

# Quantitative Image Feature Engine (QIFE): an Open-Source, Modular Engine for 3D Quantitative Feature Extraction from Volumetric Medical Images

Sebastian Echegaray<sup>1</sup>  · Shaimaa Bakr<sup>2</sup> · Daniel L. Rubin<sup>1,3</sup> · Sandy Napel<sup>1</sup>

Published online: 6 October 2017  
© Society for Imaging Informatics in Medicine 2017

**Abstract** The aim of this study was to develop an open-source, modular, locally run or server-based system for 3D radiomics feature computation that can be used on any computer system and included in existing workflows for understanding associations and building predictive models between image features and clinical data, such as survival. The QIFE exploits various levels of parallelization for use on multiprocessor systems. It consists of a managing framework and four stages: input, pre-processing, feature computation, and output. Each stage contains one or more swappable components, allowing run-time customization. We benchmarked the engine using various levels of parallelization on a cohort of CT scans presenting 108 lung tumors. Two versions of the QIFE have been released: (1) the open-source MATLAB code posted to Github, (2) a compiled version loaded in a Docker container, posted to DockerHub, which can be easily deployed on any computer. The QIFE processed 108 objects (tumors) in 2:12 (h/mm) using 1 core, and 1:04 (h/mm) hours using four cores with object-level parallelization. We developed the Quantitative Image Feature Engine (QIFE), an open-source feature-extraction framework that focuses on modularity, standards, parallelism, provenance, and integration. Researchers can easily integrate it with their existing segmentation and

imaging workflows by creating input and output components that implement their existing interfaces. Computational efficiency can be improved by parallelizing execution at the cost of memory usage. Different parallelization levels provide different trade-offs, and the optimal setting will depend on the size and composition of the dataset to be processed.

**Keywords** 3D Image features · Feature extraction · Quantitative imaging · Radiomics

## Introduction

Radiomics [1–7] is science that studies the extraction and quantification of explicit image features from imaging studies (e.g., CT, MR, ultrasound, optical coherence tomography, mammography, microscopy) and their relationship to other clinical data (e.g., survival, molecular phenotype, comorbidities). It involves delineation of a volume of interest (VOI) (e.g., surrounding a tumor or a region of interstitial lung disease), followed by computation of features of the VOI and the voxels within it (e.g., shape, margin sharpness, intensity distribution, and texture). Many researchers are engaged in radiomics studies, and most often, they use their own software to compute the image features. The use of disparate computation engines leads to difficulties in comparing results across research groups and in aggregating them for better-powered studies. In response to this, we developed a free, open-source, modular, feature extraction framework that executes a set of tested feature-extraction algorithms that can be used in radiomics research. It is also designed to run on a server so that research groups can use the same exact computations, allowing their results to be compared and aggregated.

---

✉ Sebastian Echegaray  
sechegaray@gmail.com

<sup>1</sup> Department of Radiology, Stanford University School of Medicine, 300 Pasteur Drive, Stanford, CA 94305, USA

<sup>2</sup> Department of Electrical Engineering, Stanford University, 650 Serra Mall, Stanford, CA 94305, USA

<sup>3</sup> Department of Medicine (Biomedical Informatics Research), Stanford University School of Medicine, 300 Pasteur Drive, Stanford, CA 94305, USA

While there have been other efforts in the academic and industrial worlds to create frameworks for collaborative computational research [8–13], there are few available tools for image feature computation and analysis. One such tool is IBEX [13], which provides an end-to-end solution with a graphical interface for researchers to annotate image datasets, compute radiomics features, and analyze the resulting feature data. A more recently released software package is Pyradiomics [14], which provides a set of python libraries that can be invoked to process one or multiple image series and compute radiomics features. Our system, the Quantitative Image Feature Engine (QIFE), is a locally run or server-based system exclusively for radiomics feature computation. Our tool differentiates itself from the others by focusing on out-of-the-box parallel processing, modularity, and its ability to be plugged into existing workflows. It accepts as input DICOM files containing volumetric data together with DICOM Segmentation Objects (DSOs) specifying the volumes of interest (objects) to be processed. Code is also provided to modify the images and/or segmentations to allow simulation of different segmentations, acquisition, and/or reconstruction conditions. We developed the QIFE using MATLAB 8.6 R2015b (Mathworks, Natick, MA), exploiting its ability to generate parallel threads for performance improvement on multiprocessor systems, and we distributed it in multiple formats, including an open-source code repository to allow collaboration and modification, and a self-contained Docker [15] image for easy tool sharing. Sharing code widely allows the QIFE to be continuously tested, and any identified bugs can be fixed and re-shared to the community [16–18]. Finally, we have put a strong emphasis on provenance and research attribution. As part of its output, the QIFE stores the date and time, the version that was run, its runtime configuration parameters, and a citation defining each feature computed, fostering reproducible science [19, 20].

“Quantitative Image Feature Engine (QIFE) Architecture” section presents the architecture of the engine describing its internal workings and organization. “Included Components” section lists and describes all components included in the engine at the moment of this publication. “Speed and Memory Tests” section describes methods and results of an experiment using a large cohort to analyze the time and memory requirements using different modes of parallel processing. “Feature Comparison” section describes how to obtain the engine, either in as source code or in a Docker image for execution. “Current Use” section discusses several use cases currently underway. “Limitations” section discusses some limitations of this software. Finally, “Conclusions” section summarizes and offers concluding remarks.

## Materials and Methods

### QIFE Architecture

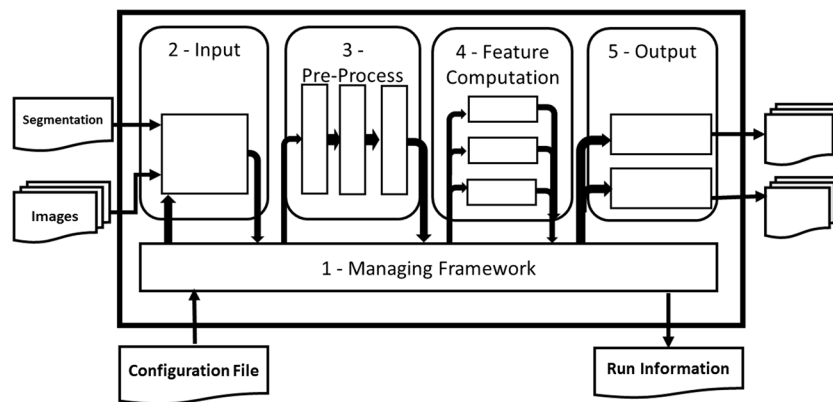
The QIFE generates quantitative features from a volume of interest, defined by a DSO, from a DICOM series. It consists of a managing framework and four stages: input, pre-processing, feature computation, and output, as shown in Fig. 1. Each stage contains one or more components that are loaded at run time. The managing framework provides functions to facilitate the communication between components and stages. The output of each stage serves as the input for the next one. We designed the stages to be modular, such that each component can be swapped out for another one as long as the new one implements the stage-defined interface. Information is passed from the framework to each component on request. When a component finishes processing, it returns its output to the framework for collation.

