A Vendor-Neutral Library and Viewer for Whole-Slide Images

Adam Goode, M. Satyanarayanan

June 2008
CMU-CS-08-136

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Although widely touted as a replacement for glass slides and microscopes, whole-slide images used in digital pathology present a challenge in analysis and interoperability. No universal data format exists for these images: analysis tools, viewers, and libraries are vendor-specific. In this paper, we present a vendor-neutral C library for reading whole-slide image files. The library is high-performance, extensible, and easily interfaced to various programming languages. An application writer need only write a program against the library's API to read multiple vendor formats. A Java-based whole-slide viewer that uses the C library is also presented.

This research was supported by the Clinical and Translational Sciences Institute of the University of Pittsburgh (CTSI), with funding from the National Center for Research Resources (NCRR) under Grant No. 1 L1 RR024153. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the CTSI, NCRR, or Carnegie Mellon University.

Keywords: whole-slide images, virtual slides, digital pathology, microscopy, glass slides, Trestle, Zeiss, Aperio, Bacus, Hamamatsu, Olympus, Diamond

1 Introduction

Whole-slide images, also known as virtual slides, are large, high resolution images used in digital pathology. Reading these images using standard image tools or libraries is a challenge because these tools are typically designed for images that can comfortably be uncompressed into RAM or a swap file. Whole-slide images routinely exceed RAM sizes, often occupying tens of gigabytes when uncompressed. Additionally, whole-slide images are typically multi-resolution, and only a small amount of image data might be needed at a particular resolution.

There is no universal data format for whole-slide images, so each vendor implements its own formats, libraries, and viewers. Vendors typically do not document their formats. Even when there is documentation, important details are omitted. Because a vendor's library or viewer is the only way to view a particular whole-slide image, doctors and researchers can be unnecessarily tied to a particular vendor. Finally, few (if any) vendors provide libraries and viewers for non-Windows platforms. Some have gone with a server approach, pushing tiles through a web server, or using Java applets, but these approaches have shortcomings in high-latency or non-networked environments.

In this paper, we present a solution to this problem in the form of a vendor-neutral library for reading whole-slide images, as well as a simple but powerful viewer built on top of the library.

2 Design and Implementation

2.1 Whole-Slide Basics

Even though there are many variations in whole-slide formats, the problem addressed by each vendor is the same. Thus there are similarities in whole-slide formats across vendors. Here are some typical characteristics of whole-slide image formats:

- They are usually based on TIFF (though JPEG has been used).
- They can store downsampled versions of the image for quicker access to lower resolutions.
- They use various kinds of lossy compression.
- They are optimized for random access (though the quality of such optimization varies greatly).
- They are effectively unbounded in width and height.
- They can store slide metadata.

2.2 Design Goals

In designing the library, we had several goals in mind:

- Provide a single, vendor-neutral API.
- Work directly with image files in a standard filesystem.
- Provide read-only access.
- Bind easily to programming languages.
- Be extensible to new formats without changing the API.

- Offer a good user experience on desktop-class hardware.
- Allow for prefetching.

2.3 Whole-Slide Abstraction

The library provides a simple vendor-neutral abstraction to a variety of whole-slide image formats. To do this, it exposes an abstract interface that remains the same regardless of the underlying format. The abstraction:

- Provides a read-only interface to extract any rectangular region of a whole-slide image.
- Represents an image as an ordered list of layers, with Layer 0 being the maximum-resolution layer and each subsequent layer being a downsampled version of the previous layer.
- Hides all vendor-specifics, including tiles and metadata, except for a simple comment which can be queried (but is not otherwise interpreted).

No image scaling is performed by the library. There is no mechanism for extracting compressed data from an image, only uncompressed RGBA data is exposed. These constraints shield applications from the details of specific image formats, and make it possible to extend the library in the future without needing to change the existing API.

Additional calls exist to determine the downsampling factor for each layer (relative to Layer 0), and to determine the next largest layer for a arbitrary downsample factor. All image coordinates are specified with respect to the coordinate system of Layer 0.

2.4 Example

An example use case by a slide viewer is as follows:

1. Open a particular file with the library. This file contains 5 layers, with downsampling factors 1.0, 2.0, 4.0, 5.0, 10.0.
2. Query the library for the best layer to use for downsampling to 2.5. The library returns Layer 1.
3. Query the library for the actual downsampling of Layer 1. The library returns 2.0.
4. Divide 2.5 by 2.0 to get the additional downsampling factor required to display the slide. Answer is 1.25.
5. Extract image data from Layer 1, using the library. The library returns image data given at downsampling factor 2.0.
6. Rescale the image down by a factor of 1.25 to get the requested downsample factor of 2.5.
7. Display the rescaled image on the screen.
8. Close the file.

