DicomAnnotator: A Configurable Open-Source Software Program for Efficient DICOM Image Annotation

Qifei Dong
Department of Biomedical Informatics and Medical Education, University of Washington
Seattle, WA 98195, USA

Gang Luo
Department of Biomedical Informatics and Medical Education, University of Washington
Seattle, WA 98195, USA

David Haynor
Department of Radiology, University of Washington,
Seattle, WA 98195-7115, USA

Michael O'Reilly
Department of Radiology, University of Washington
Seattle, WA 98195-7115, USA

Ken Linnau
Department of Radiology, University of Washington
Seattle, WA 98195-7115, USA

Ziv Yaniv
Medical Science & Computing, LLC.,
Rockville, MD 20852, USA
and
National Institute of Allergy and Infectious Diseases, National Institutes of Health,
Bethesda, MD 20814, USA

Jeffrey G. Jarvik
Departments of Radiology, Neurological Surgery and Health Services, University of Washington,
Seattle, WA 98104-2499, USA

Nathan Cross
Department of Radiology, University of Washington
Seattle, WA 98195-7115, USA
Email: nmcross@uw.edu
Telephone: (850) 339-2438
Fax: (206) 598-8475

Acknowledgements

Research reported in this publication was supported by the University of Washington CLEAR Center for Musculoskeletal Research. The CLEAR Center is supported by the National Institute of Arthritis and Musculoskeletal and Skin Diseases (NIAMS) of the National Institutes of Health under Award Number P30AR072572. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Ziv Yaniv's work was supported by the Intramural Research Program of the U.S. National Institutes of Health, National Library of Medicine.

Conflict of Interest

Qifei Dong reports grants from NIH/NIAMS, during the conduct of the study.

Dr. Luo reports grants from NIH/NIAMS, during the conduct of the study.

Dr. Haynor reports grants from NIH/NIAMS, during the conduct of the study.

Dr. Linnau reports grants from Siemens Healthineers, personal fees from Siemens Healthineers, other from Cambridge Press, outside the submitted work.

Dr. Jarvik reports grants from NIH/NIAMS, during the conduct of the study; and Springer Publishing: Royalties as a book co-editor; GE-Association of University Radiologists Radiology Research Academic Fellowship (GERRAF): Travel reimbursement for Faculty Board of Review; Wolters Kluwer/UpToDate: Royalties as a chapter author.

Dr. Cross reports grants from NIH/NIAMS, during the conduct of the study; personal fees from Philips Medical, other from GE Medical, outside the submitted work.

All other authors report no conflict of interest.

Abstract

Modern, supervised machine learning approaches to medical image classification, image segmentation, and object detection usually require many annotated images. As manual annotation is usually labor-intensive and time-consuming, a well-designed software program can aid and expedite the annotation process. Ideally, this program should be configurable for various annotation tasks, enable efficient placement of several types of annotations on an image or a region of an image, attribute annotations to individual annotators, and be able to display Digital Imaging and Communications in Medicine (DICOM)-formatted images. No current open-source software program fulfills these requirements. To fill this gap, we developed DicomAnnotator, a configurable open-source software program for DICOM image annotation. This program fulfills the above requirements and provides user-friendly features to aid the annotation process. In this paper, we present the design and implementation of DicomAnnotator. Using spine image annotation as a test case, our evaluation showed that annotators with various backgrounds can use DicomAnnotator to annotate DICOM images efficiently. DicomAnnotator is freely available at https://github.com/UW-CLEAR-Center/DICOM-Annotator under the GPLv3 license.

Keywords

Image annotation, DICOM, open source, software design, machine learning

1. Background

Modern, supervised machine learning approaches to image classification, segmentation, and object detection typically require many annotated images. Sa *et al.* [1] used 974 annotated X-ray images to build a model for intervertebral disc detection. Esteva *et al*. [2] used 129,450 annotated clinical images to construct a model for classifying skin cancers. Performing manual image annotation is labor-intensive, tedious, and time-consuming, but often necessary for generating a high-quality dataset. Annotators frequently need to place several types of annotations in multiple areas of an image. In tasks with ambiguity or subjectivity, multiple experts whose time is expensive need to participate in the annotation task to reach consensus, multiplying the effort required for each dataset.

A well-designed software program can expedite image annotation. Ideally, the annotation program should meet the following requirements:

1) Customizable configurations should be available to support diverse annotation tasks. Tasks could require using differing types of labels, such as one label for the whole image vs. several other labels for the regions of interest in the image. Also, differing shapes might be needed to segment the regions of interest in the image. For instance, four corner points can outline a vertebral body on a spine image, whereas, a more complex shape would be needed to segment a lobulated or irregular mass.
2) The program should allow efficient placement of several types of annotations on one image, such as polygons to outline the regions of interest (termed *bounding polygons*), labels for the regions of interest, and a label for the whole image.
3) Several labels could be needed for each bounding polygon or region of interest. Consider an annotation task dealing with many regions of interest, each with its own identifier (see Fig. 1). For each region, some programs require the annotator to put a bounding polygon on the region, select its identifier from a long list, and then input the labels [3, 4]. Instead of having this inefficient workflow, an annotation program should support an optimized annotation approach, where multiple labels can be efficiently applied to a region of interest.