The Managing Framework (1) is the fixed structure of the engine, within which the stages and components communicate. It parses the configuration file at the beginning of the run, loads the necessary components, provides global helper functions for each component, facilitates communication between components, and collates the results from each stage.

The configuration file is a text file that defines which components to load, sets global configuration parameters, and overrides individual default parameters in each of the component. Appendix 1 lists all of parameters currently accepted by this configuration file, and Appendix 2 gives an example of such a configuration file.

The input Stage (2) loads the data that will be processed by the engine. This includes DICOM files containing one or more image series and one or more DSOs describing the location of the VOIs within each relevant DICOM image series. The components in this stage receive the path and filenames or Unique identifier (UID) [21] of the image series that need to be processed and loads them into memory. The output of this stage includes an array storing metadata for each slice (e.g., slice location, thickness, etc.) and two 3D matrices, one for the intensity values in the scan and one for the binary mask representing the volume of interest. The output of this stage is submitted to the Managing Framework.

The Managing Framework then submits the output of the Input Stage to the Pre-processing Stage (3), which then executes code to alter the original scan data. Components in the pre-processing stage can be invoked to apply filtration, add noise, and/or alter the VOI to facilitate experiments, e.g., to test stability of the features extracted with respect to imaging and segmentation conditions. The components of this stage are applied sequentially, using the processing order established in a configuration table, with the output of each component applied to the input of the following component. After all components have executed, the stage outputs its results to the Managing Framework.



**Fig. 1** The Quantitative Image Feature Engine (QIFE) architecture. The rounded rectangles represent stages, and the rectangles within represent components. Data flows occur from left to right. The Managing Framework (1) reads the configuration file and sets up all stages for processing. The input stage (2) loads the segmentation and image files into the engine, and then the pre-processing stage (3) applies its components sequentially to normalize or otherwise modify the input

The Managing Framework then submits the output of the Input Stage to the Feature Computation Stage (4), which contains components that compute quantitative features describing voxel data within the VOIs defined by the supplied DSOs. Each feature component can return a single value, a group of values, and/or an image. Finally, this stage collates all results into a structure and submits it to the Managing Framework.

Finally, the Managing Framework submits the results of the Feature Computation Stage to the Output Stage (5), which contains components to convert the outputs to user-readable formats. Components in this stage can generate multiple outputs depending on the configuration. For example, the CSV component groups results per patient and writes them into CSV files for easy import in other tools, and the cross-sectional image generator component outputs example images.

We next describe the specific components that we have implemented.

### Included Components

In this section, we describe the components currently included in the QIFE. For an up-to-date list and description of all features in the current version of the engine included, please refer to our Github repository <https://github.com/riipl/3dpipeline>.

#### Input Stage

Components in the input stage load data from outside the engine and send it to the managing framework for further processing by the following stages. The engine currently implements a single component to load DICOM image files and DSOs for image and segmentation data, respectively.

The DSO/DICOM loader loads DSOs [22] and their referenced DICOM [21] image files. Parameters required by this

data appropriately. Next, the feature computation stage (4) runs each of its components independently and extracts quantitative descriptors of the images and segmentation. Finally, the output stage (5) generates files summarizing the results. At the end of execution, the managing framework outputs a log file with the information required to reproduce the results

component include the path of the directory containing the original DICOM images and the path of the directory containing the DSOs. This component consists of three different sub-components: indexing, segmentation-loading, and scan reading.

First, the indexing sub-component scans all the directories provided and checks the type of all files with the extension “DCM.” It then determines if the file is a valid DICOM image or DSO; if not, the file is logged and ignored. The component generates three hash tables: (1) mapping the DSOs’ UIDs to their file path, (2) mapping DICOM images’ UIDs to their file path, and (3) mapping DICOM Series’ UIDs and Instance number [21] to their file path. The first and second hash tables are used to minimize the time required to link the individual bit planes in the DSOs to the correct DICOM images. The third hash table is used to find DICOM images that are not referenced by the DSOs but might be needed for certain features that use values outside the segmentation (e.g., edge features). The hash tables can be stored in the root directories of the provided paths to avoid re-computation in future runs with the same dataset.

Next, the segmentation-loading sub-component loads the DSO corresponding to the scan being analyzed into memory. This sub-component sorts the segmentation planes by their slice location parameter in the DSO’s DICOM header.

Finally, the object-reading sub-component loads the corresponding DICOM images referenced by the DSO. As some computations (for example, edge features) will require images superior and inferior to the segmentation volume itself, a padding parameter (in mm) is provided. In this case, images superior and inferior to the most superior and inferior aspects, respectively, of the segmentation VOI that are closer than the distance provided by the padding parameter are loaded into the appropriate locations superior and inferior to the VOI.

The intensity values of all loaded images are then normalized [23, 24] by their slope and intercept metadata values if present in the DICOM image files [21].

This component generates three outputs: (1) a 3D integer matrix containing the possibly normalized values from the DICOM images corresponding to those referenced by the DSO and possibly padded superior and inferior to it, (2) a 3D binary matrix same size as (1) with a value of 1 wherever the voxel is part of the segmentation and zero otherwise, and (3) a cell array with a cell per each DICOM image loaded containing all their image metadata.

#### *Pre-processing Stage*

The pre-processing stage receives the segmentation and image data loaded by the input stage and runs components that may modify these data. Each component in this stage runs sequentially using the output of each component as the input to the next. This stage then outputs the modified segmentation and intensity data. The engine currently implements components that can deform the DSO, bridge small gaps, eliminate all but the largest volume, and fill holes in the original segmentation. These can be enabled or disabled individually using the configuration file as desired.

**Segmentation Deformation** This sub-component can be used to apply morphological operations [25] to a supplied segmentation. This component takes as input parameters the desired operation (i.e., erosion or dilation) and the size of the spherical structuring element in millimeters.

**Topology Preservation** This sub-component merges disjoint components of segmentations that are within  $N$  voxels of each other using a morphological closing operation, with  $N$  being a user-defined parameter. This can be required, e.g., when the segmentation deformation sub-component splits a segmentation into two or more disjoint regions during erosion.

