Dicom_wsi: A Python Implementation for Converting Whole-Slide Images to Digital Imaging and Communications in Medicine Compliant Files

Qiangqiang Gu[1], Naresh Prodduturi[1], Jun Jiang[1], Thomas J. Flotte[2], Steven N. Hart[1]

[1]Department of Health Sciences Research, Division of Biomedical Statistics and Informatics, Mayo College of Medicine, Rochester, Minnesota, USA,
[2]Department of Laboratory Medicine and Pathology, Mayo College of Medicine, Rochester, Minnesota, USA

Submitted: 07-Oct-2020 Revised: 19-Jan-2021 Accepted: 27-Feb-2021 Published: 11-May-2021

Abstract

Background: Adoption of the Digital Imaging and Communications in Medicine (DICOM) standard for whole slide images (WSIs) has been slow, despite significant time and effort by standards curators. One reason for the lack of adoption is that there are few tools which exist that can meet the requirements of WSIs, given an evolving ecosystem of best practices for implementation. Eventually, vendors will conform to the specification to ensure enterprise interoperability, but what about archived slides? Millions of slides have been scanned in various proprietary formats, many with examples of rare histologies. Our hypothesis is that if users and developers had access to easy to use tools for migrating proprietary formats to the open DICOM standard, then more tools would be developed as DICOM first implementations. **Methods:** The technology we present here is dicom_wsi, a Python based toolkit for converting any slide capable of being read by the OpenSlide library into DICOM conformant and validated implementations. Moreover, additional postprocessing such as background removal, digital transformations (e.g., ink removal), and annotation storage are also described. dicom_wsi is a free and open source implementation that anyone can use or modify to meet their specific purposes. **Results:** We compare the output of dicom_wsi to two other existing implementations of WSI to DICOM converters and also validate the images using DICOM capable image viewers. **Conclusion:** dicom_wsi represents the first step in a long process of DICOM adoption for WSI. It is the first open source implementation released in the developer friendly Python programming language and can be freely downloaded at https:// github.com/Steven N Hart/dicom_wsi.

Keywords: Digital Imaging and Communications in Medicine, informatics, infrastructure, whole-slide imaging

Introduction

Digital Imaging and Communications in Medicine (DICOM) has been the standard file and communication format for medical imaging data beginning in the 1990s.[1] While it has had a dramatic influence on interoperable systems in radiology, several challenges limited its deployment into digital pathology. First, whole-slide images (WSIs) are typically much larger than other medical images. Common image sizes for digital fluoroscopy, interventional radiology, and computed tomography are typically between 512 × 512 × 8 (2.1 megapixels) and 1024 × 1024 × 12 (12.5 megapixels),[2] whereas WSIs are routinely 80000 × 60000 × 3 (14400 megapixels). Second, access patterns are different in pathology than radiology. Given the large size, it is infeasible for pathologists to pan and zoom across a full resolution image, so different levels of resolution are necessary. However, a specification exists for storing DICOM in WSI[3] which has accounted for the unique requirements for digital pathology, and should therefore be the ideal storage format for WSIs, since most existing institutional infrastructure can support DICOM-based communication protocols.

Address for correspondence: Dr. Steven N. Hart, Department of Health Sciences Research, Division of Biomedical Statistics and Informatics, Mayo College of Medicine, Rochester, Minnesota, USA. E-mail: hart.steven@mayo.edu

Access this article online

Quick Response Code:

Website: www.jpathinformatics.org

DOI: 10.4103/jpi.jpi_88_20

How to cite this article: Gu Q, Prodduturi N, Jiang J, Flotte TJ, Hart SN. Dicom_wsi: A python implementation for converting whole-slide images to digital imaging and Communications in Medicine compliant files. J Pathol Inform 2021;12:21.

Available FREE in open access from: http://www.jpathinformatics.org/text.asp?2021/12/1/21/315772

In 2017 and 2018 at the Pathology Visions Conference, connectathons were hosted by the Digital Pathology Association for vendors of slide scanning instruments to prove their DICOM compatibility.[4] Findings from such events will undoubtedly lead to rectifying any deficiencies from a technology perspective, yet no existing platform offers DICOM WSI files as the default recommended output, instead opting for (usually) proprietary image formats that are difficult to access. The Aperio GT450 (Leica Biosystems) is the first commercially available scanner that can provide native DICOM images, however, due to the availability of viewers and integration into preexisting systems, the images are most frequently converted to .svs files in their image server.

