exper/XX/images/
The data results from gather or an external results file	/home/reeves/dbase/results/XX/
The image results for collect	/home/reeves/xbase/YY/results/exper/XX/images
The event record	/home/reeves/simlog/YY

Appendix C. Action programs and scripts

Within a SAM the programs that can be invoked and executed by the framework are called actions and they can be selected with the a= option of vsimba:

vsimba … a=<action-name>

An action is run by the framework as follows:

<action-name> <root> [-v] [-d]

The first argument is always present and is the given name of the image to be processed that is used as the root name for all the files relating to that image, as explained below. The other flags are optional and appear only if they are requested when executing the vsimba command.

The SAPI has the following requirements for action commands:

1. The *action* command must accept at least a single argument, which is the given name (<root>) of the image to be processed. The action command uses the <root> name to prefix all files that it accesses or creates, with the exception of temporary files. (Notice that, if the script does not need this information, it can safely ignore it as long as the presence of an argument does not cause the script to halt or raise an error)
2. All files associated with <action> should be accessed or created in the local directory in which <action> is executed (i.e., the action command should not create files with path names contacting directories). There are two additional classes of scripts that generate a result file and a result image and whose outputs have special locations as detailed later on.
3. The input image will be made accessible to the <action> by having a file name (or link) in the current directory to "<root>.if". Other files may be made available according to the SAPI configuration file for the module; all of these files will have the file prefix <root>.

4. All files created by <action> must have the name prefix <root> to make them unique. Further, they must have distinguishing extension, a unique subfile ID (SID). The module program developer must maintain the set of SIDs for a module.
5. The action command should allow two additional parameters that are useful for program debugging
    a. –v, this option specifies a verbose operation of the action command in which messages may be printed. The usual default mode for an action command is to not print any messages unless there is a fatal error.
    b. –d, this option specifies a debug mode in which additional messages may be printed and any temporary files that the action creates will not be automatically deleted.

Other information related to the context in which the script has been called are accessed through environmental variables:

VSPROJECT	the two-letter codename of the project
VSCASE	the case ID of the current image
VSSAM	the name of the SAM being executed
VSSAMDIR	the directory where the current action and all of the other SAM files reside
VSREPORTDIR	the directory where non-image, project-wide results are stored
VSRESDIR	the directory where project-wide result images are stored
VSDSNAME	the name of the case list used to generate the results
VSFIRST	this variable is set to a non zero value when the first item in the case list is being evaluated
VSLAST	this variable is set to a non zero value when the last item in the case list is being evaluated

Example script

A simple template script is provided for reference

```
#!/bin/sh

# insert your custom code
#     use "$1" for the value of <root>

# remove all temporary files with a prefix tmp
rm -f tmp*
```