**Maximum Connected Volume Selection** This sub-component groups all connected voxels and, if there is more than one group, eliminates all but the group with the largest volume.

**Hole Filling** This sub-component finds unsegmented regions that are completely surrounded by segmented regions and includes them in the segmentation to be processed.

#### *Feature Computation Stage*

Components in the feature computation stage receive as inputs the (potentially) modified segmentation and image intensity values generated by the pre-processing stage and compute quantitative descriptors of the volumes of interest specified

by the segmentation and the data they contain. Each component in this stage runs independently of the others, i.e., the output of each component does not affect any other component. The output of this stage is an aggregated list of all the descriptors generated by the components. The engine currently implements components to extract the following features: size distribution features, intensity distribution, edge sharpness, local volume invariant integral (a metric of shape), surface roughness, sphericity, and Haralick's texture features.

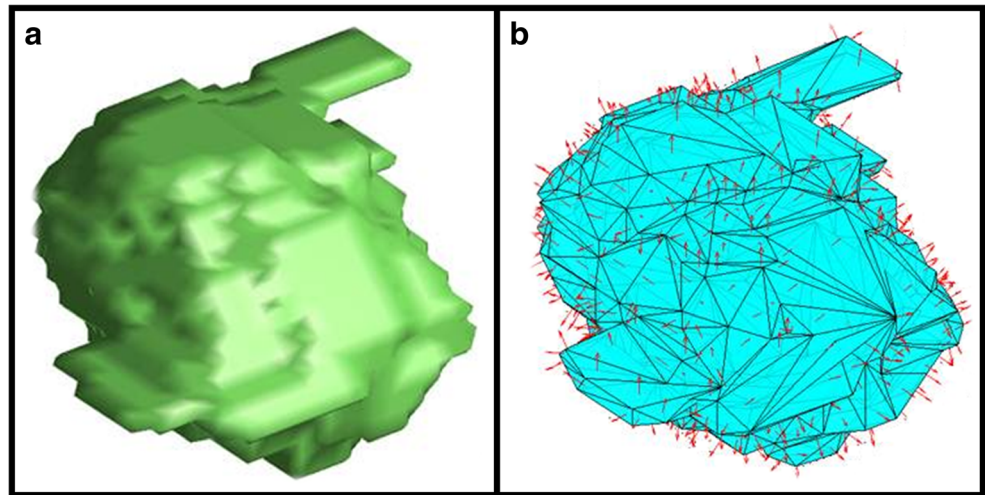
**Size Distribution Features** This component computes the size of the volume of interest. It reports its volume ( $\text{mm}^3$ ), its largest cross-sectional area ( $\text{mm}^2$ ), and its largest diameter (mm).

**Intensity Distribution Features** This component characterizes the global distribution of intensity values by computing summary statistics [26] (mean, standard deviation, geometric mean, harmonic mean, minimum value, maximum value, kurtosis, trimmed mean, skewness) of the intensity value distribution in the segmentation.

**Edge Sharpness Features** This component describes the statistics of the intensity changes between the interior and exterior of the segmentation. To define the surface of the segmentation, we use the MATLAB function *isosurface* [27] to create a triangle mesh that fits the segmentation. We then decimate the mesh using MATLAB's *reducepatch* function [28] to a value set in the configuration file (defaults to 600 triangles). Next, the component calculates normal vectors at the centroid of each triangle of the mesh, as shown in Fig. 2, and then trilinearly interpolates the voxel intensities along the normals every 1 from 5 mm outside to 5 mm inside the segmented volume. Finally, we fit a sigmoid function [29, 30], characterized by scale,  $S$ , and window,  $W$ , parameters, to the interpolated intensity points using MATLAB's *nlinfit* [31] (nonlinear regression) with the following parameters: max iterations = 300; TolFun =  $1\text{e-}8$ ; TolX =  $1\text{e-}8$ ; Display = off; DerivStep =  $\text{eps}^{(1/3)}$ ; FunValCheck = on; Robust = on; WgtFun = bisquare. The fit can fail when the values inside and outside the VOI are similar, e.g., when a lung tumor is next to the chest wall, or when the normal traverses noisy data or heterogeneous tissues; in these cases, the component does not record the  $S$  and  $W$  parameters. Finally, the component returns summary statistics of the distribution of  $S$  and  $W$  parameters, and the percentage of normals for which the fit failed.

**Local Volume Invariant Integral (LVII) Feature** This component characterizes the local curvature of the volume of interest (VOI) by generating a sphere of size  $N$  (configuration options in Appendix 1) centered at each point along the boundary of the VOI and calculating the percentage of the sphere filled by the VOI. This percentage will vary depending

**Fig. 2** **a** A mesh obtained by running the marching cubes algorithm on a segmented tumor. **b** The decimated mesh and the normal to each face



on the local curvature of the VOI; the component returns summary statistics (mean, standard deviation, geometric mean, harmonic mean, minimum value, maximum value, kurtosis, trimmed mean, skewness) of the distribution over all points on the surface of the VOI. Figure 3 shows some examples of LVII illustrated in 2D for simplicity.

**Roughness Feature** Surface roughness [32] characterizes the surface of the volume of interest (VOI) by computing surface deviations compared to a simulated smooth surface. To compute surface roughness, the QIFE generates a simulated smooth surface by first generating two new volumes using the morphological closing and opening operations, respectively, followed by combining them with a voxel-wise “and” operation. The size of the structuring element used in the morphological operations determines the cutoff frequency of this smoothing filter (configuration options found in Appendix 1). Second, the shortest distance between this smooth surface to each of the boundary voxel is calculated to find the deviations. This component returns summary statistics of the (average, standard deviation, kurtosis, skewness, root mean squared, maximum valley depth, maximum peak height, maximum

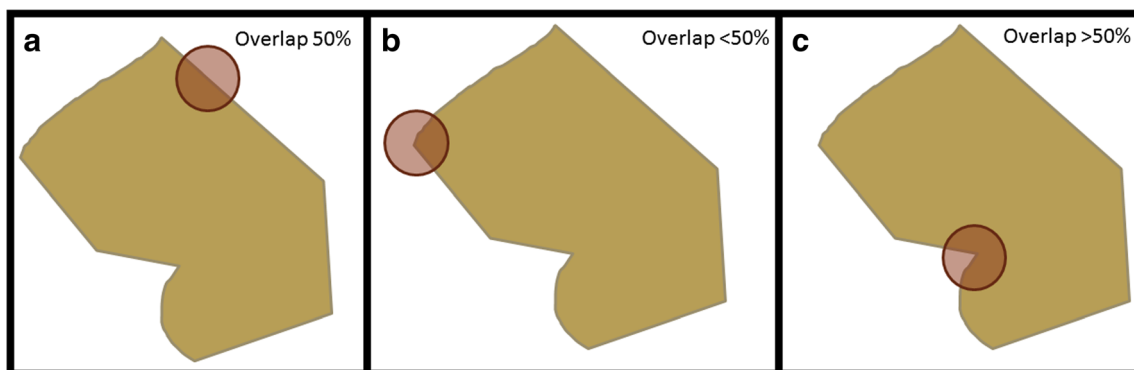
height of profile of this difference (Japanese Industrial Standard (JIS) [33], and German Institute for Standardization (DIN) [34]).