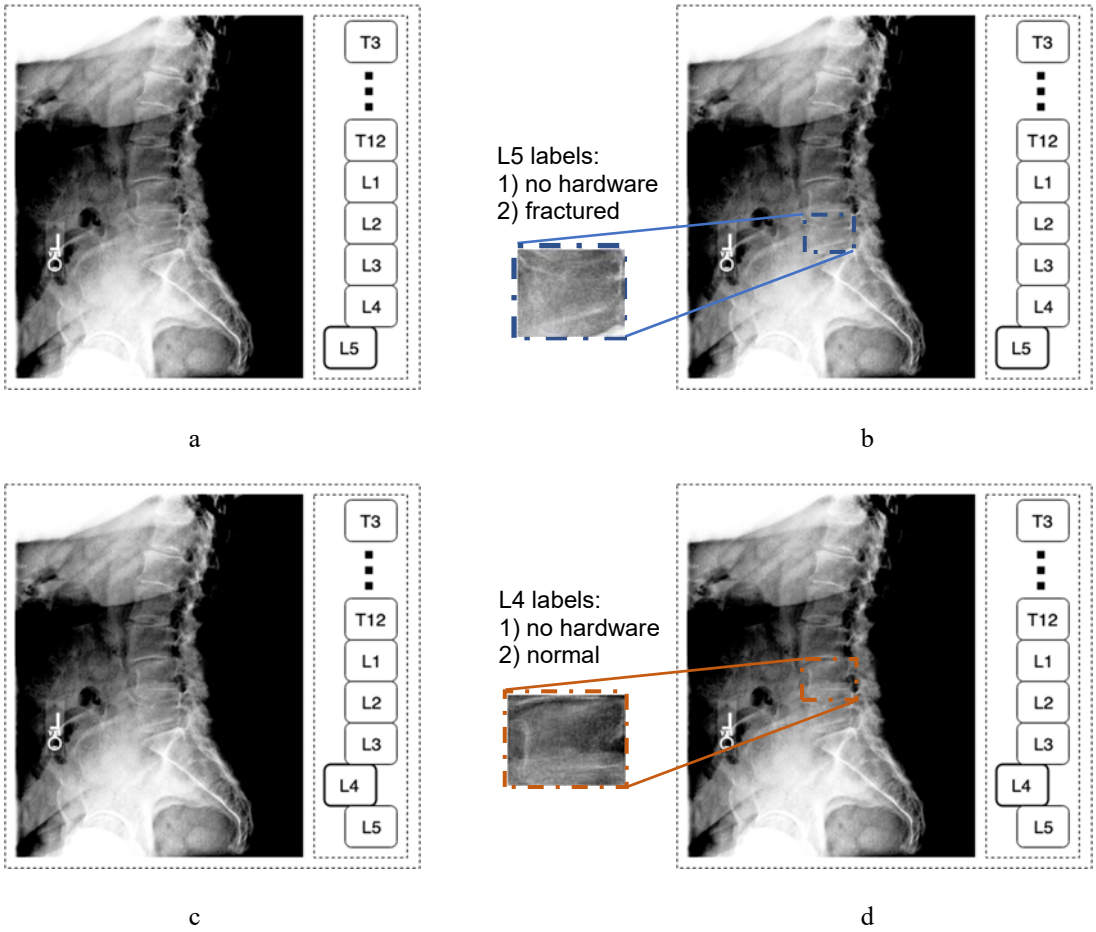


Fig. 1 This demonstrates our approach to annotating multiple regions of interest in an image. The subfigures are: **a** the "L5" region identifier is selected, **b** the annotator provides a bounding polygon for L5 and then inputs the labels for L5 (no hardware, fractured), **c** the "L4" identifier is automatically selected after completing the last labels for L5, and **d** the annotator provides a bounding polygon for L4 and then inputs the labels for L4 (no hardware, normal). This process continues until image annotation is complete

4) For consensus annotations, the program should track which label was made by which annotator. This can facilitate comparing the annotations and computing inter-reader agreement.
5) Most medical images are of a higher bit depth than the standard 8-bit graphic formats like Joint Photographic Experts Group (JPEG) and Graphics Interchange Format (GIF) support. Digital Imaging and Communications in Medicine (DICOM) is the standard format encapsulating medical images, patient information, and relevant imaging metadata [5]. Within a DICOM file, images are often stored in a high-fidelity state, either uncompressed or compressed with lossless and less frequently lossy compression. Depending on the modality, the bit depth is

often higher than 8-bit. Ideally, a program should allow direct viewing of DICOM formatted medical images with window/leveling capabilities to access the full bit depth of the original data. Given the high spatial resolution of many medical images, additional image manipulation functions like zooming and panning are also useful.

Multiple annotation programs for general [3, 4, 6-10] and medical [11-15] images are publicly available. But, to the best of our knowledge, none of them fulfills all of the requirements mentioned above. In addition, some of these programs have limited licensing models that are challenging for a small project's budget. To fill this gap, we developed DicomAnnotator, a configurable open-source software program for DICOM image annotation. This program satisfies the above requirements and provides several user-friendly features to aid and accelerate the annotation process. Although this program is designed for annotating DICOM images, it also supports displaying and annotating JPEG, Portable Network Graphics (PNG), and Tagged Image File Format (TIFF).

2. Methods

This program was designed to fulfill the requirements mentioned above through a user-friendly graphical user interface (GUI). We designed several basic functional modules to satisfy the requirements and four ancillary features to improve user experience. These modules and features are illustrated below using our spine image annotation task as a running example. Given a spine image, our annotators needed to perform five basic annotation sub-tasks: 1) assign an osteoporosis category [16] (normal, possible osteopenia, and definite osteopenia) to the whole image; 2) outline the vertebral bodies using bounding polygons, which are quadrilaterals not necessarily oriented with the x and y axes; 3) determine each vertebral body's anatomic level (e.g., L4, L5, and S1); 4) label whether each vertebral body had any overlying artificial structure (e.g., spine hardware, metallic object, or catheter); and 5) score each vertebral body using one of several fracture classification systems (e.g., the Modified Algorithm-Based Qualitative (m-ABQ) method [17], the Semi-Quantitative method [18], or the Algorithm-Based Qualitative method (ABQ) [19]). In the rest of the paper, we call the labels for the whole image "*image labels*" and the labels for the regions of interest "*region labels.*"

2.1 Functional Modules

The annotation program is divided into five modules: 1) the configuration file, 2) the login page, 3) the image viewing module, 4) the annotation module, and 5) the result file.

The configuration file eases the configuration process, as the annotator can modify the configuration file without touching the program's source code to adapt the program to a new annotation task. The login page allows the annotator to enter a username to track the annotator's annotations. The image viewing module supports image displaying, zooming, panning, and windowing/leveling. The annotation module is used for placing annotations on an image. The result file stores the annotation results and other information generated during the annotation process.

As Fig. 2 shows, the annotator uses the modules sequentially to complete an annotation task, although they are integrated into a common user interface. Before running the program for the first time, the configuration for the given annotation task is set up in the configuration file. After the program starts running, the annotator needs to provide a username on the login page. Next, the main page for image viewing and annotation loads. The annotator uses the GUI to annotate and navigate through the images in the dataset. Throughout this process, the annotations are stored in a result file for later use.

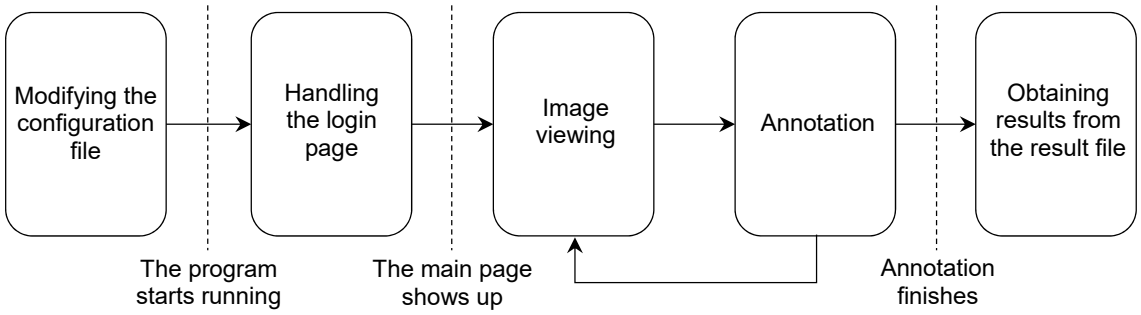


Fig. 2 The steps to using DicomAnnotator to annotate an image dataset

In this section, we show these modules' functionality using a radiograph. In Section A of the Appendix, we show how our program can handle computed tomography (CT) images.