2.5 Vendor-Neutral Implementation

The library is implemented in C99 and uses GLib[8] extensively for memory allocation, data structure manipulation, and text file parsing. We rely upon LibTIFF[4], IJG's JPEG library[2], and JasPer[1] for image reading and processing of the various formats. Additional details about the API are in Appendix A.

Our design is structured similarly to the device driver model found in operating systems. Application-facing code is linked to vendor-specific code by way of internal constructors and function pointers. Roughly 90% of the code is in these device driver files, implementing specific functionality for reading vendor formats. The rest of the code is in the generic application-facing interface.

To support a new vendor, one need only implement a constructor, a `read_region` function, a `destroy` function, a `get_dimensions` function, and a `get_comment` function. Code already in the library for performing operations on JPEG and TIFF images can be leveraged by new vendor-specific code.

We currently support two file formats: Trestle and Hamamatsu. We have preliminary support for the Aperio format, and plan on adding Bacus format as well. These enhancements will be added in future releases of the library and will require no changes to applications built on top. Additional formats will be added as necessity and priorities dictate.

2.6 API Summary

Details are in Appendix A.

2.6.1 Basic usage

- **can_open**

  Do a quick check to see if a whole slide image is valid.

- **open**

  Open a whole slide image.

- **get_region_num_bytes**

  Compute minimum buffer size for given image region.

- **read_region**

  Copy ARGB data from a whole slide image.

- **close**

  Close a whole slide image handle.

2.6.2 Image and layer navigation

- **get_layer_count**

  Get the number of layers in the whole slide image.

- **get_layer0_dimensions**

  Get the dimensions of layer 0 (the largest layer).

- **get_layer_dimensions**

  Get the dimensions of a layer.

- **get_layer_downsample**

  Get the downsampling factor of a given layer.

- **get_best_layer_for_downsample**

  Get the best layer to use for displaying the given downsample.

2.6.3 Performance optimizations

- **give_prefetch_hint**

  Give a non-blocking hint that a region is likely to be needed soon.

- **cancel_prefetch_hint**

  Cancel an existing prefetch hint.

2.6.4 Metadata access

- **get_comment**

  Get the comment (if any) for this image.

3 Applications

3.1 Java Viewer

For our first application, we built a simple viewer in Java for viewing one or two whole-slide images simultaneously. The user can zoom in and out, pan around, and jump back to the center. Both keyboard and mouse controls are available. A tablet-based interface is also provided. In dual-slide mode (shown in Figure 1), the user can choose to pan and zoom each slide separately or link both together for simultaneous navigation. Incidentally, this application demonstrates the use of our C library in a Java application.

3.2 PathFind

PathFind (Figure 2) is a Diamond[7] application for pathologists. It incorporates the whole-slide library and allows the pathologist to zoom and navigate just as is done with a microscope and glass slides today. The PathFind interface allows the pathologist to identify regions of interest on the slide at any level of magnification and then search for similar regions via Diamond across multiple slide formats. The results returned by Diamond can be viewed and compared with the original image, and the case data for each result can also be retrieved.