**Sphericity Feature** This component characterizes how similar the shape of the volume of interest (VOI) is to a sphere by calculating [35]:

$$\psi = \frac{\pi^{\frac{1}{3}}(6V)^{\frac{2}{3}}}{A}$$

where  $\psi$ ,  $V$ , and  $A$  are the sphericity, volume, and surface area of the segmentation, respectively.

**Haralick’s Texture Features** To characterize texture features, this component generates gray-level co-occurrence matrices (GLCM) [36] from the raw intensity values for each voxel within the segmentation for 13 directions corresponding to the 26 connected voxels [37] and at the distances specified by the configuration file (configuration options found in Appendix 1). The Haralick’s Features from each of these 13 matrices are aggregated (average, standard deviation, minimum and maximum) by distance, making the reported values



**Fig. 3** 2D Example of LVII. The percentage of the *circle* filled by the VOI changes depending on the local curvature

**Table 1** List of Haralick features extracted from the gray-level co-occurrence matrix. Details of the implementation of each of these features can be found in the references

Energy [33, 35]	Entropy [35]	Correlation [33, 35]
Contrast [33, 35]	Homogeneity [35]	Variance [33]
Sum of means [33]	Inertia [33, 35]	Cluster shade [35]
Cluster tendency [35]	Max probability [35]	Inverse variance [33, 35, 36]

rotational invariant. Table 1 lists the texture features we compute; this component outputs the aggregated statistics generated for each distance.

### Output Stage

The components in the output stage receive lists of feature values from the feature computation stage and generate user-readable files and images to be used in further analysis. The engine currently implements components that produce the following outputs per segmented VOI: a comma-separated feature value file, a run information file, and example cross-sectional images/segmentations.

**CSV Exporter** The CSV exporter writes the feature names (defined in each feature component configuration) and results generated by the feature-computation stage to a CSV file [38].

**Run Information Exporter** This component creates a file detailing the run date and time, the UUIDs of the files that were processed, the configuration, and the software version used to run the experiment. This file allows researchers to repeat the experiment, as it documents when it ran, the version of the QIFE and components it was using, the UUID of the objects that were processed, and the references defining each computation performed.

**Cross-sectional Image Generator** This component outputs an image file displaying the slice with the largest cross-

sectional VOI area in each of three directions (sagittal, coronal, axial), overlaid with the outline of the DSO. This image is useful for debugging and demonstration purposes as it gives a summary of the volume used to generate the results. Figure 3. 2D Example of LVII. The percentage of the circle filled by the VOI changes depending on the local curvature

Figure 4 shows an example of a processed lung nodule from a CT image volume.

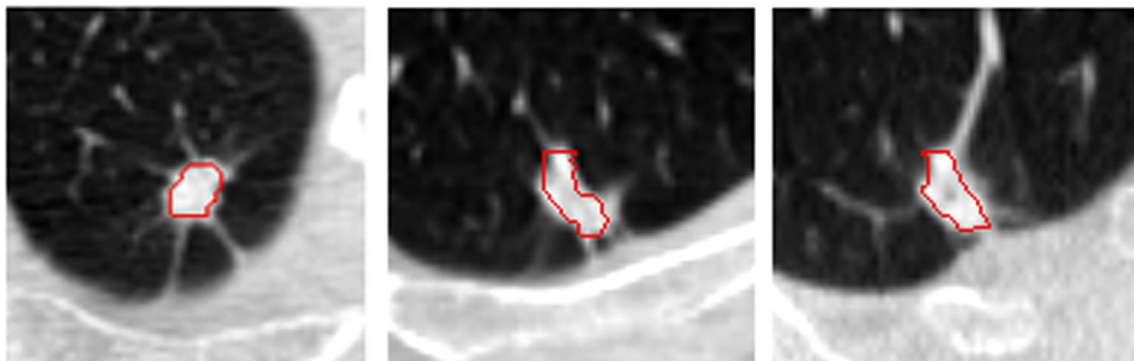
### Evaluation

To evaluate the computational efficiency of the QIFE as a function of parallelization choice, we extracted features from 108 CT scans presenting lung tumors ranging in size from 0.37 to 371.53 cm<sup>3</sup> (mean = 31.35 cm<sup>3</sup>; s.d. = 66.78 cm<sup>3</sup>). We ran the QIFE with all features active at their default settings. We repeated the experiment for different parallelization levels (none, object, feature, internal) and stored the total running time and maximum memory usage for each run.

### Results

#### Speed and Memory Tests

Table 2 shows the total run time and the peak memory usage for each parallelization level. Total run time was obtained using MATLAB's profiler; peak memory usage was obtained using Sysinternals Process Monitor.



**Fig. 4** Output from the cross-sectional image generator showing, from left to right, axial, coronal, and sagittal cross-sections of a lung CT-scan and the tumor boundaries given by the supplied segmentations (solid red

line). The location of each cross-section corresponds to the maximum cross-sectional area for each orientation

**Table 2** Total run-time (seconds) and peak memory usage (GB) for the QIFE while processing 5 CT lung scans presenting masses from 0.81 to 59.04 cm<sup>3</sup> (mean 14.11<sup>3</sup>, s.d. 25.21 cm<sup>3</sup>). Total run time was obtained using MATLAB's profiler, peak memory usage was obtained using

Parallelization level	Total time (hh/mm)	Peak memory (GB)
None	2:12	2.53
Object	1:04	7.90
Feature	2:00	6.98
Internal	2:07	4.53

Sysinternals Process Monitor. We used a Clevo P370EM laptop, containing an Intel I7-3740QM (4 cores, all available to the engine) CPU clocked at 2.70GHz and 16.0 GB of RAM and a solid-state drive (Samsung EVO 850)

Running without parallelization resulted in the longest running time with the least memory, while parallelizing at the object level took the least time but required the most amount of memory, as multiple copies of the engine must be memory-resident at the same time. Feature-level parallelization did not improve speed much compared to no parallelization. This is to be expected because (a) there is overhead, which is not reduced by parallelization, and (b) we used only four cores and of the nine feature classes executed in parallel; some are computationally intensive and others are significantly less so. In this scenario, it is unlikely that the computationally intense feature classes will always be processed in parallel, effectively limiting the expected performance increase.