2.1.1 Configuration File

By modifying some attributes in the configuration file, an annotator can adapt the program to a new annotation task. For example, in the spine annotation task, one sub-task is to assign an osteoporosis label to the whole image. The osteoporosis label has three candidate categories: normal, possible osteopenia, and definite osteopenia. In the configuration file, the annotator can type these categories under the "image label" attribute, which is used to customize

the candidate categories of an image label. Then when the program runs, these categories will appear on the program's user interface.

Adjusting some of the attribute values in the configuration file can improve user experience. One example is the "zooming speed" attribute, which determines how quickly the image zooms in/out when the annotator scrolls the mouse wheel.

Customizing the attributes could affect several functional modules including the annotation module, the image viewing module, and the result file. Table 1 lists the main attributes in the configuration file, the attributes' meanings, and the modules affected by the attributes. An example of the configuration file is given in Section A of the Appendix.

Table 1 The main attributes in the configuration file, the attributes' meanings, and the modules affected by the attributes

Attribute	Meaning	Affected module
List of region identifiers	List of regions needing to be labeled for each image, e.g., the anatomic levels (L1, L2, L3…).	The annotation module
Number of vertices of the bounding polygon	Number of vertices of the polygon that is used to capture each region of interest.	
List of region labels	Possible values of each region label. Here, the annotator provides a two-dimensional array. Each element of this array appears on a separate line and contains the possible values of a region label. In the spine image annotation example, the annotator assigns two types of region labels. Accordingly, the array is displayed in two lines. The first line lists the possible values of whether any artificial object overlays the vertebral body: yes and no. The second line lists the possible category values used in a fracture classification system like the m-ABQ method [17].	
Image label	Possible values of the image label.	
Input directory	Directory from which the program reads the input images.	The image viewing module
Zooming speed	Determines how fast the image zooms in/out when the annotator scrolls the mouse wheel.	
Windowing/leveling sensitivity	Controls the rate at which the window and the level change when the mouse moves a unit of length.	
Name/path of the result file	Specifies the file storing the results.	The result file

2.1.2 Login Page

The login page includes a field for an annotator to enter his/her username. When an annotator makes an annotation, the program maps the annotation to the supplied username and records the activity in the result file. In this way, the

program knows the annotation is made by this annotator. When a group of annotators collaborate on an annotation task, this helps the program track which annotation is made by which annotator.

The login page has another function allowing the annotators to quickly set some basic configurations. For an annotation task, if only a few configurations need to be specified, directly customizing them on the login page is more efficient than editing the configuration file. For instance, Fig. 3 shows the login page for the spine annotation task. The first entry is for the annotator to input his/her username. The next two rows are used to select the configurations: the fracture scoring system and the order of the region identifiers shown on the program's main page. When the "START" button is clicked, the program will document the username and open the main page.

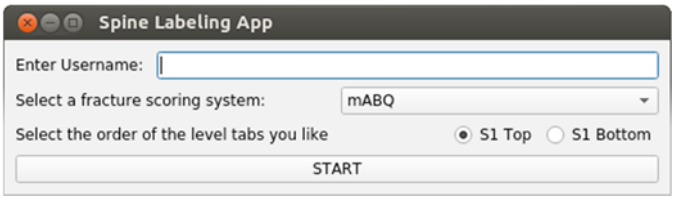


Fig. 3 The login page demonstrating the username field, a dropdown to select a classification system, and radio buttons to determine the order of region identifiers shown on DicomAnnotator's main page

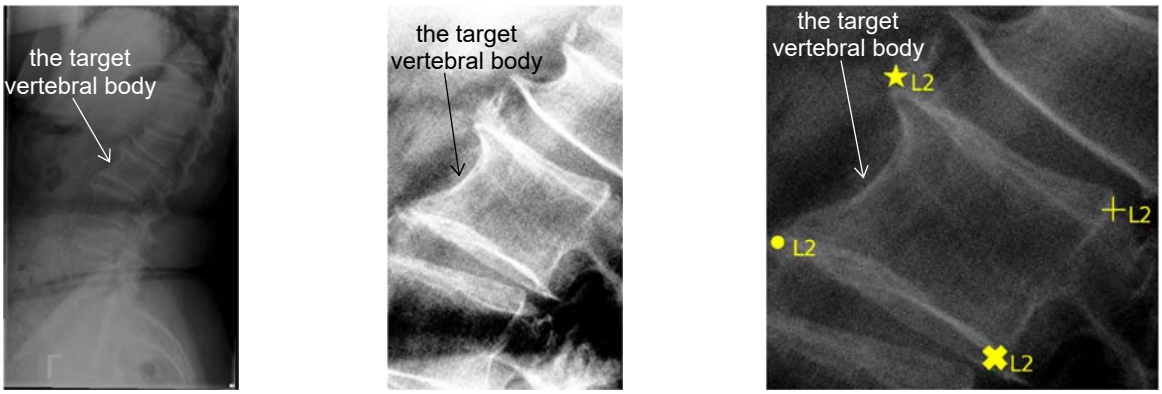
2.1.3 Image Viewing Module

The image viewing module supports displaying DICOM images with a high bit depth and resolution, regardless of the medical imaging modality. It also supports displaying JPEG, PNG, and TIFF images. Fig. 4 is a screenshot showing how our program displays a DICOM image on the main page.