Exporting images natively to DICOM format is only part of the solution. Millions of WSIs have already been scanned in vendor-native formats. File converters are necessary to make these slides accessible to DICOM in future, but few have been made publicly available. Herrmann *et al*.[5] describe a python-based implementation, but as of yet have not released code. Clunie,[6] went a step forward to describe not only how to encode digital pathology images into but also described how to incorporate a dual personality into the image via a Java implementation-making the DICOM file both valid and invalid at the same time.[7] However, this was intended to be a proof of concept, rather than a full-fledged implementation of a conversion tool. Orthanc[8] and Google[9] have released their own converters in C++ based on the OpenSlide library[10] (*Dicomizer* and *wsi2dcm*, respectively).

Programming languages such as C++ and Java are languages that are compiled into machine instructions before execution. These languages are generally faster, but are more difficult to learn. Interpreted languages such as Python are generally easier to learn since an interpreter provides a layer of abstraction away from the machine code and into more human readable implementations. To make code more approachable to a larger group of programmers, we have chosen Python as the desired implementation for a DICOM converter.

In this article, we present *dicom_wsi*, a python toolkit to convert WSI to DICOM. As all customizable parameters are set in a YAML file, our modular framework makes WSI to DICOM conversion highly configurable [Figure 1]. To save archive space, sparse tiling was implemented to decrease file sizes. To provide auxiliary support for downstream analysis, annotations from other tools (such as QuPath[11]) can be saved into DICOM files. In all, many advanced features were added to this toolkit, making our toolkit compatible to many existing packages and approachable to many application scenarios.

Methods

Generation of Digital Imaging and Communications in Medicine files

DICOM files are generated using the pydicom[12] library wherever possible. Much, but not all, of the data needed for conversion of proprietary slide formats to DICOM can be automatically extracted (or are constants) from proprietary WSIs. To account for this, *dicom_wsi* needs only a few key-value pairs from a YAML file, which are broken up into the following sections: "General," "BaseAttributes," and "Constants."

The "General" section provides options that are not ultimately added into the DICOM output files, but rather tell *dicom_wsi* how to create the files. This includes parameters like the slide name, what the output name should be, how

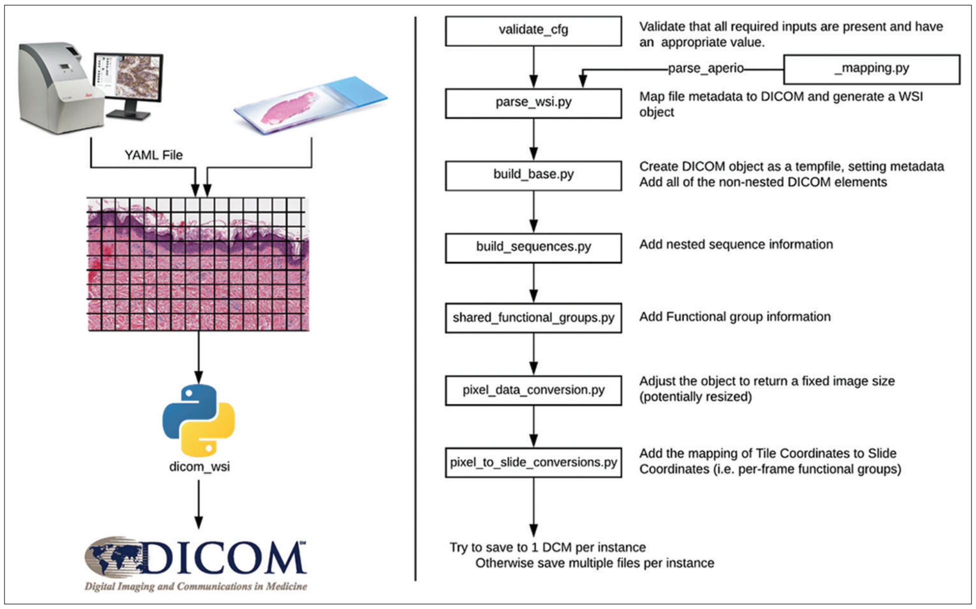
YAML File
dicom_wsi
DICOM
Digital Imaging and Communications in Medicine
validate_cfg
Validate that all required inputs are present and have an appropriate value.
parse_aperio
_mapping.py
parse_wsi.py
Map file metadata to DICOM and generate a WSI object
build_base.py
Create DICOM object as a tempfile, setting metadata
Add all of the non-nested DICOM elements
build_sequences.py
Add nested sequence information
shared_functional_groups.py
Add Functional group information
pixel_data_conversion.py
Adjust the object to return a fixed image size (potentially resized)
pixel_to_slide_conversions.py
Add the mapping of Tile Coordinates to Slide Coordinates (i.e. per-frame functional groups)
Try to save to 1 DCM per instance
Otherwise save multiple files per instance