### Feature Comparison

To demonstrate the features of the QIFE, we processed four lung nodules of different in shapes, sizes, and compositions, as shown in Fig. 5. Nodules A and C are mostly solid, while nodules B and D present ground glass components (Table 3). Comparison of chosen features between the four nodules is shown in Fig. 5. Nodules A and C are solid tumors, while nodules B and D are part solid and part ground glass. As expected, solid tumors (nodules A and C) present a higher mean intensity and lower texture variance (mean GLCM variance) than the tumors with partial ground glass (nodules B and D). Nodules A and B show a high surface variation, giving them higher roughness values compared to nodules C and D. Finally, nodules A, B, and C are mostly convex, while nodule B presents multiple concavities, therefore having the lowest sphericity. It shows selected feature output by the QIFE for these four nodules. The volume feature shows that nodules A and B are similar in size, while nodule C and nodule D are much smaller. Nodules B and D present a lower mean intensity, as expected due to the lower density of the ground-glass portion of the tumor. The roughness feature characterizes high frequency surface variations; as Fig. 5 shows, Nodules A and B surfaces are less smooth, resulting in a higher roughness values. The

sphericity feature is proportional to the ratio of surface area to volume and is therefore higher for more spherical objects, resulting in higher values for nodules A, C, and D. Finally, tumors with ground-glass components (nodules B and D) have more heterogeneous density and therefore have lower mean intensities and larger texture features (GLCM).

### Deployment and Collaboration

#### Code Repository

The QIFE Version 1.0 has been released as a BSD-licensed [39] open-source project in Github, (<https://www.github.com/riipl/3dpipeline>). Collaboration from the research community is encouraged either by contributing code, creating pull requests, or by testing and reporting. The complete source code can be downloaded from the repository; Mathworks MATLAB 8.6 R2015b is necessary to run the engine.

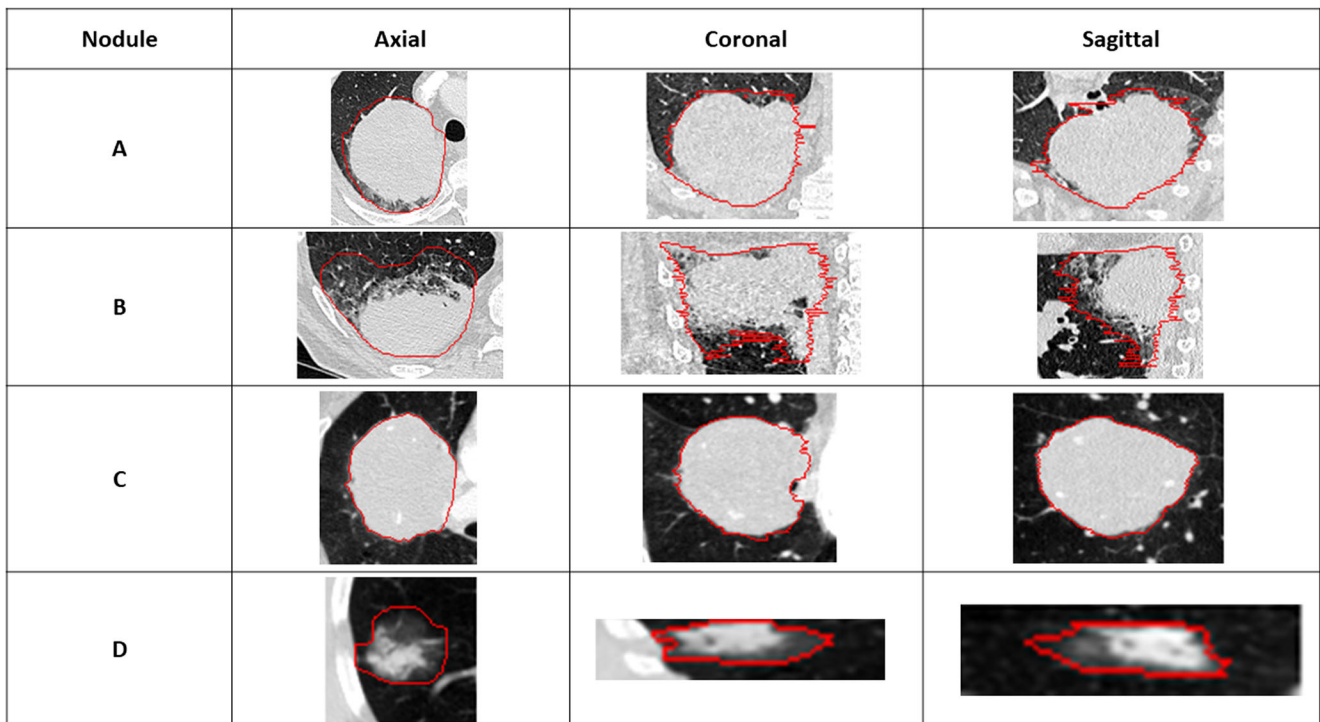
#### Docker

We have also released a compiled version of the engine. The compiled version of the engine requires that the correct release (9.0.1 at the time of this publication) of MATLAB's Compiler Runtime (MCR) be present in the host computer. To simplify deployment, we have provided a Docker container hosted in Dockerhub ([https://hub.docker.com/r/riipl/3d\\_qifp/](https://hub.docker.com/r/riipl/3d_qifp/)), containing the correct MCR and the compiled QIFE, making it completely portable, and avoiding any possible mismatch between the QIFE and the user's computer.

### Discussion

#### Current Use

The QIFE is currently deployed as part of an internal web application, known as the Quantitative Image Feature



**Fig. 5** Four example nodules: Each row shows three cross-sectional views of a different nodule. The location of each slice corresponds to the maximum cross-sectional area for each orientation

Pipeline (QIFP), at Stanford University. A web interface was created to act as a wrapper between the Docker module containing the QIFE and the users. Researchers can upload their scans and segmentations, and the application returns a CSV with their results after processing.

The QIFE was also used in a study of feature sensitivity to segmentation to evaluate whether digital biopsies could be used to speed up segmentation while retaining information [40]. In this study, the QIFE used pre-processing components that deformed the original 3D digital biopsies to simulate multiple manual-segmentations.

Finally, the QIFE was also used in a study to compare feature extraction implementations and showed reasonable correlations with features of the same type [41].