Fig. 4 The main page of DicomAnnotator that is divided into 12 panels. Panel 1: radio buttons used to select an operation mode; Panel 2: a group of buttons that allow the user to move between images, remove annotations, reset the image display, manually save annotations, and display help text.; Panel 3: text box showing details about the currently displayed image and the annotation process; Panel 4: buttons used to set an image to unreadable when it is of low or non-diagnostic quality and to horizontally flip the image; Panel 5: buttons used to flag/unflag an image for later review and to navigate through the flagged images; Panel 6: commenting system where comments from any user are displayed and new comments can be added; Panel 7: indicator of whether new annotations have been stored in the result file; Panel 8: canvas displaying an image; Panel 9: radio buttons for assigning an image label; Panel 10: annotation table which has been configured to apply multiple annotations to each region of interest; Panel 11: buttons used to toggle off the annotated points in the image and to invert the image's grayscale; Panel 12: text boxes showing the identifiers of the regions that are not assigned the default region label like "Normal"

The image viewing module supports zooming, panning, and windowing/leveling. In Fig. 5b, the target vertebral body (L2) has been enlarged and centered using zooming and panning. The image's contrast and brightness has been adjusted using windowing/leveling. By default, zooming is done by scrolling the mouse wheel, panning by holding down the right mouse button and moving the mouse, and window/level adjustment by holding down the "shift" button

on the keyboard and moving the mouse horizontally and vertically to adjust the window and level, respectively. The program can record the window and level values adjusted by the annotator in the result file, allowing the annotator to view the image with these adjusted values when returning to this image.



a b c

Fig. 5 Illustration of user's interaction with the image in the annotation process. **a** The target vertebral body for annotation is identified by the user using the default display parameters; **b** The display window and level, zooming and panning are employed to optimize visualization of the target vertebral body; **c** The boundary of the target vertebral body is marked by placing points at its four corners

2.1.4 Annotation Module

The annotation module is used to place bounding polygons, region labels, and image labels. The approach to placing bounding polygons and region labels satisfies the third requirement mentioned in Section 1. We created an annotation table (Panel 10 of the main page) with multiple tabs, each representing a region of interest like a vertebral body in the spine annotation task. Before placing the bounding polygon and the region labels, the associated tab is highlighted (e.g., S1 is highlighted in Fig. 4). The annotator interacts with the widgets inside the tab to assign the related labels and clicks the canvas to place the bounding polygon.

A bounding polygon is outlined by putting a given number of points on the canvas as the polygon's vertices. Each point is given by clicking a place on the canvas. As shown in Fig. 5c, by clicking the four corners of the vertebral body, the associated points are placed, forming a quadrilateral for extracting the vertebral body.

Finally, an image label is assigned using Panel 9 (see Fig. 4). Fig. 6 shows the final annotations of a spine image.

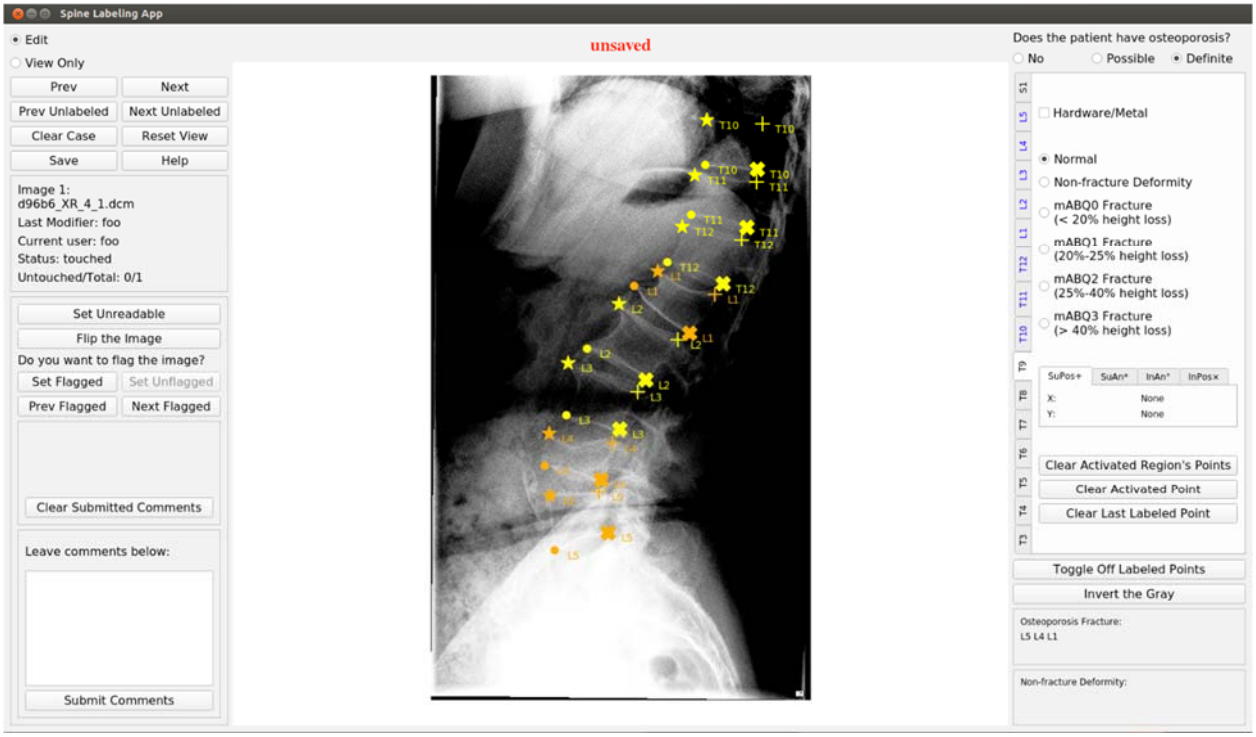


Fig. 6 The final annotations of an example spine image

2.1.5 Result File

The result file stores the annotation results. It also records certain useful information including the username, the window and level values adjusted by the annotator for each image, the annotator's comments (see Section 2.2.2 for the commenting system), and whether each image has been annotated. The username tracks which annotator places which annotation. With the window/level information, the annotator can review an image with its previously adjusted window and level values. The annotator's comments remind the annotator of the concerns raised during annotation. If during the annotation process, the annotator closed and then restarts the program, the recorded information on whether each image has been annotated enables the program to load the first unannotated image and resume the annotation process.

2.2 Ancillary Features

We use four ancillary features to make the annotation program more user-friendly. First, two operation modes, "Edit" and "View Only," are used to enable and disable the annotation module, respectively. Second, a commenting

system allows annotators to leave comments on each image during annotation. Third, an automatic window and level adjustment function is used to reduce the need to adjust the window and level of an image. Fourth, two functions, one for splitting an image set into several parts and the other for merging the result files from multiple annotators, are available to help multiple annotators collaborate on the same annotation task.

2.2.1 Operation Modes

Our software has two operation modes. The "Edit" mode allows the annotator to annotate the images. The "View Only" mode disables the annotation module, preventing new annotations to be put on the images and allowing the annotator to only view the images and the previously placed annotations. As Fig. 7 shows, when the "View Only" mode is chosen, the widgets for placing annotations (e.g., the annotation table) are disabled. On the canvas, the function for placing the polygon's vertices is disabled, while image manipulations are still allowed.