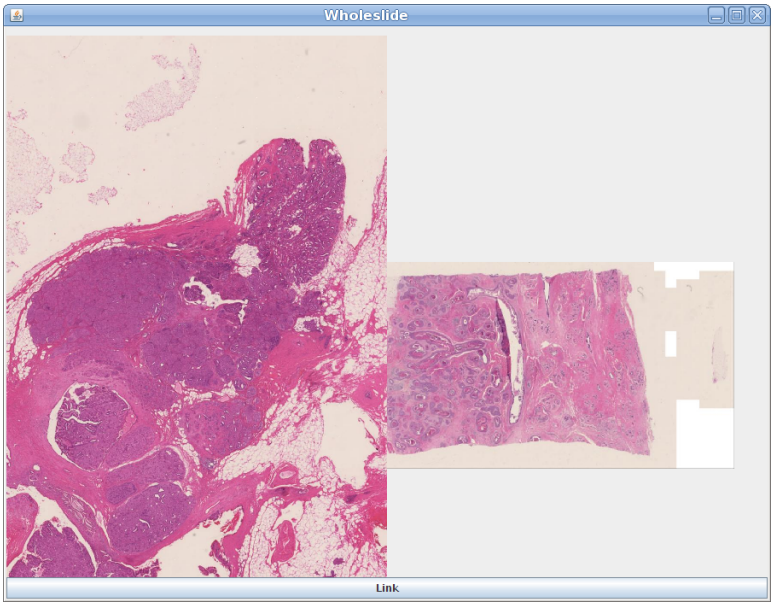

Figure 1: Java-based Whole-Slide Viewer

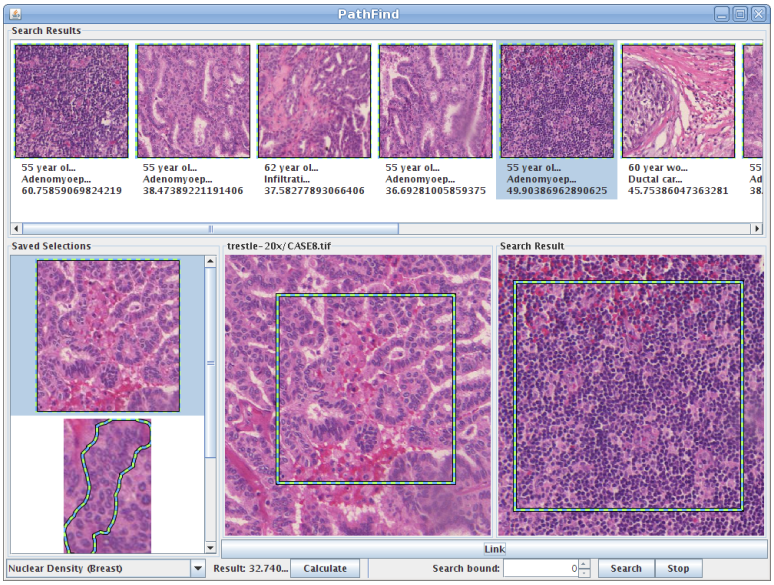

Figure 2: PathFind

Analyses can be performed using ImageJ[6] macros. We currently support macros written by our users, as well as macros written by other users of ImageJ. One ImageJ plugin we use is Colour Deconvolution[3]. We plan on extending this plugin to support automatic calibration of stains using machine learning, to improve the quality of results. We are also planning to extend PathFind to train user-specific classifiers as part of normal workflow. We are exploring the use of GENIE[5] for generating classifiers in PathFind. We expect that a cascade of ImageJ macros and user-trained

classifiers will be common in PathFind. We envision controlled sharing of user-specific classifiers over the Internet.

4 Future Work

Future work on the library includes: supporting more whole-slide image formats, better error reporting, better performance, and support for Windows platforms. Future work on the Java-based viewer includes: more keyboard navigation shortcuts, rotation of slides, increased performance, and better zooming and scrolling display. Future work on PathFind includes: server-side searching of whole-slide images, auto calibration for color deconvolution, reduction of false positives in searches, ability to do range-based searches, image rotation, more keyboard shortcuts (including prev/next ROI, prev/next result, and prev/next result from current case), and support for multiple chained filters in the interface.

5 Acknowledgements

We wish to thank Drazen Jukic, Jonhan Ho and Laura Drogowski for their critical review and feedback of the software described in this document. This research was supported by the Clinical and Translational Sciences Institute of the University of Pittsburgh (CTSI), with funding from the National Center for Research Resources (NCRR) under Grant No. 1 L1 RR024153. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the CTSI, NCRR, or Carnegie Mellon University.

References

[1] ADAMS, M. D. JasPer. http://www.ece.uvic.ca/ mdadams/jasper/.

[2] INDEPENDENT JPEG GROUP. The Independent JPEG Group's JPEG software. http://www.ijg.org/.

[3] LANDINI, G. Colour deconvolution. http://www.dentistry.bham.ac.uk/landinig/software/cdeconv/cdeconv.html.

[4] LEFFLER, S. LibTIFF. http://www.remotesensing.org/libtiff/.

[5] PERKINS, S. J., THEILER, J. P., BRUMBY, S. P., HARVEY, N. R., PORTER, R. B., SZYMANSKI, J. J., AND BLOCH, J. J. Genie: a hybrid genetic algorithm for feature classification in multispectral images. B. Bosacchi, D. B. Fogel, and J. C. Bezdek, Eds., vol. 4120, SPIE, pp. 52–62.

[6] RASBAND, W. ImageJ. http://rsb.info.nih.gov/ij/, 1997-2007. U. S. National Institutes of Health, Bethesda, Maryland, USA.

[7] SATYANARAYANAN, M., SUKTHANKAR, R., GOODE, A., HUSTON, L., MUMMERT, L., WOLBACH, A., HARKES, J., GASS, R., AND SCHLOSSER, S. The opendiamond platform for discard-based search. Tech. Rep. CMU-CS-08-132, School of Computer Science, Carnegie Mellon University, May 2008.