### Limitations

One limitation of QIFE is that it does not use the latest advances in GPU technology to speed up computation. We decided to forego this optimization as access to computers with powerful GPUs is not always a possibility, and we wanted to make the QIFE universally compatible. This limitation can be addressed at the component level. While the engine logic does not use the GPU, our components could implement GPU functionality in the future. Components using the GPU should either have a fallback to use the CPU in case a GPU is not present or show in their documentation that a GPU is required for them to function.

**Table 3** Comparison of chosen features between the four nodules shown in Fig. 5. Nodules A and C are solid tumors while nodules B and D are part solid, part ground glass. As expected, solid tumors (nodules A and C) present a higher mean intensity, and lower texture variance (mean GLCM variance) than the tumors with partial ground

glass (nodules B and D). Nodules A and B show high surface variation, giving them a higher roughness values compared to nodules C and D. Finally, nodules A, B, and C are mostly convex, while nodule B presents multiple concavities, therefore having lowest sphericity

Feature	Volume (mm <sup>3</sup> )	Mean intensity (HU)	Roughness JIS @1 mm (mm)	Sphericity	Mean GLCM variance @2 mm
Nodule A	371,856.44	−21.56	6.83	0.57	3.56
Nodule B	361,845.55	−215.49	9.07	0.35	10.39
Nodule C	104,399.57	−10.54	4	0.65	1.96
Nodule D	15,233.23	−525.52	1.53	0.73	6.93

Another limitation is that currently the QIFE only computes 3D features, while there are multiple studies in the literature showing the usefulness of 2D features. For example, in previous studies, our group created a 2D feature extraction engine and successfully used it for image retrieval [42] and to generate features that correlate with gene expression [43]. Integrating other features such as these could expand the applicability of the QIFE.

## Conclusions

In this paper, we presented the QIFE, a portable, open-source feature extraction framework that focuses on modularity, standards, parallelism, provenance, and integration. We describe the different stages of QIFE and the components that can be loaded and executed at each stage.

Integration with existing research workflows can be implemented by changing the input-stage and output-stage components. By separating the input-output logic from feature processing, researchers can easily add MATLAB code to read other input formats, generate new features, and implement new output types.

The QIFE also outputs information about the configuration and components involved in generating a set of results. By storing these log files, provenance of result sets is recorded and experiments can be repeated or compared across datasets with confidence on the similarity of the processing engine.

Finally, we also tested the QIFE at different parallelization levels measuring memory and time taken for each setting. Speed-ups can be obtained by parallelizing execution at the cost of memory usage. Different parallelization levels provide different trade-offs, and the optimal setting will depend on the size and composition of the dataset to process and the number of processing cores available.

**Acknowledgements** This research was funded in part by the following grants from the National Institutes of Health: R01 CA160251, U24 CA180927, U01 CA187947, and U01-CA190214.

**Funding** This work was supported by the National Institutes of Health Grants R01 CA160251, U01 CA187947, U01-CA190214, and U24 CA180927.

## Compliance with Ethical Standards

**Conflict of interest** Dr. Napel is a consultant for Carestream, Inc. and is on the scientific advisory boards of Echo Pixel, Inc., Fovia, Inc., and RadLogics, Inc.

## Appendix 1 Configuration Parameters for each component

### Global

Parameter name	Default Value	Description
inputRoot	N/A	Root directory for input. (All input folders are relative to this directory)
outputRoot	N/A	Root directory for output. (All output folders are relative to this directory)
parallelMode	none	What parallelization strategy to use (None, Object, Feature, Internal)
numberOfProcessors	max	If parallelMode is other than None, then the software creates a processing pool with numberOfProcessors processors. “Max” uses all available
uidToProcess	all	A list of UID to be processed by the QIFE. If “all” it processes all volumes loaded by the input stage

### Input Stage

#### *DSO/DICOM loader component*

Parameter name	Default value	Description
dicomFolder	N/A	Folder relative to inputRoot where the DICOM sets are stored
dsoFolder	N/A	Folder relative to inputRoot where the DSOs are stored
recomputeHashTable	false	Computes the UID hash tables even if a cache index is found in the directory
saveHashTable	true	Saves a cache of the UID hashtables in their root directories
padding	10	Millimeters to go outside the VOI when loading the VOI

### Preprocessing stage

#### *Segmentation deformation*

Parameter name	Default value	Description
operation	N/A	Operation to perform in the VOI (“erosion” or “dilation”).
sizeOfElement	N/A	Size of the ball used to perform the operation specified.

*Topology preservation*

Parameter Name	Default value	Description
sizeOfGap	N/A	Maximum size of gaps to be bridged.

*Maximum connected volume selection*

Parameter name	Default value	Description
connectivity	26	What connectivity determines that a voxel is part of the same volume (Possible values: 6, 18, 26)

*Hole filling*

Parameter name	Default value	Description
connectivity	26	What connectivity determines that a voxel is part of the same volume (Possible values: 6, 18, 26)

**Feature Computation Stage***Size distribution features*

Parameter name	Default value	Description
featureRootName	size	The prefix to add to all results generated by this component

*Intensity distribution features*

Parameter name	Default value	Description
featureRootName	intensity	The prefix to add to all results generated by this component

*Edge sharpness features*

Parameter name	Default value	Description
featureRootName	edge	The prefix to add to all results generated by this component
normalLength	5	Length in millimeters of normals in each direction.
numberOfNormals	600	Number of normals after triangulation and decimation
numberOfSamplingPoints	21	Number of intensity samples along a normal

*Local volume invariant integral (LVII) feature*

Parameter name	Default value	Description
featureRootName	lvii	The prefix to add to all results generated by this component
sphereRadius	1,2,3,4,5	List of radii for the Sphere used to calculate intersections separated by commas

*Roughness feature*

Parameter name	Default value	Description
featureRootName	roughness	The prefix to add to all results generated by this component
patchSize	3	Maximum distance in mm for a voxel to be considered in the same patch when roughness is computed

*Sphericity feature*

Parameter name	Default value	Description
featureRootName	sphericity	The prefix to add to all results generated by this component

*Haralick's texture features*

Parameter name	Default value	Description
featureRootName	haralick	The prefix to add to all results generated by this component
distance	1,2,3	Distances in mm at which to calculate the GLCM.
grayLevels	16	Number of gray levels to quantify intensity values to.