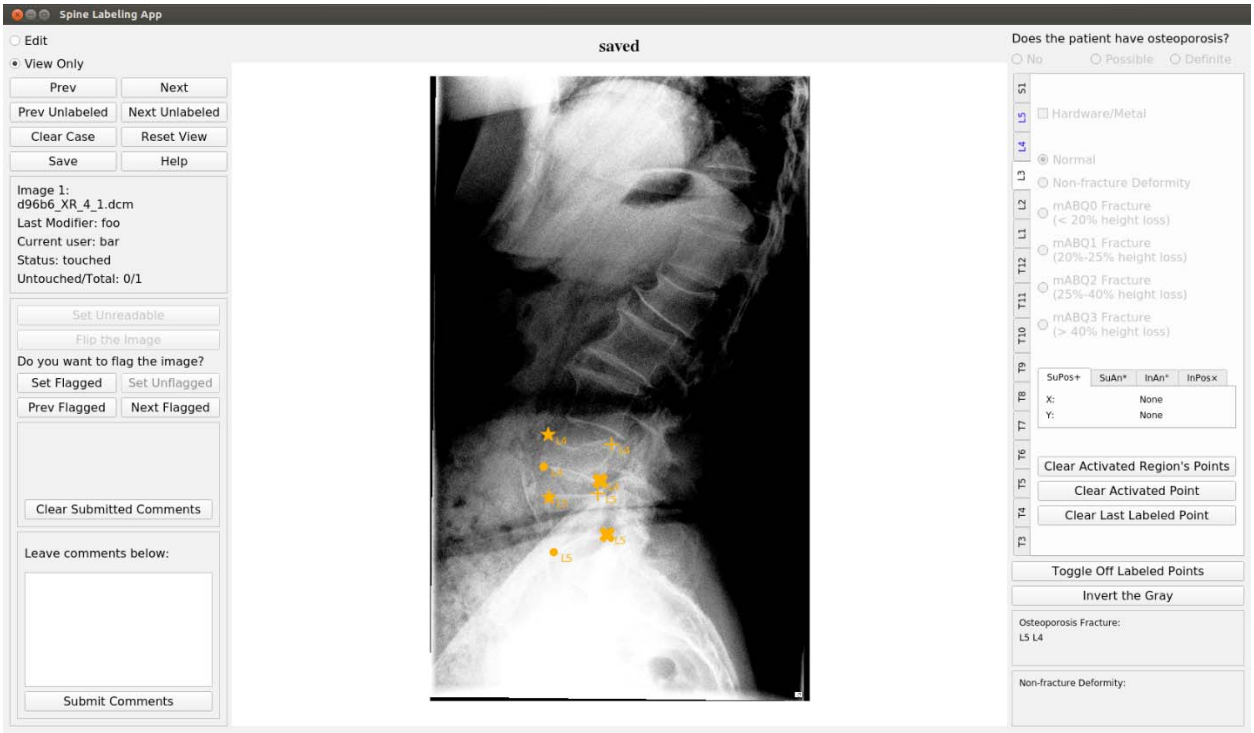


Fig. 7 The main page in the "View Only" mode demonstrating the annotations with the manipulation tools greyed out to prevent accidental alteration of the annotations. The user can return to the "Edit" mode by clicking the radio button in the upper left

The “View Only” mode is useful when an annotator checks the annotations. Disabling the annotation module can avoid unintended modification to the existing annotations, e.g., clicking the canvas and accidentally placing an extra point. Depending on whether the annotator is annotating the image or checking the annotations supplied by others, our program can automatically select a proper mode. If the image is previously annotated by another annotator, the program regards the current annotator to be checking the existing annotations and switches to the “View Only” mode. Otherwise, the program runs in the “Edit” mode.

Regardless, annotators can manually switch the mode by selecting one of the radio buttons on Panel 1 (see Fig. 4). When the annotator tries to switch from the “View Only” mode to the “Edit” mode, a dialog (Fig. 8) pops up asking the annotator for confirmation. This further reduces the risk of accidentally modifying the existing annotations.

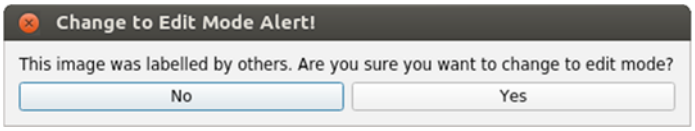


Fig. 8 The confirmation dialog that is displayed when switching from the “View Only” mode to the “Edit” mode

2.2.2 Commenting System

During the annotation process, an annotator could have concerns on issues like the image quality and the placement of certain annotations. We embedded a commenting system into the program to document the annotator’s thoughts. Panel 6 of Fig. 4 gives the commenting system’s user interface. For each image, comments can be entered in the text box which are saved to the result file after the annotator clicks “submit comments.” The commenting system allows different annotators to comment on the same image. Fig. 9 shows how such comments are displayed.

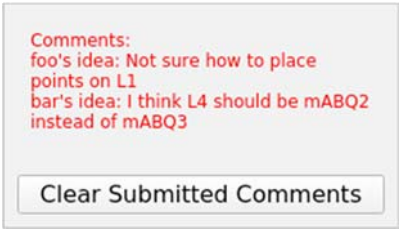


Fig. 9 An example of comments displayed in the comment panel for an image

2.2.3 Automatic Window and Level Adjustment

To reduce an annotator's efforts of adjusting the window and level, we embedded an automatic window and level adjustment function based on contrast stretching [20, Chapter 3]. Given an image, let B denote the image's bit depth. 2^B-1 is the maximum intensity level the image can reach. A patch at the center of the image is extracted. The patch's width and height are 1/4 of the image's width and height, respectively. Let p_{max} and p_{min} denote the maximum and minimum intensity levels of the patch, respectively. In the regions of interest on the image, we regard most pixels' intensity levels to be within the range of p_{min} to p_{max}. For each image pixel, we map its intensity level I to

$$I'=(2^B-1)(I-p_{min})/(p_{max}-p_{min}).$$

This horizontally stretches the image's histogram, mapping p_{min} and p_{max} in the original histogram to zero and 2^B-1 in the transformed histogram, respectively. After doing the mapping, some pixels' intensity levels could become $>2^B$-1 or <0. For each intensity level $>2^B$-1, we set it to 2^B-1. For each intensity level <0, we set it to 0.