[8] THE GTK+ PROJECT. GLib. http://www.gtk.org/.

A Whole-Slide Library API

A.1 Function Documentation

A.1.1 wholeslide_public bool ws_can_open (const char * *filename*)

Do a quick check to see if a whole-slide image is valid.

Parameters: ***filename*** The filename to check.

Returns: If **ws_open()** (p. 9) will succeed.

A.1.2 wholeslide_public void ws_cancel_prefetch_hint (wholeslide_t * *wsd*, uint32_t *prefetch_id*)

Cancel an existing prefetch hint.

Parameters: ***wsd*** The whole-slide image handle.

prefetch_id An identifier returned by **ws_give_prefetch_hint()** (p. 9).

A.1.3 wholeslide_public void ws_close (wholeslide_t * *wsd*)

Close a whole-slide image handle.

Parameters: ***wsd*** The whole-slide image handle.

A.1.4 wholeslide_public uint32_t ws_get_best_layer_for_downsample (wholeslide_t * *wsd*, double *downsample*)

Get the best layer to use for displaying the given downsample.

Parameters: ***wsd*** The whole-slide image handle.

downsample The downsample factor.

Returns: The layer identifier.

A.1.5 wholeslide_public const char* ws_get_comment (wholeslide_t * *wsd*)

Get the comment (if any) for this image.

Parameters: ***wsd*** The whole-slide image handle.

Returns: The comment for this image.

A.1.6 wholeslide_public void ws_get_layer0_dimensions (wholeslide_t * *wsd*, uint32_t * *w*, uint32_t * *h*)

Get the dimensions of layer 0 (the largest layer).

Parameters: ***wsd*** The whole-slide image handle.

→ ***w*** The width of the image.

→ ***h*** The height of the image.

A.1.7 wholeslide_public uint32_t ws_get_layer_count (wholeslide_t * *wsd*)

Get the number of layers in the whole-slide image.

Parameters: ***wsd*** The whole-slide image handle.

Returns: The number of layers.

A.1.8 wholeslide_public void ws_get_layer_dimensions (wholeslide_t * *wsd*, uint32_t *layer*, uint32_t * *w*, uint32_t * *h*)

Get the dimensions of a layer.

Parameters: ***wsd*** The whole-slide image handle.

layer The desired layer.

→ ***w*** The width of the image.

→ ***h*** The height of the image.

A.1.9 wholeslide_public double ws_get_layer_downsample (wholeslide_t * *wsd*, uint32_t *layer*)

Get the downsampling factor of a given layer.

Parameters: ***wsd*** The whole slide image handle.

layer The desired layer.

Returns: The downsampling factor for this layer.

A.1.10 wholeslide_public size_t ws_get_region_num_bytes (wholeslide_t * *wsd*, uint32_t *w*, uint32_t *h*)

Compute minimum buffer size for given image region.

Parameters: ***wsd*** The whole slide image handle.

w The width of the region.

h The height of the region.

Returns: The minimum number of bytes needed to hold the uncompressed image data for the region.

A.1.11 wholeslide_public uint32_t ws_give_prefetch_hint (wholeslide_t * *wsd*, uint32_t *x*, uint32_t *y*, uint32_t *layer*, uint32_t *w*, uint32_t *h*)

Give a non-blocking hint that a region is likely to be needed soon.

Parameters: ***wsd*** The whole slide image handle.

x The top left x-coordinate.

y The top left y-coordinate.

layer The desired layer.

w The width of the region.

h The height of the region.

Returns: A unique identifier for this prefetch hint.

A.1.12 wholeslide_public wholeslide_t* ws_open (const char * *filename*)

Open a whole slide image.

Parameters: ***filename*** The filename to open.

Returns: A new handle to an open whole slide image.

A.1.13 wholeslide_public void ws_read_region (wholeslide_t * *wsd*, uint32_t * *dest*, uint32_t *x*, uint32_t *y*, uint32_t *layer*, uint32_t *w*, uint32_t *h*)

Copy ARGB data from a whole slide image.

This function reads and decompresses a region of a whole slide image into the specified memory location. `dest` must be a valid pointer to enough memory to hold the region. To compute the proper size for `dest`, use **ws_get_region_num_bytes()** (p. 8).

Parameters: ***wsd*** The whole slide image handle.

dest The destination buffer for the ARGB data.

x The top left x-coordinate.

y The top left y-coordinate.

layer The desired layer.

w The width of the region.

h The height of the region.