Figure 1: Overview of Digital Imaging and Communications in Medicine conversion. The left shows the required inputs (YAML file and whole slide image) and the Digital Imaging and Communications in Medicine output. The right panel steps through each of the steps that the code performs with summaries of the tasks to be completed by that code base

many levels to extract, etc., The "ImageFormat" parameter tells *dicom_wsi* what compression algorithm to use (None, ".jpg," or ".jp2" [JPEG2000]), and a "CompressionAmount" parameter controls the level of compression for JPEG images. Other parameters such as how many tiles should be included per DICOM file and how large those tiles should be are also found in this section. Finally, if the image tiles should be compressed before storing in DICOM, then those parameters are also set here.

The "BaseAttributes" section contains the nonnested (i.e. Sequence) key-value pairings of DICOM elements, which enables dicom_wsi to recognize what are the required metadata needs to be stored alongside with the WSIs. Typical features in this section include the patient name, birthdate, when the study/series were taken, and whether the image should be tiled sparsely or fully (see below).

Sparse tiling

To decrease the amount of file space used, the DICOM standard allows for image tiles to be sparse or full. By setting "BaseAttributes.DimensionOrganizationType" to TILED_FULL, the output DICOM file will contain each image tile present in the source image. However, this can contain large amounts of whitespace, slide edges, and artifacts. Therefore, *dicom_wsi* supports the TILED_SPARSE option as well. When selected, each tile is first converted into a grayscale image and the percentage of pixels ("General.threshold," default 50%) that exceed some value ("General.background_range," default = 80). If the criteria are not satisfied, then that tile is not stored in the final DICOM file.

Annotations

Another important feature of *dicom_wsi* is the ability to store digital annotations. Annotations fall into one of four classes: Point, Rectangles, and Areas. Points can be used to mark the location of mitoses, rectangles are more suited for defining regions of interest, and areas can be for highly variable regions like those used for tumor segmentation. According to the DICOM standard, annotations are saved in "GraphicAnnotationSequence." For each frame, annotations could be saved as "TextObjectSequence" or "GraphicObjectSequence" under "ReferencedImageSequence" depend on the geometry type of the annotation. If annotations are available for the WSI, make sure to define the "Annotations" attribute in the "General" section of the YAML file.

Validation of DICOM Files was performed using *dciodvfy* from the *dicom3tools* package v1.0.[13] Importantly, *dciodvfy* does not verify the integrity of the images, only the metadata and text fields, nor does it currently support the TILED_SPARSE option, so errors in validation are not completely solved by this validation tool.

To read proprietary image files, *dicom_wsi* leverages the OpenSlide API and toolkit.[10] As OpenSlide natively supports multiple vended solutions (Aperio [.svs, .tif], Hamamatsu [.vms, .vmu, .ndpi], Lecia [.scn], MIRAX [.mrxs], Philips [.tiff], Sakura [.svslide], Trestle [.tif], Ventana [.bif,.tif], Generic tiled TIFF [.tif]), it was a logical choice for reading image files. However, to ensure that *dicom_wsi* was capable of fully converting these file types, we tested DICOM conversion of these 8 formats available from the OpenSlide website.

More detailed instructions for running *dicom_wsi* can be found at https://dicom-wsi.readthedocs.io/en/latest/index.html.