2.2.4 Splitting an Image Set and Merging Result Files

Multiple annotators can use our program to collaboratively annotate a set of images in the following way:

1) Subsets of the image set are made, one per annotator. Each subset of images is put into a separate folder.
2) For each annotator, a copy of the program and the folder containing the assigned images is moved into another folder termed the annotation folder. The annotation folder is transferred to one of the following locations:
   a. A server accessible by all of the annotators: Each annotator's folder goes into the corresponding home folder. The annotator then logs into his/her account and uses the program to annotate the images.
   b. When each annotator uses his/her own computer: Electronically send (e.g., SCP, secure FTP, and website download) the annotation folder and the instructions for installing the program on his/her own computer.
   c. In a cloud environment (e.g., Amazon Web Services, Google Cloud, and Microsoft Azure): The annotation folder is copied to the annotator's individual cloud instance. The annotator could use Virtual Network Computing (VNC), which allows him/her to access the GUI of his/her cloud instance.
3) Each annotator runs DicomAnnotator and annotates the assigned images. If the annotator uses a Linux system, there is a Bash script to start DicomAnnotator with an associated environment containing all the necessary dependencies. This gives Linux novices an easy way to run DicomAnnotator without having to manage the Python environment and type complex Linux commands.

4) After the images are annotated, the administrator of the annotation team can obtain and merge the annotators' result files.

The program offers the functions of splitting the image set into multiple subsets and merging the result files from multiple annotators.

2.3 Usability Evaluation

2.3.1 Users' Backgrounds

To understand the usability of DicomAnnotator, we conducted a usability evaluation on six users of various backgrounds. As Table 2 shows, half of the users were neuroradiologists. The other half were undergraduate and graduate students. These six users had a wide variety of familiarity with medical imaging: from no experience to several decades of experience. Before doing the annotation, the neuroradiologists had never seen the program. The undergraduate and graduate students had used the program to view a small number of images. The number of cases that each user reviewed with the program was recorded.

Table 2 Description of each user in the usability evaluation listing his/her occupation, medical imaging experience, and the number of images each annotated

User ID	**Background**	**Experience with medical imaging (years)**	**Number of images annotated**
1	Neuroradiologist	5	9
2	Neuroradiologist	7	9
3	Neuroradiologist	31	9
4	Graduate student	0	9
5	Undergraduate student	0	60
6	Undergraduate student	0	60

The literature suggests using six users can identify most of the usability problems. Nielsen [21, Chapter 6] proposed an empirical formula $N(1-(1-L)^n)$ to estimate the number of usability problems that a group of users can find collectively. Here, N is the total number of usability problems. L is the percentage of usability problems a user can

find on average. A typical value of L is 31%. n is the number of users. This formula indicates that six users can identify ~89% of the usability problems.

2.3.2 Process of Usability Evaluation

As Fig. 10 shows, the usability evaluation process had two components: an annotation session followed by a survey session. In the annotation session, the six users were asked to annotate spine radiographs. DicomAnnotator's log file was used to analyze the program's usability. In the survey session, we did a Web-based survey on these users.

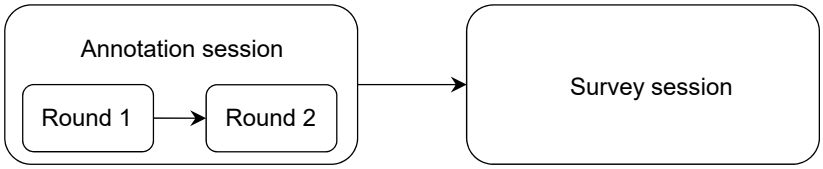


Fig. 10 The usability evaluation process

The program and radiograph DICOM files were used on a server running the Ubuntu operating system (16.04.6 LTS). From his/her personal computer, each user remotely logged into the server via a VNC client to do the annotation tasks. Before the annotation session, we created a user account for each user on the server and gave each user a Bash script to run the program.

During the annotation session, the users performed several annotation tasks including: 1) assigning an osteoporosis category to the whole image; 2) identifying the four corners of each visible thoracic/lumbar vertebral body; 3) determining each vertebral body's anatomic level; 4) identifying any overlying artificial structure; 5) using the m-ABQ method to score each vertebral body [17]; 6) identifying unreadable radiographs due to the wrong study type (cervical spine, chest, abdominal, or pelvic radiographs), a non-lateral image, or low image quality; and 7) identifying incorrect orientation of the image and flipping it as needed to ensure the patient faces to the left of the screen.

As Fig. 10 shows, the annotation session included two rounds. The first round used nine radiographs. After being introduced to the program, each of the three radiologists and the graduate student did all of the tasks on the nine radiographs. The second round used two different sets of radiographs: the first with 20 radiographs and the second with 40 radiographs. After being oriented to the program and some basic spine anatomy, two undergraduate students with no prior experience in medical imaging did tasks 2, 3, 4, 6, and 7. Each of these two students practiced on the first set of radiographs and then formally annotated the second set of radiographs. In each of the two rounds, each user

was given two weeks to do the assigned tasks at his/her own pace. The data logged by the program were used to estimate the average amount of time the user spent on a radiograph. In the second round, the 20 radiographs used for practice were unused for estimating the average amount of time the user spent on a radiograph.

In the survey session, we used Google Forms to do a Web-based survey to understand the users' thoughts on the program. We used the usability factors defined in Dabbs *et al*. [22] to design our survey questions. The usability factors and the survey questions are listed in Section B of the Appendix. Each of the first seven questions measures a distinct usability factor. The last open-ended question is used to gather additional user feedback. In Question 4, if a user chose one or more errors, we asked the user to also list the errors in a text box.

3. Results

3.1 DicomAnnotator

We built DicomAnnotator in Python 3.7.3 and used the Anaconda3 distribution to manage packages. The main packages include PyQt5 for designing the user interface, SimpleITK [23] for reading DICOM images, Matplotlib for displaying images, and NumPy for processing arrays and matrices. DicomAnnotator and its installation instructions are available at https://github.com/UW-CLEAR-Center/DICOM-Annotator.

3.2 Usability Evaluation

3.2.1 The Average Amount of Time to Annotate a Radiograph

Table 3 lists each user's average amount of time to annotate a radiograph, as well as the mean and the standard deviation of these numbers in each round in the annotation session. The undergraduate students did only some, but not all of the annotation tasks assigned to the neuroradiologists and the graduate student. As tasks take different amounts of time to complete, there is no basis to directly compare the average amount of time the undergraduate students spent on annotating a radiograph with that of the neuroradiologists and the graduate student.

Table 3 A summary of each user's average amount of time needed to annotate a radiograph. Note: users 5 and 6 did only some, but not all of the annotation tasks assigned to users 1-4

User ID	**Average amount of time to annotate a radiograph (seconds)**	**Mean (standard deviation) of the average amount of time needed to annotate a radiograph (seconds)**
1	243.1	284.1 (43.7)
2	275.4	
3	345.9	
4	271.8	
5	91.8	82.8 (12.7)
6	73.8	

3.2.2 The Survey Results

For each round in the annotation session and each of the survey questions 1, 2, 3, 5, 6, and 7, Table 4 shows the mean and the standard deviation of the ratings the users gave in their responses to the question, as well as the *p*-value of the *t*-test that checks whether these two rounds have the same mean of the ratings.