**Output stage***CSV Exporter*

Parameter name	Default value	Description
filename	out.csv	Filename for the csv file relative to out folder
Transpose	false	Transpose the data in the CSV (headers in the first column)

*Run information exporter*

Parameter name	Default value	Description
filename	Info.txt	Filename for the run information file

*Cross-sectional image generator*

Parameter name	Default value	Description
folderRoot	.	Folder relative to output root where to save the generated images. By default it uses the output root folder.
windowLevelPreset	ctLung	Window and Level preset. By default it assumes Lung CT

*Reference Generator*

Parameter name	Default value	Description
filename	references.bib	Filename for the bib file relative to out folder

**Appendix 2 Example Configuration File**

The configuration file defines which components are loaded in each stage, and can override default parameters (shown in Appendix 1). The file follows the following syntax:

Category|ParameterName = VALUE (The separator is a pipe "|").

Category can be:

- global (sets parameter for the whole engine)
- input (sets parameter for the input stage)
- preprocessing (sets parameters for the preprocessing stage)
- featureComputation (sets parameters for the feature computation stage)
- output (sets parameters for the output stage), or
- a specific component name to override its defaults.

Multiple parameters can be set using comma as a separator.

Comments are defined with a semicolon at the beginning of a line. The following is an example configuration file:

```
; Global Parameters
; Disables parallel mode
global|parallelMode = "none"
; Use the maximum number of processors
global|numberOfProcessors = "max"
; Process all files included in the input directory
```

```
global|uidToProcess = "all"
; Components to load
; Input components to load
input|component = "dsoLoader"
; Preprocessing components to load
preprocessing|components =
"maximumConnected,holeFilling"
; Feature computation components to load
featureComputation|components =
"information,size,intensity,sphericity,roughness,
edgeSigmoidFitting,lvii,glcm,connectedRegions"
; Output components to load
output|components = "csvOutput,
maxAreaImage,references"
; Component parameters to override (See Appendix 1 for
definition)
; Number of Normals in the Edge Sigmoid Feature
edgeSigmoidFitting|numberOfNormals = 1200
; Window and Level preset
maxAreaImage|windowLevelPreset = "ctLung"
```

**References**

1. Lambin P, Rios-Velazquez E, Leijenaar R, Carvalho S, van Stiphout RGP, Granton P, Zegers CML, Gillies R, Boellard R, Dekker A et al.: Radiomics: Extracting more information from medical images using advanced feature analysis. *Eur J Cancer* 48(4):441–446, 2012
2. Aerts HJWL, Velazquez ER, Leijenaar RT, Parmar C, Grossmann P, Carvalho S, Bussink J, Monshouwer R, Haibe-Kains B, Rietveld D, Hoebbers F, Rietbergen MM, Leemans CR, Dekker A, Quackenbush J, Gillies RJ, Lambin P, Cavalho S, Bussink J et al.: Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach. *Nat Commun* 5:4006, 2014
3. Kumar V, Gu Y, Basu S, Berglund A, Eschrich SA, Schabath MB, Forster K, Aerts HJWL, Dekker A, Fenstermacher D, Goldgof DB, Hall LO, Lambin P, Balagurunathan Y, Gatenby RA, Gillies RJ et al.: Radiomics: The process and the challenges. *Magn Reson Imaging [Internet]* 30(9):1234–1248, 2012. <https://doi.org/10.1016/j.mri.2012.06.010>
4. Coroller TP, Grossmann P, Hou Y, Rios Velazquez E, Leijenaar RTH, Hermann G, Lambin P, Haibe-Kains B, Mak RH, Aerts HJWL: CT-based radiomic signature predicts distant metastasis in lung adenocarcinoma. *Radiother Oncol [Internet]* 114(3):345–350, 2015. <https://doi.org/10.1016/j.radonc.2015.02.015>
5. Parmar C, Velazquez ER, Leijenaar R, Jermoumi M, Carvalho S, Mak RH, Mitra S, Shankar BU, Kikinis R, Haibe-Kains B, Lambin P, Aerts HJWL: Robust radiomics feature quantification using semiautomatic volumetric segmentation. *PLoS One* 9(7):1–8, 2014
6. Gatenby RA, Grove O, Gillies RJ: Quantitative imaging in cancer evolution and ecology. *Radiology [Internet]* 269(1):8–15, 2013 Available from: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3781355&tool=pmcentrez&rendertype=abstract>
7. Leijenaar RTH, Carvalho S, Velazquez ER, van Elmpot WJC, Parmar C, Hoekstra OS, Hoekstra CJ, Boellaard R, Dekker A, Gillies RJ, Aerts HJWL, Lambin P: Stability of FDG-PET Radiomics features: An integrated analysis of test-retest and inter-