Results

Conversion of reference images

CMU-1-JP2K-33005.svs was downloaded from the OpenSlide website and converted into DICOM with *dicom_wsi, wsi2dcm,* and *Dicomizer*. All comparisons were made inside the Docker container provided in the code and run on a MacBook Pro with 16GB memory and 2.2 GHz Intel Core i7 microprocessor [Table 1]. The Orthanc *Dicomizer* was the fastest algorithm, taking only 2 min to complete the conversion using the TILED_FULL strategy. The TILED_SPARSE feature is not supported in *Dicomizer*. *dicom_wsi*, was the slowest algorithm-taking twice as long as *wsi2dcm*, however, it also yielded the smallest file size (130MB for TILED_SPARSE) which was almost identical to the size of the source file (127MB).

Validation of Digital Imaging and Communications in Medicine files

All examples emitted warnings for the Multi-frame Functional Groups Module, specifically the Plane Position (Slide) Sequence Attributes (Tag 0048,021A) nested within the Shared Functional Groups Sequence Attribute (Tag 5200,9229). DICOM files containing the TILED_SPARSE encoding yielded the errors "Number of items in Per-frame Functional Groups Sequence does not match Number of Frames" and "Number of values of DimensionIndexValues does not match number of items in DimensionIndexSequence." We interpret both of these reported errors not as a limitation to the existing implementation, but to the validator itself, since the files were still able to be processed through command line tools leveraging pydicom and image data.[7]

Extracting Annotations from Digital Imaging and Communications in Medicine

Annotations were extracted with a python script ("*extract_*

Table 1: Metrics for processing the CMU-1-JP2K-33005.svs example file

	Real	Size (MB)	Files#
CMU-1-JP2K-33005.svs	-	127	1
dicom_wsi (TILED_SPARSE)	9m25.914s	130	9
wsi2dcm (TILED_SPARSE)	4m5.475s	157	21
wsi2dcm (TILED_FULL)	4m5.875s	158	21
dicom_wsi (TILED_FULL)	11m245.203s	198	20
Dicomizer (TILED_FULL)	2m22.826s	223	26

annotations.py") available in the main *dicom_wsi* code base. As most individuals will not immediately have annotations from which to work with, we provide some example annotations for the Aperio example CMU-1-JP2K-33005.svs file available on the OpenSlide website. We used QuPath[11] to draw the annotations, which have no physiological relevance, only to show how the different data types can be stored inside DICOM. Results are extracted in a standard JavaScript Object Notation string.

Extracting Tiles from Digital Imaging and Communications in Medicine

DICOM files that were TILED_FULL were able to be displayed in the Orthanc DICOM Viewer with the WSI Library.[14] DICOM files that were TILED_SPARSE were not supported by the viewer. Instead, we also provide a python script to extract tiles from DICOM files (*extract_image_patches.py*). Extracted image tiles were able to be viewed with nonspecific image viewers found on Windows and Mac computers.

Discussion

In this article, we have presented *dicom_wsi* as a method for converting WSI into DICOM compliant files. Implementing the DICOM standard for digital pathology has been well defined, but poorly implemented. Our goal is to make the transition easier, particularly when institutions are planning to replace physical slides with digital ones. Rather than constantly reinventing new concepts for storage, WSI archiving should conform to DICOM standards to ensure interoperability with other medical systems and infrastructures built over the last few decades.

There are still a number of limitations to the ideal state of *dicom_wsi*. First, compared with other existing toolkit to convert WSIs into dicom standard format, our *dicom_wsi* method requires more time convert WSIs into dicom by either passing full image tiles stored in dicom format or only storing the image tiles with the none-interesting tiles been parsed in advance. Future improvements to the code, in particular the steps writing and compressing JPEG, would likely improve the performance of this task.

Second, even though *dicom_wsi* toolkit can convert 8 different formats, this does not account for all possible formats. A notable exception is the Philips iSyntax format.[15] As this format is not supported by OpenSlide, it must first be converted into an appropriate alternative. The *dicom_wsi* toolkit provides an example for how to convert the iSyntax to a TIFF file, which can then be converted as usual. Hopefully, as vendors continue to innovate, they consider interoperability as a prerequisite and therefore directly output DICOM by default or at least into a format that is accessible to the community at large.