Table 4 A summary of the ratings for the survey questions in the first round and second round of the annotation session, and the *p*-values to describe the difference in the mean of the ratings between these two rounds. The survey questions used to assess the usability factors are listed in Table 5. Note: results of survey question 4 are not included because it uses a list of ranges which are difficult to summarize with a single mean and standard deviation. However, only one user reported errors

Sequence number of the question	**Usability factor**	**Mean (standard deviation) of the ratings given by the users in the first round of the annotation session**	**Mean (standard deviation) of the ratings given by the users in the second round of the annotation session**	***p*-value**
1	Learnability	4.50 (0.58)	3.00 (1.41)	0.39
2	Effectiveness	5.00 (0.00)	4.50 (0.71)	0.50
3	Efficiency	4.75 (0.50)	3.50 (0.71)	0.15
5	Flexibility	4.50 (0.58)	4.00 (1.41)	0.71
6	Memorability	4.75 (0.50)	4.00 (1.41)	0.60
7	User satisfaction	4.75 (0.50)	4.00 (1.41)	0.60

In Question 4, five users reported encountering no severe error in the program. One user in the second round of the annotation session reported running into severe errors 4-10 times. In particular, the user reported errors running the Bash script to start the program, and said that every time he/she was able to get around by copying the commands from the Bash script to the terminal and running them one by one to start the program. Thus, it is likely that either that copy of the Bash starting script contained some errors or the user used the Bash script incorrectly, rather than the errors being attributable to DicomAnnotator. This behavior has not been reproduced and no other users have since complained of this problem.

The users' comments to Question 8 include: 1) "This is an excellent piece of software that is intuitive and remarkably stable;" 2) "Windowing/leveling remains somewhat challenging;" 3) "I wish that there was a brightness/contrast bar to adjust the images instead of having to press down on shift and move your mouse a certain way, because it's not very straightforward to know which way to shift the mouse at which angle to adjust the brightness." The first two comments

came from the users in the first round of the annotation session. The third comment came from a user in the second round of the annotation session.

4. Discussion

Our annotation program fulfills the requirements mentioned in Section 1 and offers several user-friendly features. The usability evaluation's annotation session showed that without previously using the program, users of various backgrounds can use our program to quickly annotate medical images with multiple types of labels. This demonstrates that the program is easy to learn and efficient to use. The survey results showed that most users thought our program is easy to learn, effective, efficient to use, and flexible. They agreed the program's features and functions are easy to remember. Overall, they were satisfied with our program. One user reported running into severe errors. Yet, after working with the user, we found the errors are likely not due to the program. As there are no contradictory user complaints, we consider the program to be stable.

Based on the users' feedback, we optimized the program. Two users requested additional options for controlling window and level settings. One of these two users suggested brightness/contrast bars. Yet, after discussing with the other users, we reached the consensus that using brightness/contrast bars is inefficient. If the brightness/contrast bars were used, an annotator would need to frequently move his/her mouse cursor between the brightness/contrast bars and the program's annotation area (the main page's canvas or annotation table). To avoid this inefficiency, we give the user multiple ways to adjust the window and level: 1) hold down the "shift" key and move the mouse horizontally and vertically to adjust the window and level, respectively; and 2) on the keyboard, press "a" or "d" to increment or decrement the window setting and "w" or "s" to increment or decrement the level setting.

There are several interesting areas for future work. The program is currently optimized for displaying single images. The tasks for annotating video or stacks of cross-sectional data could require other types of annotations, like assigning labels on the entire video or stack of images from a cross-sectional series. This would require new functionality to be built into the program. Moreover, some annotation tasks could require a specific shape or aspect ratio of the bounding polygon like a rectangle. New functions for constraining the bounding polygon's shape can be built into the program.

5. Conclusion

Conducting medical image classification, image segmentation, and object detection usually requires many annotated images. To reduce the burden of manual annotation, we designed DicomAnnotator, a DICOM image annotation program. It integrates multiple functional modules to meet several annotation requirements and provides four ancillary user-friendly features. The program is easy to learn, is efficient to use, and allows annotators to quickly make several types of annotations on a large set of DICOM images.

Appendix

A. An Example of Modifying the Configuration File for Annotating CT Images

We give an example of modifying the configuration file for annotating spine CT images. This annotation task requires an annotator to annotate lumbar vertebral bodies in a sequence of four sub-tasks: 1) decide each vertebral body's anatomic level; 2) outline each vertebral body by placing four points at the vertebral body's corners and two points in the middle of the vertebral body's endplates; 3) for each vertebral body, check whether it is fractured and whether any artificial object overlays it; 4) for the CT image, decide an osteopenia score for the spine based on Saville's method [24], which used five different grades (Grades 0 to 4) to describe various osteopenia severities. Sub-task 2 is to place the bounding polygon. Sub-task 3 is to assign labels to each region of interest. Sub-task 4 is to provide the image label.

For this annotation task, we modify the configuration file as shown in Fig. 11. Fig. 12 shows DicomAnnotator's main page after the annotation task is done. In the configuration file, the image_label_description attribute describes the image label and is shown at the top right corner of the main page (see Fig. 12), where the annotator selects an image label. The region_labels attribute gives all possible region labels. There, the checkbox sub-attribute shows those possible region labels, each of which appears as a separate check box in the annotation table on the main page (see Fig. 12). The radiobuttons sub-attribute shows those possible region labels that are listed as radio buttons in the annotation table on the main page (see Fig. 12). The radiobuttons sub-attribute is equivalent to the "list of region labels" attribute listed in Table 1. Table 1 also describes the other attributes shown in Fig. 11.

```
{
        "input_directory": "images/",
        "zooming_speed": 1.2,
        "window_level_sensitivity": 1,
        "path_for_result_file": "results.csv",
        "list_region_identifiers": ["L5", "L4", "L3", "L2", "L1"],
        "image_label_description": "Osteopenia grade",
        "image_label": ["0", "1", "2", "3", "4"],
        "region_labels":
        {
                "checkbox": "Artificial structure overlying",
                "radiobuttons": [["Normal", "Fractured", "Unsure"]]
        },
        "number_bounding_polygon_vertices": 6
}
```

Fig. 11 The configuration file for the spine CT image annotation task in JSON format

Fig. 12 DicomAnnotator demonstrating display and annotations of an example sagittal lumbar spine CT image

B. Usability Factors and Survey Questions

The usability factors and the survey questions are listed in Table 5.