- observer variability. *Acta Oncol (Madr)* [Internet] 52(7):1391–1397, 2013 Available from: <http://www.ncbi.nlm.nih.gov/pubmed/24047337>
8. Machine Learning | Microsoft Azure [Internet]. Available from: <https://azure.microsoft.com/en-us/services/machine-learning/>
9. Ludäscher B, Altintas I, Berkley C, Higgins D, Jaeger E, Jones M, Lee EA, Tao J, Zhao Y: Scientific workflow management and the Kepler system. *Concurr Comput Pract Exp* 18(10):1039–1065, 2006
10. Parker SG, Johnson CR: SCIRun: A Scientific Programming Environment for Computational Steering [Internet]. In: *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*. New York: ACM, 1995. Available from: <http://doi.acm.org/10.1145/224170.224354>
11. Hull D, Wolstencroft K, Stevens R, Goble C, Pocock MR, Li P, Oinn T: Taverna: A tool for building and running workflows of services. *Nucleic Acids Res* 34(WEB. SERV. ISS):729–732, 2006
12. Taylor I, Shields M, Wang I, Harrison A: The triana workflow environment: Architecture and applications. *Work e-Science Sci Work Grids*:320–339, 2007
13. Zhang L, Fried DV, Fave XJ, Hunter LA, Yang J, Court LE: IBEX: An open infrastructure software platform to facilitate collaborative work in radiomics. *Med Phys* [Internet] 42:1341–1353, 2015 Available from: <http://scitation.aip.org/content/aapm/journal/medphys/42/3/10.1118/1.4908210>
14. van Griethuysen J, Fedorov A, Parmar C, Hosny A, Aucoin N, Narayan V, Beets-Tan R, Fillion-Robin JC, Pieper S, Aerts HJWL: Computational Radiomics System to Decode the Radiographic Phenotype. *Accepted Cancer Res* 2017. <https://github.com/Radiomics/pyradiomics>
15. Boettiger C: An introduction to Docker for reproducible research. *ACM SIGOPS Oper Syst Rev* [Internet] 49(1):71–79, 2015 Available from: <http://arxiv.org/abs/1410.0846>
16. Ince DC, Hatton L, Graham-Cumming J: The case for open computer programs. *Nature* [Internet] 482(7386):485–488, 2012 Available from: <http://www.ncbi.nlm.nih.gov/pubmed/22358837>
17. Bitzer J, Schröder PJH: Bug-fixing and code-writing: The private provision of open source software. *Inf Econ Policy* 17(3):389–406, 2005
18. Aberdour M: Achieving quality in open source software. *IEEE Softw* [Internet] (September):58–64, 2007 Available from: <http://www.computer.org/portal/web/csdl/doi/10.1109/MS.2007.2>
19. Simmhan YL, Plale B, Gannon D: A survey of data provenance in e-science [internet]. *SIGMOD Rec.* 34(3):31–36, 2005 Available from: [http://doi.acm.org/10.1145/1084805.1084812%5C, http://dl.acm.org/ft\\_gateway.cfm?id=1084812&type=pdf](http://doi.acm.org/10.1145/1084805.1084812%5C, http://dl.acm.org/ft_gateway.cfm?id=1084812&type=pdf)
20. Davidson SB, Freire J: Provenance and scientific workflows. *Proc 2008 ACM SIGMOD Int Conf Manag data - SIGMOD '08* [Internet], 2008, p 1345. Available from: <http://www.scopus.com/inward/record.url?eid=2-s2.0-57149126952&partnerID=tZOTx3y1>
21. Mildenberger P, Eichelberg M, Martin E: Introduction to the DICOM standard. *Eur Radiol* 12(4):920–927, 2002
22. DICOM Standards Committee WG 17 (3D). Supplement 111: Segmentation Storage SOP Class. In: *Digital Imaging and Communications in Medicine (DICOM)*. Rosslyn, Virginia, 2006, p 22209
23. Liu B, Zhu M, Zhang Z, Yin C, Liu Z, Gu J: Medical image conversion with DICOM. *Can Conf Electr Comput Eng*:36–39, 2007
24. Riesmeier J, Eichelberg M, Jensch P: An approach to DICOM image display handling the full flexibility of the standard's specification. *Med Imaging 1999 Image Disp* 3658(February):363–9, 1999
25. Jonker PP: Morphological operations on 3D and 4D images: From shape primitive detection to skeletonization. In: *Lecture Notes in Computer Science*. 2000, pp 371–91
26. Norris N: General means and statistical theory. *Am Stat* [Internet] 30(1):8–12, 1976 Available from: <http://www.tandfonline.com/doi/abs/10.1080/00031305.1976.10479125>
27. Mathworks. isosurface [Internet]. Matlab Ref. [cited 2016 Oct 19]. Available from: <https://www.mathworks.com/help/matlab/ref/isosurface.html>
28. reducepatch [Internet]. Mathworks MATLAB 2016a Doc. Available from: <https://www.mathworks.com/help/matlab/ref/reducepatch.html>
29. Han J, Moraga C: The influence of the sigmoid function parameters on the speed of backpropagation learning. *From Nat to Artif Neural Comput* [Internet] 930:195–201, 1995. doi:[https://doi.org/10.1007/3-540-59497-3\\_175](https://doi.org/10.1007/3-540-59497-3_175)
30. Xu J, Napel S, Greenspan H, Beaulieu CF, Agrawal N, Rubin D: Quantifying the margin sharpness of lesions on radiological images for content-based image retrieval. *Med Phys* 39(9):5405–5418, 2012
31. nlinfit [Internet]. Mathworks MATLAB 2016a Doc.2016. Available from: <https://www.mathworks.com/help/stats/nlinfit.html>
32. Degarmo EP, Black J, Kohser RA: *Materials and processes in manufacturing*, 9th edition. Hoboken: Wiley, 2003
33. Definition and Designation of Surface Roughness. JIS B 0601. Japanese Industrial Standard, 1982
34. Surface Texture Symbols [Internet]. The American Society of Mechanical Engineers, 1996. Available from: <https://www.asme.org/products/codes-standards/y1436m-1996-surface-texture-symbols>
35. Wadell H: Volume, shape, and roundness of quartz particles. *J Geol* [Internet] 43(3):250–280, 1935 Available from: <http://www.journals.uchicago.edu/doi/10.1086/624298>
36. Haralick RMM, Shanmugam K, Dinstein IH: Textural Features for Image Classification. *IEEE Trans Syst Man Cybern* [Internet] [cited 2010 Nov 6];SMC-3(6):610–21, 1973. Available from: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=4309314](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4309314)
37. Kong TY, Roscoe AW, Rosenfeld A: Concepts of digital topology. *Topol Appl* [Internet] 46(3):219–262, 1992 Available from: <http://www.sciencedirect.com/science/article/pii/016686419290016S>
38. Shafranovich, Y.: Common Format and MIME Type for Comma-Separated Values (CSV) File, RFC 4180, October 2005. <https://tools.ietf.org/html/rfc4180>. Accessed 2017-05-01
39. Opensource.org. The BSD 2-Clause License [Internet]. Licenses 2016. Available from: <https://opensource.org/licenses/BSD-2-Clause>
40. Echegaray S, Nair V, Kadoch M, Leung A, Rubin D, Gevaert O, Napel S: A rapid segmentation-insensitive “digital biopsy” method for Radiomic feature extraction: Method and pilot study using CT images of non-small cell lung cancer. *Tomography* [Internet] 2(4):283–294, 2016. Available from: <http://digitalpub.tomography.org/i/763956-vol-2-no-4-dec-2016/52>
41. Kalpathy-Cramer J, Mamomov A, Zhao B, Lu L, Cherezov D, Napel S, Echegaray S, McNitt-Gray M, Lo P, Sieren JC, Uthoff J, Dilger SKN, Driscoll B, Yeung I, Goldgof D: Radiomics of lung nodules: a multi-institutional study of robustness and agreement of quantitative imaging features. *Tomography* 2(4):430–437, 2016. <https://doi.org/10.18383/j.tom.2016.00235>
42. Napel SA, Beaulieu CF, Rodriguez C, Cui J, Xu J, Gupta A, Korenblum D, Greenspan H, Ma Y, Rubin DL: Automated retrieval of CT images of liver lesions on the basis of image similarity: Method and preliminary results. *Radiology* [Internet] 256(1):243–252, 2010. <https://doi.org/10.1148/radiol.10091694>
43. Gevaert O, Mitchell LA, Achrol AS, Xu J, Echegaray S, Steinberg GK, Cheshier SH, Napel S, Zaharchuk G, Plevritis SK: Glioblastoma Multiforme: Exploratory Radiogenomic analysis by using quantitative image features. *Radiology* [Internet] 273(1):168–174, 2014. <https://doi.org/10.1148/radiol.14131731>