Third, visualization tools that support DICOM like Orthanc[14] do not take advantage of LOCALIZER for navigation or TILED_SPARSE for display. Without using the LOCALIZER attribute, each layer of the WSI is loaded into memory in bulk, rather than just the levels and resolutions needed. This makes some high resolution levels inaccessible in the viewer, as these image sizes can easily overwhelm modern workstations. DICOM files that were TILED_SPARSE could not be rendered by Orthanc because it does not explicitly know how to "fill in" the missing tiles.

Despite limitations and lack of mature tooling in the field, *dicom_wsi* represents an incremental step forward for making DICOM the standard format for WSIs. Currently it allows users to transform any archived WSIs saved in an OpenSlide-supported format into standard DICOM format. It also provides users with the option of saving complete or sparsely tiled images. Finally, *dicom_wsi* represents the first tool available to allow digital annotations of WSIs to be stored in the same DICOM files as the corresponding images-opening up exciting possibilities for collecting valuable manual annotations in future.

Acknowledgments

This work was funded by the Center for Individualized Medicine (SNH), The Leon Lowenstein Foundation (SNH), and the Kern Center for the Science of Healthcare delivery (SNH and TJF).

Financial support and sponsorship

Nil.

Conflicts of interest

There are no conflicts of interest.

References

1. DICOM Standard. Available from: http://dicom.nema.org. [Last accessed on 13 Feb 20].
2. Archiving, Chapter 2: Medical Image Data Characteristics-Society for Imaging Informatics in Medicine. In: SIIM.org. Available from: https://siim.org/page/archiving_chapter 2. [Last accessed on 13 Feb 20].
3. DICOM Whole Slide Imaging. Available from: Available from: http://dicom.nema.org/Dicom/DICOMWSI/. [Last accessed on 13 Feb 20].
4. Clunie D, Hosseinzadeh D, Wintell M, De Mena D, Lajara N, Garcia-Rojo M, *et al*. Digital Imaging and Communications in Medicine whole slide imaging connectathon at digital pathology association pathology visions 2017. J Pathol Inform 2018;9:6.
5. Herrmann MD, Clunie DA, Fedorov A, Doyle SW, Pieper S, Klepeis V, *et al*. Implementing the DICOM standard for digital pathology. J Pathol Inform 2018;9:37.
6. Clunie DA. Dual-personality DICOM-TIFF for whole slide images: A migration technique for legacy software. J Pathol Inform 2019;10:12.
7. Generated Documentation (Untitled). Available from: http://www.dclunie.com/pixelmed/software/javadoc/index.html?com/pixelmed/convert/TIFFToDicom.html. [Last accessed on 17 Feb 20].
8. Jodogne S. The orthanc ecosystem for medical imaging. J Digit Imaging 2018;31:341-52.
9. GoogleCloudPlatform. GoogleCloudPlatform/wsi-to-dicom-converter. In: GitHub. Available from: https://github.com/GoogleCloudPlatform/wsi-to-dicom-converter. [Last accessed on 21 Jan 20].
10. Goode A, Gilbert B, Harkes J, Jukic D, Satyanarayanan M. OpenSlide: A vendor-neutral software foundation for digital pathology. J Pathol Inform 2013;4:27.
11. Bankhead P, Loughrey MB, Fernández JA, Dombrowski Y, McArt DG, Dunne PD, *et al*. QuPath: Open source software for digital pathology

image analysis. Sci Rep 2017;7:16878.

12. Mason D, Scaramallion, Rhaxton, Mrbean-Bremen, Suever J, Vanessasaurus, *et al*. Pydicom/Pydicom: 1.3.0; 2019. Available from: https://zenodo.org/record/3333768#.YFVPHbRKjBI. [Last accessed on 2020 Jul 22]. [doi: 10.5281/zenodo.3333768].
13. DICOM Validator-Dciodvfy. Available from: https://www.dclunie.com/dicom3tools/dciodvfy.html. [Last accessed on 22 Jul 20].
14. Jodogne S, Bernard C, Devillers M, Lenaerts E, Coucke P. Orthanc – A Lightweight, Restful DICOM Server for Healthcare and Medical Research. 2013 IEEE 10th International Symposium on Biomedical Imaging; 2013. [doi: 10.1109/isbi. 2013.6556444].
15. Philips. Philips' iSyntax for Digital Pathology. In: The Pathologist 24 Nov 2016. Available from: https://thepathologist.com/app-notes/philips-isyntax-for-digital-pathology. [Last accessed on 23 Jul 20].