Table 5 These survey questions were used to determine the usability of DicomAnnotator across a variety of factors

Sequence number of the question	Usability factor	Question
1	Learnability	Initially, how easy was it to learn to use the program and its functions? Ratings are on a 1-5 scale with anchors of difficult/easy.
2	Effectiveness	Does the program include all of the functions you need for completing your image annotation task? Ratings are on a 1-5 scale with anchors of not at all/everything I needed.
3	Efficiency	Did the program help you annotate the images efficiently? Ratings are on a 1-5 scale with anchors of inefficient/efficient.
4	Errors	When using the program, how many times did you encounter severe errors such as software crash, software having no response, and annotation not stored? The choices for the response are 0, 1-3, 4-10, 11-20, and 20+.
5	Flexibility	Does the program give enough shortcuts and ways of access (e.g., window/level, pan, and navigating images) for the annotation task? Ratings are on a 1-5 scale with anchors of cumbersome or hard to use functions/fast and easy access to all of the functions.
6	Memorability	Are the program's features and functions easy to remember? Ratings are on a 1-5 scale with anchors of hard/easy.
7	User satisfaction	Overall, are you satisfied with the program? Ratings are on a 1-5 scale with anchors of unsatisfied/satisfied.
8		Please provide comments, thoughts, or features that you wish DicomAnnotator to have, if any, in the text box below.

Authors' Contributions

NC, DH, and JGJ conceptualized and designed the study. QD did the coding implementation and evaluation of DicomAnnotator. QD, DH, and NC performed literature review. QD wrote the initial draft of the paper. GL and NC

extensively edited and revised the paper. DH, MO, KL, ZY, and JGJ revised the paper. DH, GL, JGJ, and NC contributed equally to the paper.

References

1. Sa R, Owens W, Wiegand R, Studin M, Capoferri D, Barooha K, Greaux A, Rattray R, Hutton A, Cintineo J, Chaudhary V: Intervertebral disc detection in X-ray images using faster R-CNN. In: Proceedings of International Conference of the IEEE Engineering in Medicine and Biology Society, 2017, pp 564-567.
2. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S: Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639):115-118, 2017.
3. Labelbox. The leading training data solution. Available at https://labelbox.com. Accessed 18 June 2019.
4. Dataturks. Best online platform for your ML data annotation needs. Available at https://dataturks.com. Accessed 18 June 2019.
5. Pianykh OS: Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide, 2nd edition, Berlin: Springer, 2012.
6. Halaschek-Wiener C, Golbeck J, Schain A, Grove M, Parsia B, Hendler J: Photostuff-an image annotation tool for the semantic web. In: Proceedings of the 4th International Semantic Web Conference, 2005, pp 6-10.
7. Tuffield M, Harris S, Dupplaw DP, Chakravarthy A, Brewster C, Gibbins N, O'Hara K, Ciravegna F, Sleeman D, Wilks Y, Shadbolt NR: Image annotation with photocopain. In: The First International Workshop on Semantic Web Annotations for Multimedia (SWAMM 2006) at WWW 2006, 2006.
8. Saathoff C, Schenk S, Scherp A: Kat: the k-space annotation tool. In: K-space plenary meeting co-located at Int. Conf. on Semantic and Digital Media Technologies, 2008.
9. Nieto X, Camps N, Marques F: GAT: a graphical annotation tool for semantic regions. Multimed Tools Appl 46(2-3):155-174, 2010.
10. Dutta A, Zisserman A: The VIA annotation software for images, audio and video. In: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp 2276-2279.
11. Rueden CT, Schindelin J, Hiner MC, DeZonia BE, Walter AE, Arena ET, Eliceiri KW: ImageJ2: ImageJ for the next generation of scientific image data. BMC Bioinformatics 18(1):529, 2017.

12. McAuliffe MJ, Lalonde' FM, McGarry D, Gandler W, Csaky K, Trus BL: Medical image processing, analysis and visualization in clinical research. In: Proceedings of the 14th IEEE Symposium on Computer-Based Medical Systems, 2001, pp 381-386.
13. Philbrick KA, Weston AD, Akkus Z, Kline TL, Korfiatis P, Sakinis T, Kostandy P, Boonrod A, Zeinoddini A, Takahashi N, Erickson BJ: RIL-Contour: a medical imaging dataset annotation tool for and with deep learning. J Digital Imaging 32(4):571-581, 2019.
14. Rubin DL, Akdogan MU, Altindag C, Alkim E: ePAD: an image annotation and analysis platform for quantitative imaging. Tomography 5(1):170, 2019.
15. MD.ai. The platform for medical AI. Available at https://www.md.ai. Accessed 18 June 2019.
16. Masud T, Mootoosamy I, McCloskey EV, O'Sullivan MP, Whitby EP, King D, Matson MB, Doyle DV, Spector TD: Assessment of osteopenia from spine radiographs using two different methods: the Chingford study. Br J Radiol 69(821):451-456, 1996.
17. Lentle BC, Berger C, Probyn L, Brown JP, Langsetmo L, Fine B, Lian K, Shergill AK, Trollip J, Jackson S, Leslie WD: Comparative analysis of the radiology of osteoporotic vertebral fractures in women and men: cross-sectional and longitudinal observations from the Canadian multicentre wsteoporosis study (CaMos). J Bone Miner Res 33(4):569-579, 2017.
18. Genant HK, Wu CY, Van Kuijk C, Nevitt MC: Vertebral fracture assessment using a semiquantitative technique. J Bone Miner Res 8(9):1137-1148, 1993.
19. Jiang G, Eastell R, Barrington NA, Ferrar L: Comparison of methods for the visual identification of prevalent vertebral fracture in osteoporosis. Osteoporos Int 15(11):887-896, 2004.
20. Gonzalez RC, Woods RE: Digital Image Processing, 4th edition, New York, NY: Pearson, 2018.
21. Nielsen J: Usability Engineering, San Francisco, CA: Morgan Kaufmann, 1993.
22. Dabbs AD, Myers BA, Mc Curry KR, Dunbar-Jacob J, Hawkins RP, Begey A, Dew MA: User-centered design and interactive health technologies for patients. Comput Inform Nurs 27(3):175-183, 2009.
23. Lowekamp BC, Chen DT, Ibáñez L, Blezek D: The design of SimpleITK. Front Neuroinform 7:45, 2013.
24. Saville PD: A quantitative approach to simple radiographic diagnosis of osteoporosis: its application to the osteoporosis of rheumatoid arthritis. Arthritis and Rheumatism 10:416-422, 1967.